Efficient Mining of Length-Maximal Flock Patterns from Large Trajectory Data

Xiaoliang Geng
Hokkaido University
N14 W9, Kita-ku
Sapporo 060-0814, Japan
gengxiaoliang@ist.hokudai.ac.jp

Takeaki Uno
National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku
Tokyo 101-8430, Japan
uno@nii.jp

Hiroki Arimura
Hokkaido University
N14 W9, Kita-ku
Sapporo 060-0814, Japan
arim@ist.hokudai.ac.jp

ABSTRACT

In this paper, we study the problem of mining a class of spatio-temporal patterns, called flock patterns, which represent a groups of moving objects close each other in a given time segment (Gudmundsson and van Kreveld, Proc. ACM GIS'06; Benkert, Gudmundsson, Hubner, Wolle, Computational Geometry, 41:11, 2008). Based on frequent-pattern mining approach, such as Apriori, Eclat, or LCM, we present an efficient depth-first mining algorithm that finds all maximally longest flock patterns appearing in a collection of trajectory data in polynomial time per pattern using polynomial space, without using table-lookup. We also present a geometic pruning technique which significantly improves the efficiency of the algorithm very much.

Keywords

spatio-temporal sequence mining, time series, graph mining, frequent and closed pattern mining, foundations

1. INTRODUCTION

1.1 Background

By the rapid progress of mobile devices and positional sensors, a massive amount of trajectory data have been accumulated. Since trajectories are sequences of real-valued locations with errors and missing values, mining of trajectory data have different characteristics from traditional transaction data mining [12, 14, 17]. Hence, research of trajectory mining has attracted a great deal of attention for recent years [7, 4].

A *trajectory database* on the time domain $\mathbb{T} = \{1, \ldots, T\}$ is a set $S = \{ s_i \mid i = 1, \ldots, n \}$ of n 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 $\mathbb{R}^2$, where its index i, called the ID, is drawn from a set of n identifiers $ID = \{1, \ldots, n\}$. GPS-trajectories of wild animals, walking people with Wifi device, Probe car data (or floating car data) are examples of such trajectory databases.

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*. For a positive number $m > 0$, called a max-width, and non-negative integers $k, r \geq 0$, called min-len and min-sup, an (m, k, r)-*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 r moving objects that move together with mutually distance at most m in L_∞-norm, that is, the largest of the x- and y-distances, along a continuous interval of at least k time points length. Flock patterns are useful in detecting a group of highly correlated entities combining spatio-tempral features.

There have been a number of existing researches on finding flock patterns in a given trajectory database [5, 8, 10]. Unfortunately, however, most of these works deals with detection of one or more flock patterns as *pattern search* problem, not as *pattern mining* problem except few work (e.g., Vieira *et al.* [15]).

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, 14, 17]. Particularly, we study the problem of finding all (m, k)-flock patterns [1] ((m, 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

A common problem to pattern mining approaches is the huge number of generated patterns that degrades the performance and comprehensiveness of pattern mining. To cope with this problem, the framework of *closed pattern mining* has been extensively studied in itemsets and graph mining [1, 14, 16, 17]. Along the above line of research on closed patterns, by extending (m, k)-FPs above, we first introduce the classes of RFPs and UFPs of closed flock patterns as follows.

Given a maximum width parameter m, it is often useful to find all (m, k)-flock patterns $P = (X, [b, e])$ whose time interval $[b, e]$ are extended leftward or rightward as long as

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

possible along time line with preserving the diameter m 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* (m, k)-flock patterns, denoted by (m, k)-RFPs and (m, 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. As expected, it is shown that a RFP is a compact representation of FPs, and moreover, an UFP is a more compact representation of RFPs,

Unlike Gudmundsson and van Kreveld's longest-duration (m, k, r)-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*. This is analogous to the situation of maximal cliques in a graph: *maximal* cliques can be efficiently enumerated per solution, while a *maximum* clique is hard to find.

1.2.2 Polynomial delay and space algorithms

First of all as a main result, we present a depth-first search mining algorithm BasicFPM (Algorithm 3) that finds all rightward length-maximal (m, k)-flock patterns P, or (m, k)-RFPs, in a given trajectory database S of n transactions of length T in $O(knT)$ time per pattern without duplicates using $O(k^2)$ words of space, where $k = |X|$ is the number of trajectories that the discovered P contains. Actually, BasicFPM is a polynomial-delay and polynomial space algorithm for (m, 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 (m, 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 (m, k)-UFPs in polynomial-delay and polynomial space, too (Theorem 3).

Thirdly, we discuss the speed-up of our depth-first algorithm based on a geometric pruning. We present our modified mining algorithm GridFPM (grid-based flock pattern miner) based on geometric constraint on the 2-dimensional plane for (m, k)-flock patterns.

1.2.3 Experimental results

Finally, we ran experiments on synthesis datasets to evaluate the efficiency and the usefulness of the proposed algorithms BasicFPM and GridFPM for RFPs and extensions for UFPs. As results, both algorithms BasicFPM and GridFPM efficiently finds RFPs and UFPs. Paricularly, the geometric pruning technique introduced in Sec. 3.3 improves the performance of GridFPM more than ten times than that of BasicFPM. As actual timing, GridFPM finds all RFPs in a trajectory database with totally 1/4 million points in less than 10 seconds on a PC. Hence, it seems useful in analysis of realworld trajectory data.

