Vol.2013-AL-144 No.3
2013/5/17

長さ極大な群れパターンを軌跡集合から効率良く発見するアルゴリズム

有村 博紀[1]　耿 暁亮[1]　宇野 毅明[2]

概要：本論文では，(Gudmundsson and van Kreveld, ACM GIS 2006; Benkert, Gudmundsson et al., Computational Geometry, 41(111), 2008) によって導入された群れパターン（flock pattern）の軌跡集合データからの発見問題について考察する．L_2 ノルムをもつ 2 次元平面上の軌跡集合に対して，最大幅と最小長さが固定の時に，最大数の軌跡を含む群れパターンを見つける問題は，直径の $2-\delta$ を許しても NP 困難である．その一方で，最小軌跡数と最大幅が固定で，長さ最大の群れパターン一つは多項式時間で求まることが示されている．これに対して，本論文では，L_∞ ノルムをもつ 2 次元平面上の n 本の長さ T の軌跡の集合に対して，最大幅 r かつ最小長さ k 以上で，長さ（方向の区間）極大の群れパターンを $O(mnT^2)$ 遅延と $O(m^2)$ 領域ですべて列挙する多項式時間遅延かつ多項式領域の列挙アルゴリズムを与える．ここに，$m = |X|$ は列挙されるパターンのサイズ（それが含む軌跡数）である．

Efficient Algorithms for Finding All Length-Maximal Flock Patterns from a Set of Trajectories

Hiroki Arimura[1]　Xiaoliang Geng[1]　Takeaki Uno[2]

Abstract: In this paper, we study the problem of finding a class of spatio-temporal patterns called (m,k)-flock patterns (Gudmundsson and van Kreveld, Proc. ACM GIS'06; Benkert, Gudmundsson, Hubner, Wolle, Computational Geometry, 41:11, 2008), which represent a groups of moving objects close each other within width at most r under L_∞-norm in a given time segment of length at least k, in a collection of 2-dimensional trajectories. For max-width $r > 0$, min-length k, and a collection S of n trajectories of legnth T, the proposed algorithm finds all length-maximal (m,k) flock patterns in an input collection of trajectory data in $O(pnT^2)$ delay and $O(p^2)$ space, $p = |X|$ is the size of ID set being enumerated. We also present a practical improvement using geometric indexes.

1. Introduction

1.1 Background

By rapid increase of a massive amount of trajectory data have been accumulated, the research on trajectory mining, i.e., efficient methods for extracting interesting patterns and rules from collections of trajectory data, has attracted a great deal of attention for recent years [4], [7].

A *trajectory database* on the time domain $\mathbb{T} = \{1,\dots,T\}$ is a set $S = \{\, s_i \,|\, i = 1,\dots,n \,\}$ of trajectories for n moving objects, where each trajectory is a a sequence $s_i = s_i[1]\cdots s_i[T]$ of T points on the 2-dimensional space $\mathbb{R}^2$ and its ID i is drawn from a set of n identifiers $ID = \{1,\dots,n\}$. For instance, GPS-trajectories of wild animals, walking people with Wifi device, Probe car data are examples of such trajectory databases [7].

1　北海道大学大学院情報科学研究科
Hokkaido University, Graduate School of Information Science and Technology, {wasa, arim}@ist.hokudai.ac.jp
2　国立情報学研究所
National Institute of Informatics, uno@nii.jp

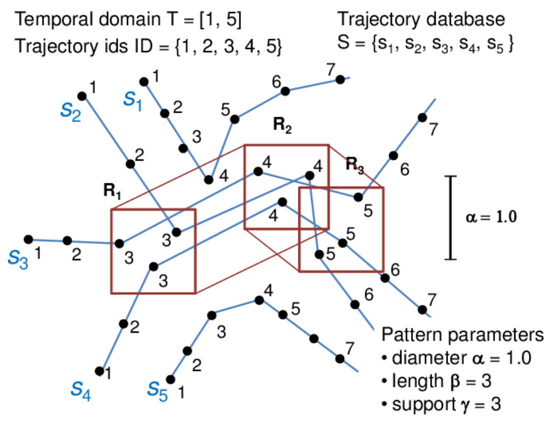

図 1 Examples of a trajectory database S_1 consisting of five trajectories $\mathsf{s}_1, \ldots, \mathsf{s}_5$ with ID set $ID = \{1, \ldots, 5\}$, and a flock pattern $P_1 = (X = \{2,3,4\}, [t_3, t_5])$ with diameter $r = 1.0$, length $k = 3$, and support $m = 3$, where each line indicates a trajectory, the figures associated with points their time, and boxes indicate $r{\times}r$ rectangles forming flock pattern P_1.