1.2.4 Contributions of this paper

The contributions of this paper is summarized as follows.

- We first introduce the classes RFP and UFP of rightward and unrestricted length-*maximal* (not maximum) flock patterns, and study the problem of enumerating all patterns in the classes (Def. 2).
- We present a time and memory efficient depth-first mining algorithm that finds the complete sets of all rightward length-maximal flock patterns in RFP without duplicates, for the first time to give a polynomial-delay and space enumeration algorithm for the class (Theorem 1). As advantage to the previous work, our algorithm works in higher dimension.
- Furthermore, we give a characterization of the class UFP of unrestricted length-maximal flock patterns, and using this property, present a polynomial delay and space algorithm for the class (Theorem. 3).
- We give a new geometric pruning technique tailored to trajectory pattern mining, which significantly improves the performance more than ten times using spatial indexes (Fig. 3 and Fig. 4).

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 [10, 9, 5]. Gudmundsson and van Kreveld [8] showed that the problem of finding at least one length-*maximum* (m, k, r)-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 r 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 [15], who took pattern mining aproach at the first time. They presented an efficient algorithm BFE with variations that finds all (m, k, r)-flock patterns by systematically combining discovered clusters by depth-first search using the idea of intersection. This seems to be one of the first paper that take *pattern mining* approach for flock patterns. They also introduced the use of geometric constraints and spatial index for mining. However, it is not polynomial space algorithm since it uses tabulation to avoid duplicated patterns.

Recently, Romero [13] presented in his Master's thesis an interesting approach of solving a flock pattern mining by reduction to frequent pattern mining, where he used LCM algorithm [14] to solve the converted problem. However, its scalability seems to need improvement.

1.4 Organization

Sec.2 gives definitions, Sec.3 presents our algorithms, and Sec.4 shows experimental results. Finally, Sec.5 concludes.

2. PRELIMINARIES

Let $\mathbb{R}$ and $\mathbb{N}$ be the set of all real numbers and all non-negative integers, respectively. Let $\mathbb{T} = \{1, \ldots, T\}$, $T \geq 0$ be the set of intergers or the *time domain*, and let $\mathbb{R}^2$ be the *2-dimensional continuous space* or the *plane*. Each element t of $\mathbb{T}$ is called *time point*, and each element $p = (x, y)$ of $\mathbb{R}^2$ a *2-dim point* or *point*, where we denote the x- and y-coordinates by $p.x = x$ and $p.y = y$. A *time interval* on $\mathbb{T}$ is an interval $[b, e] = \{b, b+1, \ldots, e\}$, where $1 \leq b \leq e \leq T$ are time points. 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.

For points $p, p' \in \mathbb{R}^2$, we denote the L_∞-distance between p and p' by $L_\infty(p, p') = \max\{|p.x - p'.x|, |p.y - p'.y|\} \geq 0$. For a set of points $A \subseteq \mathbb{R}^2$ in the plane, the L_∞-*width* of A (or simply the *width* of A) is defined by

$$|| A ||_\infty = \max_{p,p' \in A} L_\infty(p, p') \geq 0, \tag{1}$$

that is, the maximum L_∞-distance between any two points in A. $|| A ||_\infty$ can be computed in linear time in the number of points in A, and in $O(d|A|)$ time in d-dimensional space $\mathbb{R}^d$ for any $d \geq 2$.

As note, we mainly consider the class of flock patterns based on geometry of L_∞-distance in this work, while the original paper by Benkert *et al.* [5] on flock patterns considered the class based on L_2-distance. Generally, however, we can define those based on any L_k-distance for $k \in \{1, 2, \ldots, \infty\}$.

2.1 Trajectories

A *trajectory* of a moving object o, called an *entity*, on the plane with time domain $\mathbb{T}$ is a sequence of T points on $\mathbb{R}^2$

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

where for every $t = 1, \ldots, T$, the t-th point $s[t] = (x_t, y_t) \in \mathbb{R}^2$ designates the location of o at time t. In this paper, we assume that all trajectories have the same length T. The *projection* of s to time interval $I = [b, e] \subseteq \mathbb{T}$ is the sub-trajectory $s[b, e] = s[b] \cdots s[e]$ consisting of the points of S in time interval I.

Let $ID = \{1, \ldots, n\}$ be the set of *trajectory identifiers* (or *ids*) of n moving objects $o_1, \ldots, o_n$ on the plane. A *trajectory database* on $\mathbb{T}$ with ID set ID is a set of n trajectories

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

where for every index $i = 1, \ldots, n$, the i-th trajectory is $s_i = s_i[1] \cdots s_i[T]$, and the set of trajectory IDs is denoted by $ID(S) = \{1, \ldots, n\}$. The *input size* of S is the total size $N = nT$ of its trajectories. In what follows, we fix the set ID and an associated trajectory database S.

2.2 Flock patterns

An *ID set* in S is any subset $X = \{i_1, \ldots, i_k\} \subseteq ID(S)$, $k \geq 0$, which designates the set $S[X]$ of trajectories with IDs in X defined as $S[X] = \{\, s_i \mid i \in X \,\}$. We do not distinguish X from its trajectories if it is clear from context.

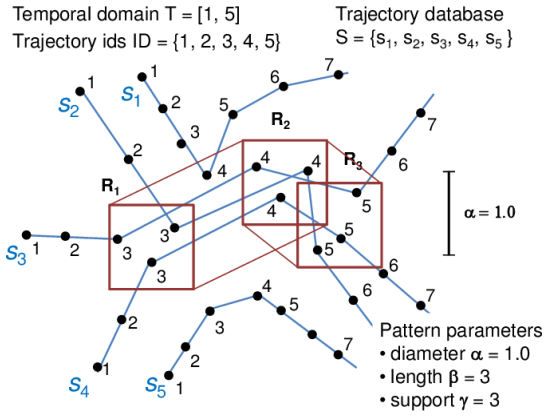

Figure 1: Examples of a trajectory database S_1 consisting of five trajectories $\mathsf{s}_1, \ldots, \mathsf{s}_5$ with ID set $ID = \{1, \ldots, 5\}$, and a flock pattern $P_1 = (X = \{2, 3, 4\}, [t_3, t_5])$ with diameter $m = 1.0$, length $k = 3$, and support $r = 3$, where each line indicates a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

DEFINITION 1. A *flock pattern* on $\mathbb{T}$ is a pair

$$P = (X, [b, e]) \subseteq 2^{ID} \times 2^{\mathbb{R}^2},$$

where $X \subseteq ID$ is called the *ID set*, and $I = [b, e]$ is a *time interval* on $\mathbb{T}$ such that $1 \leq b \leq e \leq T$. The time points b and e are called the *start* and *end (time) point*. We define the *support* and *length* of P by $supp(P) = |X| \leq n$ and $len(P) = e - b + 1 \leq T$, respectively.

It is often convenient to leave the end point e unspecified. A pattern P is *partial* if its end point is $e = *$ indicating ∞ or T, that is, $P = (X, [b, *])$. As semantics, the *denotation* of a flock pattern $P = (X, [b, e])$ with $\ell = e - b + 1$ in S is the set of all sub-trajectories of length ℓ

$$D_S(P) = \{\, s_i[b, e] \mid i \in X \,\} \subseteq 2^{(\mathbb{R}^2)^*},$$

which consists of the projections of all trajectories of S to the time interval $[b, e]$ whose IDs belong to X.

For any $t \in [b, e]$, let us denote by $S[X][t] \subseteq \mathbb{R}^2$ the set of all points at the specified time t, called the *time slice of X at time t*. That is, $S[X][t] = \{\, s_i[t] \in \mathbb{R}^2 \mid i \in X \,\}$. Then, the *width* of P, denoted by $|| P ||_\infty$, in S is defined by the maximum width of the t-th time slice $S[X][t]$ of $D_S[P]$ over all $t \in [b, e]$. That is,

$$\begin{aligned} || P ||_\infty &= \max_{t \in [b,e]} || D_S[P][t] ||_\infty \\ &= \max_{t \in [b,e]} || S[X][t] ||_\infty \\ &= \max_{t \in [b,e]} \max_{p,p' \in S[X][t]} L_\infty(p, p'). \end{aligned} \tag{2}$$

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

- an *m-flock pattern* is any flock pattern P such that $||\,P\,||_\infty \leq m$,
- an (m,k)*-flock pattern* is any m-flock pattern P with $len(P) \geq k$, and
- an (m,k,r)*-flock pattern* is any (m,k)-flock pattern P with $supp(P) \geq r$.

LEMMA 1 (ANTI-MONOTONICITY OF (m,k)-PATTERNS). *Let $P_i = (X_i, I_i)$ are two flock patterns, where $i = 1, 2$. If P_2 is an (m,k)-flock pattern in S and if $X_1 \subseteq X_2$ and $I_1 \subseteq I_2$, then P_1 is also an (m,k)-flock pattern in S.*

Similar lemmas hold for the class of m-flock patterns, but not for the class of (m,k,r)-flock patterns. From Lemma 1 above, we can effectively prune descendants in pattern search for m- and (m,k)-flock patterns similarly to anti-monotone pruning in frequent itemset mining (e.g. [17]).

2.3 Length-maximal flock patterns

For a given maximum width parameter m, it is often useful to find all (m,k)-flock patterns $P = (X, [b,e])$ whose time interval $[b,e]$ are extended leftward or rightward along time line as long as possible preserving the diameter m (See [8]), in the sense of L_∞-distance here. This idea of *length-maximal mining* will reduce the number of solutions and running time than just finding (m,k)-patterns of legnth exactly L for all the possible values of $L \geq k$.

DEFINITION 2. Let $m > 0$ and $k \geq 0$ are max-width and min-len parameters. Let $P = (X, [b,e])$ be an (m,k)-flock pattern in S.

- P is said to be *rightward length-maximal* if there is no other (m,k)-flock pattern $P' = (X, [b,e'])$ such that $e < e'$.
- P is said to be *(unrestricted) length-maximal* if there is no other (m,k)-flock pattern $P' = (X, [b',e'])$ such that $[b,e] \subset [b',e']$, i.e., $[b,e]$ is properly included in $[b',e']$,

A rightward (or unrestricted resp.) length-maximal (m,k)-flock pattern is abbreviated as an (m,k)-*RFP* (or an (m,k)-*UFP* resp.).