For such trajectory databases, Laube, Kreveld, and Imfeld [10] and Gudmundsson and van Kreveld [8] introduced a class of spatio-tempral patterns, called *flock patterns* (See Fig. 1). For a positive number $r > 0$, called a max-width, and non-negative integers $k, m \geq 0$, called min-len and min-sup, an (r,k,m)*-flock pattern* in a trajectory database S is a pair $P = (X, [b,e])$ of a set X of trajectory ids and a time interval $I = [b,e]$ on $\mathbb{T}$ that represents a set of at least m moving objects that move together with mutually distance at most r in L_∞-norm, that is, the largest of the x- and y-distances, along a continuous interval I of length at least k time points. Flock patterns are useful in detecting a group of highly correlated entities combining spatio-tempral features.

In this paper, we focus on pattern mining approach that makes complete mining of all patterns in an input database that satisfy a given set of constraints, as in frequent pattern mining [12], [13], [17]. Particularly, we study the problem of finding all (r,k)-flock patterns [*1] ((r,k)-FP for short), thus with max-width and min-len constraints, from an input database of n trajectories of length T.

1.2 Main results

1.2.1 Classes of length-maximal flock patterns

Along the above line of research on closed patterns [1], [13], by extending (r,k)-FPs above, we first introduce the classes of RFPs and UFPs of closed flock patterns as follows. Given a maximum width parameter r, it is often useful to find all (r,k)-flock patterns $P = (X, [b,e])$ whose time interval $[b,e]$ are extended leftward or rightward as long as possible along time line with preserving the diameter r of trajectories, instead of separately discovering all flock patterns of various lengths above length constraint $\geq k$.

Then, we introduce the classes of *rightward length-maximal* and *unrestricted length-maximal* (r,k)-flock patterns, denoted by (r,k)-RFPs and (r,k)-UFPs, respectively, where a RFP can be extended rightward at given start time b, while an UFP can be extended either the start time b leftweard or the end time e rightward yielding more flexibility and compression.

Unlike Gudmundsson and van Kreveld's longest-duration (r,k,m)-flock patterns [8] for which a search problem for a pattern is NP-hard, the classes of RFPs and UFPs allow polynmial time computation of search due to the local nature of *maximality* than the global nature of *maximumlity*.

1.2.2 Polynomial delay and space algorithms

First of all as a main result, we present a depth-first search mining algorithm RFPM (Algorithm 2) that finds all rightward length-maximal (r,k)-flock patterns P, or (r,k)-RFPs, in a given trajectory database S of n transactions of length T in $O(mnT)$ delay (time per pattern) and $O(m^2)$ space, where $m = |X|$ is the number of trajectories that the discovered P contains. Actually, BasicFPM is a polynomial-delay and polynomial space algorithm for (r,k)-RFPs without using any tabulation to avoid duplicates (Theorem 1). We note that our algorithm works in the d-dimensional continuous space with large $d \geq 2$ by adding a factor of $O(d)$.

Next, for the class of (r,k)-UFPs (unrestricted length-maxmal flock patterns), for which extension is possible for both sides, we give a characterization of UFPs using a technique, called *leftward extension check*. Using this property, we show that a modification of the algorithm BasicFPM finds all (r,k)-UFPs in $O(mnT^2)$ delay and $O(m^2)$ space, where $m = |X|$ is the number of trajectories that the discovered P contains. (Theorem 2).

1.3 Related work

There are two lines of researches on trajectory mining: trajectory clustering [4], [11] and disk-based trajectory pattern mining [5], [7], [10].

The study of flock pattern mining started in the latter context [5], [9], [10]. Gudmundsson and van Kreveld [8]

[*1] Our (r,k,m)-flock patterns use L_∞-distance on $\mathbb{R}^2$ to define the diameter $\leq m$, while the original (r,k,m)-flock patterns of Benkert *et al.* [5] used L_2-distance on $\mathbb{R}^2$.

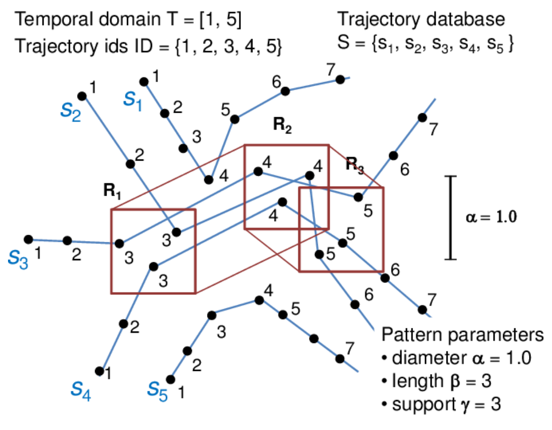

図 2 Examples of a trajectory database S_1 on $ID = \{1, \ldots, 5\}$ and $\mathbb{T} = [1, 7]$ and a $(1.0, 2, 2)$-flock pattern $P_1 = (X_1, I_1) = (\{2, 3, 4\}, [3, 5])$ with diameter $||\, P_1 \,||_\infty{}^{S_1} \leq 1.0$, length $len(P_1) = 3$, and support $supp(P_1) = 3$. Here, each line indicates a trajectory and the numbers attached to points are time stamps.

showed that the problem of finding at least one length-*maximum* (r, k, m)-flock pattern is NP-hard, while they gave an efficient 2-approximation algorithm, although it does not make complete enumeration of all flock patterns. Benkert *et al.* [5] proposed an $(2+\varepsilon)$ approximation algorithm for fixed-length flock patterns, whose running time is polynomial in m and $\frac{1}{\varepsilon}$, but exponential in the length k of a pattern, thus not polynomial delay.

Most closely related work is the work by Vieira, Bakakov, and Tsotras [14], who took pattern mining aproach at the first time. They presented an algorithm that finds all (r, k, m)-flock patterns by systematically combining discovered clusters by depth-first search using the idea of intersection. Unfortunately, their algorithm is neither polynomial delay nor polynomial space from theoretical point of view.

1.4 Organization

Sec.2 gives definitions, Sec.3 presents our algorithms, and Sec.**??** shows experimental results. Finally, Sec.4 concludes.

2. Preliminaries

2.1 Basic definitions

Let $\mathbb{R}$ and $\mathbb{N}$ be the set of all real numbers and all non-negative integers, respectively. For integers a, b $(a \leq b)$, we denote by $[a, b] = \{a, a+1, \ldots, b\}$ the discrete interval between a and b. If $a \leq b$ are real numbers, then $[a, b]$ denotes a continuous interval in $\mathbb{R}$ as usual. For a set A, $|A|$ denotes the cardinality of A, and A^* denotes the set of all possibly empty, finite sequences over A.

2.2 Trajectory Database

Let n and $T \geq 0$ are pre-determined nonnegative integers, which indicate the number of moving objects and the maximum value for discrete time stamps, respectively. Let $\mathbb{R}^2$ be the 2-dimensional continuous space, or the *plane*.

A *trajectory database* on the space domain $\mathbb{R}^2$ and the time domain $\mathbb{T} = \{1, \ldots, T\}$ is a finite set

$$S = \{\, s_i \mid i = 1, \ldots, n \,\} \subseteq (\mathbb{R}^2)^T \qquad (1)$$

of the trajectories for n moving objects $o_1, \ldots, o_n$, where for every $i = 1, \ldots, n$,

- the index i, called the *trajectory ID*, is drawn from a set of n identifiers $ID = \{1, \ldots, n\}$, and
- the i-th trajectory s_i is a sequence

  $$s_i = s_i[1] \cdots s_i[T] \in (\mathbb{R}^2)^T$$

  of T points on the 2-dimensional space $\mathbb{R}^2$ such that its t-th point is $s_i[t] = (x_{it}, y_{it}) \in \mathbb{R}^2$.

Example1 In Fig. 2, we show an example of a trajectory database S, which consists of five trajectories of length $T = 7$.

For example, GPS-trajectories of wild animals, walking people with Wifi device, Probe car data (or floating car data) are instances of such trajectory databases.

2.3 The class of flock patterns

For such trajectory databases, we introduce the class $\mathcal{FP}$ of spatio-tempral patterns, called flock patterns, based on L_∞-norm as follows [*2]. Formally, the class of flock patterns is defined as follows.

Definition1 (FP) *A* flock pattern *on* $\mathbb{T}$ *is a pair* $P = (X, [b, e])$*, where*

- $X \subseteq ID$ *is a finite set of ids, called the* ID set *of* P*, and*
- $I = [b, e]$ *is a discrete interval in* $[0, T]$ *with* $b \leq e \leq T$*, where* b *and* e *are called the* start *and* end time *of* P*.*

We define the support, length, and width of a flock pattern as follows.

- The *support* of P, denoted by *supp*(P), is defined by the number of trajectory (ID) contained in X, that is, $supp(P) = |X|$.
- The *length* of P, denoted by $len(P)$, is the width of the interval I, that is, $len(P) = e - b + 1$.

Clearly, we have $0 \leq supp(P) \leq n$ and $0 \leq len(P) \leq T$.

Example2 In Fig. 2, we show an example of a flock pattern $P_1 = (X_1, I_1)$, where the ID set is $X_1 = \{2, 3, 4\}$ and the interval is $I_1 = [3, 5]$.

[*2] The original version of flock patterns are defined based on L_2-norm in [8], [10].