EXAMPLE 1. (RIGHTWARD AND UNRESTRICTED LENGTH-MAXIMAL FLOCK PATTERNS) Fig. 2 illustrates the notion of rightward and unrestricted length-maximal patterns with diameter m in a trajectory database S containing five trajectories s_1, s_2, s_3, s_4, and s_5 on time line $\mathbb{T} = [1, 15]$, where $b_1, \ldots, b_{10} \in \mathbb{T}$ indicate the start times of flock patterns $P_1, \ldots, P_{10}$ with the same ID set $X = \{1, 2, 3, 4, 5\}$, respectively.

In the figure, we first observe that the flock pattern $P_1 = (X, [b_1, e_1])$ starting at $b_1 = 3$ is rightward length-maximal since the end point $e_1 = 7$ cannot be extended rightward

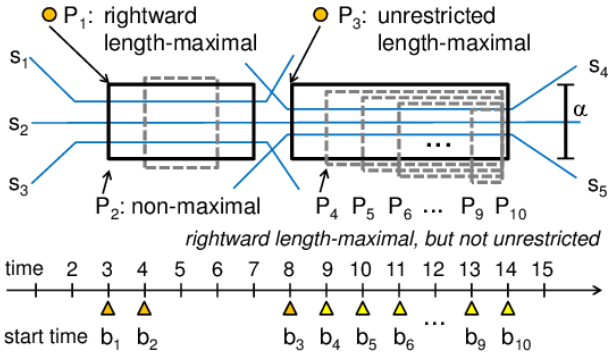

Figure 2: Concepts of rightward and unrestricted length-maximal flock patterns as well as non length-maximal flock patterns on a time line, where each line indicates a trajectory and each rectangle a flock pattern.

any more, while the $P_3 = (X, [b_3, e_3])$ starting at $b_3 = 8$ is unrestricted length-maximal since neither of the start point $b_3 = 8$ or the end point $e_3 = 14$ can be extended further rightward or leftward.

On the contrary, the flock pattern P_2 at $b_2 = 4$ is neither rightward or unrestricted length-maximal since the begin point $b_2 = 4$ can be exntended leftward to $b_2 = 2$ and the end point $e_2 = 6$ can be rightward to $e_2 = 7$, respectively.

We also observe that for the unrestricted length-maximal pattern $P_3 = (X, [b_3, e_3])$ with start point $b_3 = 8$ and length $\ell = e_3 - b_3 + 1 = 7$, there exist $\ell - 1 = 6$ rightward length-maximal patterns sharing the common end point 14 with different starts point from 9 to 14. □

From the second observation in the above example, we can say that an unrestricted length-maximal flock pattern compactly represents a set of flock patterns in S than rightward length-maximal ones do, and a rightward length-maximal flock pattern than ordinary ones. We will come back this issue more formally in Sec. 3.2.

Let $r \subseteq \{m, k, r\}$ be a set of constraints, and S a given database. Then, we denote by $\mathcal{UFP}(r)(S)$ the set of all unrestricted length-maxmal (m,k)-flock patterns appearing in S. Similarly, we define $\mathcal{RFP}(r)(S)$ and $\mathcal{FP}(r)(S)$ as notations as the corresponding subclasses of rightward length-maxmal (m,k)-flock patterns and ordinary (m,k)-flock patterns in S.

In this paper, we mainly consider the mining problem for the classes $\mathcal{RFP}(m,k)$ and $\mathcal{UFP}(m,k)$ of rightward and unrestricted (m,k)-flock patterns. The next lemma follows from the definition. As seen below, we have the inclusion:

$$\mathcal{UFP}(r) \subseteq \mathcal{RFP}(r) \subseteq \mathcal{FP}(r). \tag{3}$$

NOTE 1. *For the class of (m,k,r)-flock patterns, the problem of finding at least one* length-maximum *patterns within the class is shown to be NP-hard [8]. However, we note that this does not exclude the possibility of finding* length-maximal*, not* maximum*, flock patterns in polynomial time*

Algorithm 1 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 RightExt(X, b_0, m, S)
2:     t ← b_0;
3:     while || S[X][t] ||_∞ ≤ m do
4:         t ← t + 1;
5:     b ← b_0; e ← t − 1;
6:     return (X, [b, e]);
```

per pattern by enumeration. This is analogous to the case of maximum and maximal cliques, where the former is NP-hard, but the latter is efficiently solvable.

Then, we have the following characterizations.

LEMMA 2 (CHARACTERIZATION). *Let $P = (X, [b, e])$ be an (m, k)-flock pattern in S. Then, we have the followings:*

1. *P is rightward length-maximal if and only if*
   (a) $||\, S[X][t] \,||_\infty \leq m$ *for all* $t \in [b, e]$, *and*
   (b) $||\, S[X][e+1] \,||_\infty > m$.
2. *P is length-maximal if and only if*
   (a) $||\, S[X][t] \,||_\infty \leq m$ *for all* $t \in [b, e]$,
   (b) $||\, S[X][b-1] \,||_\infty > m$, *and* $||\, S[X][e+1] \,||_\infty > m$.

In the above definition, we defined $||\, S[X][t] \,||_\infty$ to be ∞ if either $t < 1$ or $t > T$ holds for convenience.

The next lemma tells us how we can compute the unique rightward and unrestricted length-maximal version of a given flock pattern as the result of a sort of *closure* operation as in closed itemset mining [14, 17]. This property is a key to our mining algorithm.

LEMMA 3 (UNIQUE RFP VERSION). *Let P be an (m, k)-flock pattern with ID set X, (1) Then, there exists the unique rightward length-maximal version of P with the same ID set X denoted by $P' = RightExt(P)$. (2) Furthermore, P' is computable in $O(k\ell) = O(kT)$ time from P, where $k = |X|$ and $\ell = len(\hat{P})$.*

PROOF. (1) Let $P = (X, [b, e])$ be any (m, k)-flock pattern. The procedure RightExt in **Algorithm** 1 correctly computes $RightExt(P)$ from X and b. (2) The time complexity is obvious from the algorithm. □

Extension to $\mathbb{R}^d$ just add $O(d)$ factor for any $d \geq 2$. The next closure property is important in the discussion on unrestricted length-maxmal flock patterns.

LEMMA 4 (UNION OF INTERVALS). *If two (m, k)-flock patterns $P_i = (X, [b_i, e_i])$, where $i = 1, 2$, have overlapping intervals $[b_1, e_1] \cap [b_2, e_2] \neq \emptyset$, then $P_{12} = (X, [b_1, e_1] \cup [b_2, e_2])$ is also a proper (m, k)-flock pattern in S.*

The proof of the above lemma is straightforward, and thus omitted. From Lemma 4 above, the next lemma immediately follows.

LEMMA 5 (UNIQUE UFP VERSION). *For any (m, k)-flock pattern $P = (X, [b, e])$, (1) there exists the unique unrestricted length-maximal version of $P' = (X', [b', e'])$, denoted by $P' = Ext(P)$, such that $X = X'$ and $[b, e] \subseteq [b', e']$ (2) Furthermore, P' is computable in $O(k\ell) = O(kT)$ time from P, where $k = |X|$ and $\ell = len(\hat{P})$.*

For the proof of the time complexity for computing $P' = Ext(P)$ above, we use a similar procedure to **Algorithm** 1 by extending the time interval of P in both sides while its width does not exceeds m. Detail is omitted.

2.4 Data mining problems

Let $\mathcal{F} \in \{\mathcal{FP}, \mathcal{RFP}, \mathcal{UFP}\}$ be any subclasses of rightward or unrestricted length-maxmal flock patterns, and $r \subseteq \{m, k, r\}$ be a set of constraints. For a given database S, we denote by $\mathcal{RFP}(r)(S)$ the set of all rightward length-maxmal (m, k)-flock patterns appearing in S. Similarly, we define the sets $\mathcal{FP}(r)(S)$ and $\mathcal{UFP}(r)(S)$.

We state our data mining problem as follows.

DEFINITION 3. (FLOCK PATTERN MINING PROBLEM FOR PATTERN CLASS $\mathcal{F}$) Let $\mathcal{F}$ be a class of (length-maximal) flock patterns and $r \subseteq \{m, k, r\}$. Given a trajectory database S, maximum width $m > 0$ and minimum length k, find all flock patterns $P \in \mathcal{F}(r)(S)$ appearing in S without duplicates.

Specifically, we focus on the flock pattern mining problem for the classes $\mathcal{RFP}(m, k)$ and $\mathcal{UFP}(m, k)$ of rightward and unrestricted length-maxmal (m, k)-flock patterns, Similarly, we can define the rightward length-maximal (m, k)-flock pattern mining problem.

For (m, k, r)-versions of the above flock pattern mining problems, we take a heuristic approach that we first enumerate all length-maximal (m, k)-flock patterns P in a database S, and then, for each found pattern P, we check if P satisfies the support constraint $\geq r$.

We easily observe that the number M of solutions of the problem, the number of m-width flock patterns appeared in an input database S, can be exponential in the input size $N = nT$ of S.

2.5 Model of computation

Our goal is to develop a *high-throughput* and *light-weight* pattern mining algorithm. This is captured by the framework of enumeration algorithms [3, 14] as follows. Let N and M be the input size and the number of patterns as solutions.

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

```
1: procedure BasicFPM(ID, S, m, k)
2:   for b_0 ← 1, ..., T do                    ▷ Each time in 𝕋
3:     for i_0 ← 1, ..., n do                  ▷ Each id in ID
4:       RecFPM({i_0}, b_0, i_0, ID, S, m, k);