To define the width, we require some definitions below. For a point $p = (x, y)$ on 2-dimensional plane $\mathbb{R}^2$, the x- and y-coordinates of p are denoted by by $p.x = x$ and $p.y = y$, respectively. For two points p and p' on $\mathbb{R}^2$, we denote the *L_∞-distance between p and p'* by $L_\infty(p, p') = \max\{|p.x - p'.x|, |p.y - p'.y|\}$. By definition, $L_\infty(p, p')$ is nonnegative, and coincides zero if and only if $p = p'$.

The *diameter* of a set $A = \{p_1, \ldots, p_n\}$ of points, denoted by $||\, A\, ||_\infty$, is the maximum L_∞-distance between any two points in A, defined by

$$||\, A\, ||_\infty = \max_{p,p' \in A} L_\infty(p, p'), \qquad (2)$$

The width $||\, A\, ||_\infty$ of a set A is always nonnegative, and equals zero if and only if A consists of a single point. We can show that $||\, A\, ||_\infty$ is linear time computable in $n = |A|$ on $\mathbb{R}^2$. For any $d \geq 2$, $||\, A\, ||_\infty$ can be computed $O(dn)$ time in $\mathbb{R}^d$, which is still linear in n for fixed d.

In an input database S, the t-th time slice, denoted by $S[X][t]$, is the set of all points that appear in the trajectories of X with time stamp t.

- The *width* $||P||_\infty^S$ of a flock pattern $P = (X, I) = (X, [b, e])$ is defined by the maximum diameter of the t-th time slice of the trajectories in X over all $t \in [b, e]$.

Actually, we have the next lemma.

Lemma1 The width of P can be computed by Algorithm 1 in $O(m\ell)$ time, where $m = supp(X)$ is the support of P and $\ell = len(P)$ is the length of P.

Algorithm 1 Computing the width $||\, P\, ||_\infty{}^S$ of a flock pattern $P = (X, [b, e])$ in a database $S = \{\, s_i \mid i = 1, \ldots, n\,\}$

```
1: $width \leftarrow 0$;
2: for $t \leftarrow b, b+1, \ldots, e$ do
3:     $S_t \leftarrow \{\, s_i[t] \mid i \in X\,\}$; // the $t$-th slice
4:     $width \leftarrow \max\{width, ||S_t||_\infty\}$;
5: return $width$;
```

Let $r > 0$ be a positive number, and $k, m \geq 0$ are nonnegative integers, respectively, called a *maximum width* (max-width), a *minimum length* (min-len), and a *minimum support* (min-sup) parameters. Then, we define:

- an *r-flock pattern* is any flock pattern P such that $||\, P\, ||_\infty \leq r$,

Consider the class of r-flock patterns in a trajectory database S.

- An *(r, k)-flock pattern* is any r-flock pattern P with $len(P) \geq k$.

Example3 The pattern P_1 of Fig. 2 in the last example has diameter $||\, P_1\, ||_\infty{}^{S_1} \leq 1.0$, length $len(P_1) = 3$, and support $supp(P_1) = 3$. Thus, it is a $(1.0, 2, 3)$-flock pattern for $r = 1.0$, $k = 2$, and $m = 3$.

In this paper, we consider all (r, k)-flock patterns in a given trajectory database.

2.4 Rightward length-maximal patterns

For a given max-width parameter $r \geq 0$, it is often useful to find only (r, k)-flock patterns $P = (X, [b, e])$ whose time interval $[b, e]$ are extended rightward along time line as long as possible preserving the diameter r (See [8]). This idea of *length-maximal mining* is expected to reduce the number of solutions and running time than just finding all (r, k)-patterns.

A flock pattern $P = (X, [b, e])$ is said to be a *rightward (rightward resp.) length-maximal* flock pattern in S if its interval cannot be extended rightward (leftward or rightward resp.) without changing the width of P in S.

Formally, it is defined as follows.

Definition2 (RFP) A flock pattern $P = (X, [b, e])$ in S is a *rightward length-maximal* flock pattern (RFP, for short) if there is no other flock pattern $P' = (X', [b', e'])$ in S such that (i) $X = X'$, and (ii) $b = b'$ and $e < e'$.

Definition3 (UFP) A flock pattern $P = (X, [b, e])$ in S is a *unrestricted length-maximal* flock pattern (UFP, for short) if there is no other flock pattern $P' = (X', [b', e'])$ in S such that (i) $X = X'$, and (ii) P' has a strictly larger interval than P, that is, $[b, e] \subset [b', e']$.

For a set Γ of constraint parameters drawn from k, m, and r, we denote by $\mathcal{FP}(\Gamma)$, $\mathcal{RFP}(\Gamma)$, and $\mathcal{UFP}(\Gamma)$ the classes of Γ-flock patterns, rightward length-maximal Γ-flock patterns, and unrestricted length-maximal Γ-flock patterns. Then, we have the inclusion $\mathcal{UFP}(r, k) \subseteq \mathcal{RFP}(r, k) \subseteq \mathcal{FP}(r, k)$ for any r and k.