5: procedure RecFPM(X, b, ID_0, S, m, k)
6:   P = (X, [b, e]) ← RightExt(X, b, m, S);
7:   if len(P) < k then
8:     return ;          ▷ P is not an (m, k)-flock pattern
9:   output P;
10:  I ← ID_0;
11:  while I ≠ ∅ do
12:    Select i = min(I); I ← I − {i};
13:    RecFPM(X ∪ {i}, b, I, S, m, k);
14:  end for
```

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 $p(N)$.

3. ALGORITHMS

In this section, we present efficient algorithms for finding the rightward and unrestricted length-maximal (m,k)-flock patterns, abbreviated as (m,k)-RFPs and (m,k)-UFPs, in a given trajectory database.

3.1 A basic algorithm for rightward length-maximal flock patterns

In Algorithm 3, we present our basic depth-first mining algorithm BasicFPM (basic flock pattern miner) and its subprocedure RecFPM for finding all rightward length-maximal (m,k)-flock patterns, or (m,k)-RFP, from a trajectory database S in poly-delay and poly-space for maximum width $m > 0$ and minimum length k.

Given a trajectory database $S = \{ s_i \mid i = 1, \ldots, n \}$, the algorithm BasicFPM invokes the subprocedure RecFPM with a singleton RFP $P_0 = (\{i_0\}, b_0, *)$ with missing end position $e_0 = *$ for every combination of trajectory id i_0 in ID and starting time b_0 in $\mathbb{T}$. Then, the recursive subprocedure RecFPM searches for all descendant RFPs of P_0 from smaller to larger according to pattern growth approach (e.g., [12]) in depth-first manner as follows.

Receiving a partial RFP $P_* = (X, b, *)$ as arguments, the recursive procedure RecFPM in Algorithm 3 tries to find the unique (m,k)-RFP version P of P_* as follows.

- First, it computes the RFP-version $P = (X, [b,e])$ from P_* by the procedure RightExt with the maximum width parameter m. This step correctly computes the RFP-version as expected according to Condition (1) of Lemma 3.
- Next, if the obtained pattern P satisfies the minimal length k, then it is a proper (m,k)-RFP belonging to $\mathcal{RFP}(m,k)(S)$, and thus output at Line 9. Otherwise, we sefely prune the subtree of descendants by the anti-monotonicity of Lemma 1.
- Finally, RecFPM expands the current ID set X by adding a new id i for every trajectory id $i \in ID$ that is not used so far. To avoid duplication of patterns, the id i is removed from the universe ID in search for descendants.

From Lemma 3, we know that any (m,k)-RFP $P = (X, [b,e])$ is reconstructed from X and b by the procedure RightExt in Algorithm 1. Actually, we see the *closure property* that $\mathsf{RightExt}(X, b, m, S) = P$ holds.

Hence, we can find all (m,k)-RFPs appearing in S as follows: We systematically enumerate all combinations $(X, b) \in 2^{ID} \times \mathbb{T}$ of ID set X and time point b in depth-first search, and whenever a pair (X, b) is enumerated, we compute the RFP $P = \mathsf{RightExt}(X, b, m, S)$ with diameter m. Then, output P if it satisfies the length constraint k and prune all descendants. This is what RecFPM exactly does.

We show the correctness of the algorithm based on a similar argument to Avis and Fukuda [3]. We show the next theorem.

LEMMA 6. (UNIQUE PARENT) *Let $k \geq 2$, $m > 0$, and $k \geq 0$. For any (m,k)-RFP $Q = (X, [b,e])$ in S with support k, there exists some (m,k)-RFP P with support $k-1$. Furthermore, P is obtained from Q by computing $P = \mathsf{RightExt}(X - \{i\}, b, m, S)$, where $i = \max(X)$ is the largest element of X.*

PROOF. From Lemmas 1, 2, 3, and 6. □

In the above lemma, we refer to the (m,k)-RFP P as the *parent* of Q, whereas Q is a *child* of P. Now, we give a tree shaped search route $\mathcal{T}_S$ for all (m,k)-RFPs in a given database S as follows. The *search graph* is a DAG $\mathcal{T}_S = (\mathcal{V}, \mathcal{E}, \mathcal{I})$ satisfying the following conditions: (i) The vertex set is the set $\mathcal{V} = \mathcal{RFP}(m,k)(S)$ consisting of all (m,k)-RFPs in S. (ii) The edge set $E \subseteq V^2$ is a set of all directed edges defined as: for any patterns P, Q in $\mathcal{V}$, there is a directed (reversed) edge from Q to P if and only if P is a parent of Q in the sense of Lemma 6. (iii) The set of roots $\mathcal{I} = \{ \{i_0\} \mid i_0 \in ID(S) \}$ is the collection of all singleton ID sets.

LEMMA 7. *$\mathcal{T}_S$ is the spanning forest for all rightward length-maximal flock patterns appearing in S with root nodes $\mathcal{I}$.*