Example4 (RIGHTWARD AND UNRESTRICTED LENGTH-MAXIMAL FLOCK PATTERNS) *Fig. 3 illustrates the notion of rightward and unrestricted length-maximal patterns in database $S = \{s_1, \ldots, s_5\}$ on $\mathbb{T} = [1, 15]$. In the figure, the flock pattern $P_1 = (X, [b_1, e_1])$ starting at $b_1 = 3$ is an RFP, while $P_3 = (X, [b_3, e_3])$ is an UFP On the contrary, the flock pattern P_2 is neither an RFP or an UFP. We also observe that for the UFP $P_3 = (X, [b_3, e_3])$ of length $\ell = 7$, there exist $\ell - 1 = 6$ RFPs with the same end point e_3 but different starts points 9 to 14.* □

From the second observation in the above example, we can say that UFPs and RFps compactly represent a set

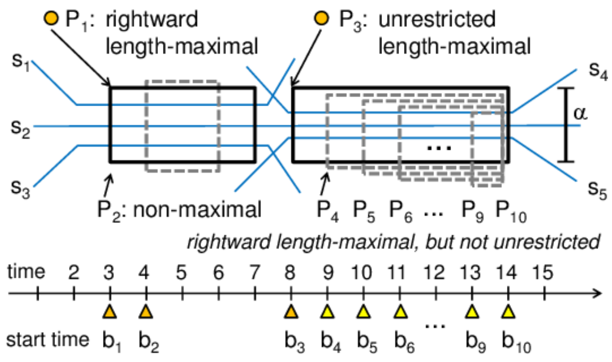
P1: rightward length-maximal
P3: unrestricted length-maximal
s1
s2
s3
s4
s5
α
P2: non-maximal
P4 P5 P6 ... P9 P10
rightward length-maximal, but not unrestricted
time 2 3 4 5 6 7 8 9 10 11 12 13 14 15
start time b1 b2 b3 b4 b5 b6 ... b9 b10

図 3 Concepts of rightward and unrestricted length-maximal flock patterns as well as non length-maximal flock patterns on a time line, where each line indicates a trajectory and each rectangle a flock pattern.

of FPs in S.

Example5 In the example of Fig. 2, the flock pattern $P_1 = (X_1, [3, 5])$ of length three is an RFP in S_1, while $P_2 = (X_1, [3, 4])$ and $P_3 = (X_1, [3])$ are non-rightward length-maximal FPs, where $X_1 = \{2, 3, 4\}$. On the other hand, P_1 has RFPs $P_4 = (X_1, [4, 5])$ and $P_5 = (X_1, [5])$.

2.5 The data mining problems

For any class name $\mathcal{C} \in \{\mathcal{FP}, \mathcal{RFP}, \ldots\}$ and any parameter values $r, k \geq 0$, we denote by $\mathcal{C}(r, k)$ the class of all (r, k)-flock patterns within the class $\mathcal{C}$. Similarly, we define the classes $\mathcal{C}(r)$, and $\mathcal{C}(r, k, m)$ as well. From now on, we consider the classes $\mathcal{FP}(r, k)$, $\mathcal{RFP}(r, k)$, and $\mathcal{UFP}(r, k)$.

We state our data mining problem as follows.

Definition4 (FLOCK PATTERN MINING PROBLEM FOR PATTERN CLASS $\mathcal{C}$) *Let $\mathcal{C}$ be a class of flock patterns. An input is a tuple (S, r, k) of an input trajectory database S, and parameter values r and $k \geq 0$. The task is to find all flock patterns P in S within class $\mathcal{C}$ without repetition that have width at most r and length at least k.*

Similarly, we can consider the flock pattern mining problem with paramters (r, k, m).

We evaluate the performance of a flock pattern mining algorithm $\mathcal{A}$ in terms of enumeration algorithms [3]. Let N and M be the input size and the number of patterns as solutions. A pattern mining algorithm $\mathcal{A}$ is said to have *polynomial delay* (poly-delay) if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial $p(N)$ in N. $\mathcal{A}$ is of *polynomial space* (poly-space) if the maximum size of its working space, in addition to that of output stream O, is bounded by a polynomial $p(N)$.

We give the model of computation in this paper as follows. Let N and M be the size of input S and the number of outputs in $\mathcal{F}$ on S, and $p(N)$ be a polynomial. In our problem, the input size is the total size $N = nT$ of an input trajectory database S. An enumeration algorithm $\mathcal{A}$ is of *polynomial enumeration time* (poly-enum) if the *amortized time* for each solution $x \in \mathcal{S}$ is bounded by a polynomial $p(N)$ in N. $\mathcal{A}$ is of *polynomial delay* (poly-delay) or *exact polynomial enumeration time* if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial $p(N)$ in N. $\mathcal{A}$ is of *polynomial space* (poly-space) if the maximum size of its working space, without the size of output stream O, is bounded by a polynomial $p(N)$.

Algorithm 2 An algorithm RFPM for finding all length-maximal (r, k)-flock patterns appearing in a given trajectory database S with ID for maximum width r and minimum length k.

```
 1: procedure RFPM($ID, S, r, k$)
 2:     for $b_0 \leftarrow 1, \ldots, T$ do  // Each start time in $\mathbb{T}$
 3:         for $i_0 \leftarrow 1, \ldots, n$ do  // Each id in $ID$
 4:             $P_0 = (\{i_0\}, [b_0, *])$;
 5:             RecRFPM($P_0, ID, S, r, k$);

 6: procedure RecRFPM($P = (X, [b, *]), ID, S, r, k$)
 7:     $P = (X, [b, e]) \leftarrow$ RH_Closure($(X, [b, *]); S, r$);
 8:     if $len(P) < k$ then
 9:         return ;  // $P$ is not an $(r, k)$-flock pattern
10:     output $P$;
11:     $ID_1 \leftarrow ID$;
12:     while $ID_1 \neq \emptyset$ do
13:         $i = deletemin(ID_1)$;
14:         $P_1 = (X \cup \{i\}, [b, *])$;
15:         RecRFPM($P_1, ID_1, S, r, k$);
16:     end while
```

3. Algorithms

In this section, we present our pattern mining algorithms for FPs and RFPs. We also give a speed-up technique using geometric indexes to prune redundant candidates.

3.1 A polynomial delay and space algorithm for RFPs

In Algorithm 2, we present the proposed algorithm RFPM (rightward flock pattern miner) for RFPs (rightward length-maximal flock patterns), the class of rightward length-maximal flock patterns, with its subprocedure RecRFPM.

3.1.1 Outline of the algorithm

We describe the computation done by the algorithm RFPM. The overall structure of RFPM is almost identical to the basic algorithm FPM. Given a database S, the main algorithm RFPM invokes the recusive subprocedure RecFPM with an initial pattern P_0 as before.

Only the difference in the top level is that RFPM iterates only $O(T)$ iteration here for the start position b_0 rather than $O(T^2)$ iteration in FPM using an initial pattern $P_0 = (\{i_0\}, b_0, *)$ with missing end position $e_0 = *$, called a *partial pattern* here.

The computation of the recursive subprocedure RecRFPM proceeds in the following steps.

- Step 1: Receiving a partial RFP $P_* = (X, b, *)$ as arguments, the recursive procedure RecFPM computes the rightward horizontal closure $P = (X, [b, e])$ from P_* by the procedure RH_Closure with max-width r.
- Step 2: Next, if the obtained RFP P satisfies (r, k)-constraints, then output it. Otherwise, we sefely prune all descendants as before.
- Step 3: Finally, RecFPM recursively calls its copy with an extended pattern $P_1 = (X \cup \{i\}, [b, *])$. To avoid duplicated generation of patterns, the id i is removed from the universe ID.

The pruning of all descendants in Step 2 above is justified by the following lemma, which says the class of (r, k)-patterns has a kind of monotonicity w.r.t. set inclusion of their ID sets.

Lemma2 (monotonicity) Let $P_i = (X_i, I_i)$ are two flock patterns, where $i = 1, 2$. If P_2 is an (r, k)-flock pattern in S and if $X_1 \subseteq X_2$ and $I_1 \subseteq I_2$ hold, then P_1 is also an (r, k)-flock pattern in S.

From this lemma, once a candidate pattern $P = (X, I)$ does not satisfy the width and length constrants, any descendant of P obtained by adding new trajectory (ids) to X no longer satisfies the constraints. Therefore, we can prune the whole search sub-space for descendants of P for (r, k)-flock patterns.

On the running time and space of the algorithm FPM, we have the following lemma.

3.1.2 Rightward horizontal closure

From the view of frequent pattern mining, RFPs in a trajectory database are a sort of *closed patterns*, which have been extensively studied in frequent itemset mining (FIM) field [13], [17] as well as formal concept analysis (FCA) field. Many efficient closed pattern mining algorithms use a class of operation, called *closure* operation, which enlarge a given, possibly non-closed pattern to obtain its *closed* version.

For RFPs, we actually have a *rightward horizontal closure* operation that extends the interval of a given non RFPs to obtain a proper RFP.