From the lemmas above, the recursive procedure RecFPM in Algorithm 3 correctly similates the depth-first search over the spanning tree $\mathcal{T}_S$. Hence, we have the following theorem.

THEOREM 1. (POLY-DELAY AND POLY-SPACE MINING FOR RFP) *Given a trajectory database S, max-width $m > 0$ and min-length k, the algorithm* BasicFPM *in Algorithm 3 finds all rightward length-maximal (m,k)-flock patterns of $\mathcal{RFP}(m,k)(S)$ in S in $O(knT)$ time per pattern without duplicates using $O(k^2)$ words of space, where $k = |X|$ is the size of ID set being enumerated, $n = |ID|$, and $T = |\mathbb{T}|$.*

For any $d \geq 2$, we can generalize the above theorem for trajectories in the d-dimensional space $\mathbb{R}^d$ by adding extra $O(d)$ terms to both delay and space.

COROLLARY 2. *The legth-maximal flock pattern enumeration problem for the class of (m,k)-RFPs is poly-delay and poly-space enumerable.*

This is because the correctness of the algorithm depends only on the computation of $||\,A\,||_\infty$ for a point set $A \subseteq \mathbb{R}^d$, which can be done in $O(d|A|)$ by taking the minima and maxima separately in each coordinates by scanning all points of A.

3.2 Mining unrestricted length-maximal flock patterns with filtering

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 $\mathcal{UFP}(m,k)(S)$ in poly-enum and poly-space.

From Lemma 2, we already know that any (m,k)-UFP is also a proper (m,k)-RFP. Conversely, Lemma 8 below shows that exactly when a given (m,k)-RFP is a proper (m,k)-UFP.

LEMMA 8. (FILTERING LEMMA) *A rightward length-maximal (m,k)-flock pattern $P = (X,[b,e])$ in S is also unrestricted length-maximal in S if and only if $||\,S[X][b-1]\,||_\infty > m$.*

PROOF. The proof immediately from Lemma 2. □

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 3 that is modified by replacing Line 9

9: **output** P;

with the following code for leftward extension test:

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

From Lemma 8, we can show the correctness of the modified procedure UnrestRecFPM for all (m,k)-UFPs belonging to $\mathcal{UFP}(m,k)(S)$ since UnrestRecFPM finds all (m,k)-RFPs as candidate, tests the discovered RFPs, and discards all non (m,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 $\#\mathcal{RFP}(m,k)(S)$ and $\#\mathcal{UFP}(m,k)(S)$ the numbers of all (m,k)-RFPs and all (m,k)-UFPs in a given database S.

In Example 1 and Fig. 2, we observe that an (m,k)-UFP has a set of *equivalent* (m,k)-RFPs with the same ID set X and end time e. Generalizing this observation, we have the following theorem, where $T = |\mathbb{T}|$.

LEMMA 9. *For any database S, we have the inequality*

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

PROOF. From the proof of Lemma 3, every unrestricted length-maximal (m,k)-flock pattern P in $\mathcal{UFP}(m,k)(S)$ that has length ℓ can have at most ℓ rightward length-maximal (m,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 $\#\mathcal{UFP}(m,k)(S)$ is not much larger than $\#\mathcal{RFP}(m,k)(S)$. Therefore, we have the following theorem for UFPs.

THEOREM 3. (POLY-DELAY AND POLY-SPACE MINING FOR UFP) *Let S be a trajectory database, $m > 0$ a max-width, and k a min-length. Then, there exists some algorithm that finds all unrestricted length-maximal (m,k)-flock patterns in S in $O(knT^2)$ time per pattern without duplicates using $O(k^2)$ words of space, where $k = |X|$ is the size of ID set being enumerated, $n = |ID|$, and $T = |\mathbb{T}|$.*

PROOF. From Lemma 9, for finding each (m,k)-UFP as solution, we test at most T (m,k)-RFP as candidate by using UnrestRecFPM as subprocedure that requires $O(knT)$ time per RFP. This completes the proof. □

3.3 Modified algorithms with geometric pruning

In this subsection, we present modifications of the basic algorithm BasicFPM for (m,k)-RFPs and for (m,k)-UFPs in the previous subsections using a geometric pruning technique described below.

In Algorithm 3, we present our modified mining algorithm GridFPM (grid-based flock pattern miner) based on geometric constraint on the 2-dimensional plane for (m,k)-patterns. The algorithm uses RecFPM in Algorithm 3 as subprocedure.

Algorithm 3 An algorithm GridFPM for finding all length-maximal (m,k)-flock patterns appearing in a given trajectory database S with ID for maximum width m and minimum length k.