Algorithm 3 An algorithm for computing the unique rightward length-maximal flock pattern. Note that $\|\, S[X][t] \,\|_\infty$ is defined to be ∞ for $t \notin [1, T]$.

```
1: procedure RH_Closure($(X, [b_0, e_0]); S, r$)
2:     $t \leftarrow b_0$;
3:     while $\|\, S[X][t] \,\|_\infty \le r$ do
4:         $t \leftarrow t + 1$;
5:     $b \leftarrow b_0$; $e \leftarrow t - 1$;
6:     return $(X, [b, e])$;
```

Definition5 (rightward horizontal closure) Let $P = (X, I = [b, e])$ be any flock pattern in a database S. Then, the *rightward horizontal closure* of P in S, denoted by RH_Closure$(P; S, r)$, is the unique flock pattern $P_{max} = (X, I = [b, e_{max}])$ such that $e_{max} \in [0, T]$ is the maximum value of end position e' satisfying the equality

$$\|\, P' = (X, [b, e']) \,\|_\infty = \|\, P \,\|_\infty. \qquad (3)$$

Note that the rightward horizontal closure operation only change the end position e, but not change the ID set X or starting time b of the original P at all.

In Algorithm 3, we show the procedure RH_Closure that computes the rightward horizontal closure of non-RFP P in $O(k\ell)$ time, where $k = supp(P) = |X| = O(n)$ and $\ell = len(P_{max}) = O(T)$.

The following lemmas show the correctness of the rightward horizontal closure. First, the key of the correctness is the following characterization, which can be easily shown from definition of RFPs.

Lemma3 (characterization) Let $P = (X, [b, e])$ be an (r, k)-flock pattern in S. Then, P is rightward length-maximal if and only if

- $\|\, S[X][t] \,\|_\infty \le r$ for all $t \in [b, e]$, and
- $\|\, S[X][e + 1] \,\|_\infty > r$,

where we extend the t-th time slice $\|\, S[X][t] \,\|_\infty$ to be ∞ if either $t < 1$ or $t > T$ holds for convenience.

From the above lemma, we have the correctness below.

Lemma4 The rightward horizontal closure P_{max} of a possibly non-rightward length-maximal r-FP P is the unique longest r-RFP such that the ID sets and the start time are identical to those of P.

Since $len(P_{max}) \ge len(P)$ always holds for P_{max}, we see that if P satisfies the (r, k)-constraint then so does the obtained RFP P_{max}. Hence, P_{max} is the unique longest (r, k)-RFP version of P that share the ID set and start time.

3.1.3 Analysis

From a similar argument to [13] based on reverse search technique of [3], we have the following time and space complexities of RFPM.

Theorem1 Let S be an input trajectory database S of n trajectories with length T. Then, the algorithm RFPM in Algorithm 2 solves the flock pattern mining problem for the class $\mathcal{RFPM}(r, k)$ of (r, k)-flock patterns in S. It uses $O(mnT)$ time per pattern and $O(m^2)$ words of space, respectively, where $m = supp(X) = |X|$ is the support of the pattern X being enumerated.

We can generalize RFPM for the case of the d-dimensional space $\mathbb{R}^d$ for every $d \geq 1$ with extra $O(d)$ factor in time and space by only modifying the procedure RH_Closure for $\mathbb{R}^d$.

3.2 An algorithm for UFPs

In this subsection, we show how to extend the algorithm BasicFPM in the previous subsection to solve the flock pattern mining problem for the class $\mathcal{UFP}(r, k)(S)$ in poly-enum and poly-space.

From Lemma 3, we already know that any (r, k)-UFP is also a proper (r, k)-RFP. Conversely, Lemma 5 below shows that exactly when a given (r, k)-RFP is a proper (r, k)-UFP.

Lemma5 (FILTERING LEMMA) A rightward length-maximal (r, k)-flock pattern $P = (X, [b, e])$ in S is also unrestricted length-maximal in S if and only if $||\, S[X][b-1] \,||_\infty > r$.

Proof: The proof from the definition of UFPs. ■

We refer to the condition in the above lemma as the *leftward extension test*. Let us denote by UnrestRecFPM the version of recursive procedure RecFPM in Algorithm 2 that is modified by replacing Line 10

```
9: output $P$;
```

with the following code for leftward extension test:

```
9: if ($||\, S[X][b-1] \,||_\infty \leq r$) then output $P$;
```

From Lemma 5, we can show the correctness of the modified procedure UnrestRecFPM for all (r, k)-UFPs belonging to $\mathcal{UFP}(r, k)(S)$ since UnrestRecFPM finds all (r, k)-RFPs as candidate, tests the discovered RFPs, and discards all non (r, k)-RFPs by the leftward extension test.

The remaining task is to show that the modified algorithm has the poly-enum and poly-space complexities. To see this, we estimate an upperbound of the number of RFPs in terms of that of UFPs. Let us denote by $\#\mathcal{RFP}(r, k)(S)$ and $\#\mathcal{UFP}(r, k)(S)$ the numbers of all (r, k)-RFPs and all (r, k)-UFPs in a given database S.

In Fig. 3, we observe that an (r, k)-UFP has a set of *equivalent* (r, k)-RFPs with the same ID set X and end time e. Generalizing this observation, we have the following theorem, where $T = |\mathbb{T}|$.

Lemma6 For any database S, we have the inequality

$$\#\mathcal{RFP}(r, k)(S) \leq T \cdot \#\mathcal{UFP}(r, k)(S). \tag{4}$$

Proof: From the proof of Lemma 4, every unrestricted length-maximal (r, k)-flock pattern P in $\mathcal{UFP}(r, k)(S)$ that has length ℓ can have at most ℓ rightward length-maximal (r, k)-flock patterns with the same ID set X and the same end point e, including P itself. Therefore, we have the next lemma. ■

The above lemma says that $\#\mathcal{UFP}(r, k)(S)$ is not much larger than $\#\mathcal{RFP}(r, k)(S)$. Therefore, we have the following theorem for UFPs.

Theorem2 (POLY-DELAY AND POLY-SPACE MINING FOR UFP) Let S be a trajectory database, $r > 0$ a max-width, and k a min-length. Then, there exists some algorithm that finds all unrestricted length-maximal (r, k)-flock patterns in S in $O(mnT^2)$ time per pattern without duplicates using $O(m^2)$ words of space, where $m = |X|$ is the size of ID set being enumerated, $n = |ID|$, and $T = |\mathbb{T}|$.

Proof: From Lemma 6, for finding each (r, k)-UFP as solution, we test at most T (r, k)-RFP as candidate by using UnrestRecFPM as subprocedure that requires $O(knT)$ time per RFP. This completes the proof. ■

4. Conclusion

This paper study the problem of complete mining of flock patterns from a large trajectory database. For the classes of rightward and unrestricted length-maxmal flock patterns, We presented a poly-delay and poly-space depth-first mining algorithm.

Our mining algorithm can work with higher dimension $d \geq 2$ due to L_∞-distance. An open question is if it is still true with other metric such as L_k-distance for any $k = 1, 2, \ldots$. There are known difference between L_∞ and L_2. For instance, linear time computation is easy for minimum bounding rectangles, while it seems rather complicated for minimum bounding circles [6]. Also, rectangles are closed under intersection, but the circles not.

From the view of *closed pattern mining* [1], it will be an interesting question if fast closed itemset mining technique, e.g., LCM [13], can be applied to closed flock patterns. Other possiblity is to study the geometric counterpart of the classes of *flexible* patterns, such as closed sequence patterns or closed sequential episodes such as

[2], [15]. Thus, extension of this framework to the flexble pattern mining will be interesting as in [7].

Massive trajectory data wil be collected on cloud platforms in future. From this view, it is interesting to study how to efficiently store, search, and mine flock patterns on trajectory data on such environment.

Acknowledgment

The authors would like to thank Kensuke Onishi, Yoshio Okamoto, Yushi Uno, Yusaku Kaneta, Koji Tsuda, Key Sekine, Satoshi Yoshida, Shuhei Denzumi, Shinichi Minato, Takuya Kida, Thomas Zeugmann, and Yuzuru Tanaka, for fruitful discussion on trajectory data mining.

参考文献

[1] H. Arimura and T. Uno. An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. *Journal of Combinatorial Optimization*, 13:243–262, 2006.

[2] H. Arimura and T. Uno. Mining maximal flexible patterns in a sequence. In *New Frontiers in Artificial Intelligence*, LNCS 4914, pages 307–317, 2007.

[3] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Math.*, 65:21–46, 1993.

[4] P. Bakalov, M. Hadjieleftheriou, E. Keogh, and V. J. Tsotras. Efficient trajectory joins using symbolic representations. In *Proc. MDM'05*, pages 86–93. ACM, 2005.

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

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[7] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proc. KDD'07*, pages 330–339. ACM, 2007.

[8] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. ACM GIS '06*, pages 35–42. ACM, 2006.

[9] J. Gudmundsson, M. van Kreveld, and B. Speck. Efficient detection of motion patterns in spatio- temporal sets. *GeoInformatica*, 11:195–215, 2007.

[10] P. Laube, M. van Kreveld, and S. Imfeld. Finding REMO — detecting relative motion patterns in geospatial lifelines. In *Spatial Data Handling*, pages 201–215. Springer, 2005.

[11] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[12] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE TKDE*, 16(11):1424–1440, 2004.

[13] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Proc. the 7th Discovery Science (DS'04)*, volume 3245 of *LNCS*, pages 16–31. Springer, 2004.

[14] M. R. Vieira, P. Bakalov, and V. J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. GIS'09*, pages 286–295. ACM, 2009.

[15] J. Wang and J. Han. Bide: efficient mining of frequent closed sequences. In *Proc. ICDE'04*, pages 79–90. IEEE, 2004.

[16] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May/Jun 2000.

[17] M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):462–478, 2005.