```
1: procedure GridFPM(X, b, k, ID, S, m, k)
2:   Let S = { s_i | i = 1, ..., n };
3:   for b_0 ← 1, ..., T do                          ▷ Each time in T
4:     Build a grid index for point set S_0 ← S[b_0];
5:                                  ▷ the set of all points at time b_0
6:     for i_0 ← 1, ..., n do                        ▷ Each id in ID
7:       p ← s_{i_0}[b_0];   δ ← m;                  ▷ initial point p
8:       R ← [p.x−δ, p.x+δ]×[p.y−δ, p.y+δ]; ▷ rectangle
9:       ID_0 ← S[b_0].RangeQuery(R);
10:      RecFPM({i_0}, b_0, i_0, ID_0, S, m, k);
11:    end
12:  end
```

Given a trajectory database S, maximum width $m > 0$ and minimum length k as arguments, the algorithm GridFPM starts with selecting a combination (i_0, b_0) of a trajectory id and a starting time as in BasicFPM.

Let $p_0 = s_{i_0}[b_0]$ be the the point of the i_0-th trajectory at time b_0 from which we start the search. Let $P = (X, [b_0, *])$ be any (m,k)-pattern that will be found during the search from the initial pattern $(\{i_0\}, [b_0, *])$. From the maximum width constraint with m and the geometry of the plane $\mathbb{R}^2$, we see that for any trajectory s_i in $D_S(X, [b, e])$ with its id i in X, the point $q = s_i[b_0]$ at time b_0 must belong to the $m \times m$ rectangular region

$$R = [p.x-\delta, p.x+\delta] \times [p.y-\delta, p.y+\delta] \subseteq \mathbb{R}^2, \tag{5}$$

where $\delta = m$.

Let $S[b_0]$ be the set of all points in trajectories of S that appear at the specified time b_0. From the above arguments, we can restrict the domain ID of the candidate trajectory ids to form an ID set X to the set of

$$ID_0 = \{\, i \in ID \mid s_i[b_0] \in S[b_0] \cap R \,\}. \tag{6}$$

By using an appropriate geometric index data structure, such as the quad tree or the range tree [6], we can compute this set ID_0 by making the range query $S[b_0].RangeQuery(R)$ on the point set $S[b_0]$ in $q = O(\log^2 \sigma)$ time or $O(\log^2 n)$ time depending on the index used, after $O(n \log n)$ time preprocessing of S, where n and $\sigma > 0$ are the number and maximumm distance of points in $S[b_0]$.

The above modification on the algrithm BasicFPM to obtain GridFPM does not change the order of the algorithm, but greatly reduces the time complexity of the algorithm as shown in Sec. 4. This requires $O(n \log n)$ preprocessing time at Line 4 for T time points and $O(\log^2 n)$ query time for $O(nT)$ time points at Line 9. Therefore, overall overhead becomes $O(nT(\log n + \log^2 n)) = O(N \log^2 n)$ time, which is linear in input size N with polylogarithmic factor.

We analyze the efficiency of this geometric pruning. Suppose that we put n random points in an $a \times a$ area $\mathbb{A}$ on the plane uniform randomly. Then, the expected number of points in any $2m \times 2m$ rectangular region R on $\mathbb{A}$ is approximately ρn, where $\rho = |R|/|\mathbb{T}| = (2m/a)^2 \leq 1$ is the ratio of the areas of R to $\mathbb{T}$. If the distribution of points is close to uniformly random, $\rho \ll q$ if $m \ll a$. This reduces the number of trajectories in an initial database by a factor of ρ, and expected to reduce the total running time of the algorithm on average. The experimental results in Sec. 4 seems to justify this observation.

4. EXPERIMENTS

We ran experiments on synthesis datasets to evaluate the efficiency of our algorithms.

4.1 Data

We used a sets of synthesis trajectory datasets using our data generator implemented in C++. Our data set is a collection of random trajectories in which copies of random patterns are implanted as follows. We first fixed $a \times a$ area $\mathbb{A}$ in the plane, where the width of the area is $a = 100.0$, and generated a set of $n = 1000$ trajectories on $\mathbb{A}$. Each trajectory has length $T = 250$ whose points locates uniform ramdom postions in $\mathbb{A}$. Next, as patterns, we generated a set M of $K = 6$ random shorter *master trajectory* with length $L_* = 200$, and for each master trajectory s in M, we embedded $C = 5$ copies of s whose location is randomly perturbated within a given width $m = 1.0$. The other parameters are varied in experiments.

4.2 Methods

We implemented our algorithms BasicFPM (BFPM) and GridFPM (GPM) in C++ and compiled by g++ of GNU, version 4.6.3. We used a PC with a Intel(R) Xeon(R) CPU E5-1620, 3.60GHz with 32GB of memory, OS Ubuntu Linux, version 12.04. We used a simple grid-based geometric index implemented in C++, where the plane is devided into $b \times b$ grid cells, and the cell corresponding is looked up by a constant time random access followed by a sequential scan of a point list with the value $b = 10$ most time.

We used the following default parameters otherwise stated: the data generator uses length $L* = 220$, width $m_* = 1.0$, the area width $a = 10.0$ for embedded patterns, the input size of $n = 500$ trajectories. The algorithm uses width $m = 1.0$, length $k = 11$, and and min-sup is $r = 5$ for patterns.

4.3 Results

4.3.1 Exp 1

In Fig. 3 and Fig. 4, we compare two algorithms BasicFPM (BFPM) and GridFPM (GFPM) for mining all RFPs in a database. In Fig. 3, we show the running time by varying the input size n from 5 to 1000 trajectories. In Fig. 4, we show the running time by varying the maximum width m of patterns, where we used a wider area $\mathbb{T}$ with width $a = 100.0$. From these results, we observe that the modified algorithm GridFPM using geometric pruning is more than ten times faster than the basic algorithm BasicFPM. In actual running time, for example, GridFPM finds all RFPs in a database of $n = 1000$ trajectories of length $T = 250$ with totally 1/4 million points in less than 10 seconds on a PC.

4.3.2 Exp 2

In Fig. 5 and Fig. 6, we compare the performance of GridFPM for three classes of patterns, FPs, RFPs, and UFPs. In the

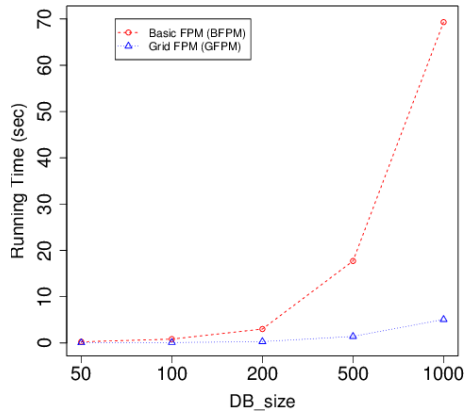

Figure 3: Exp 1: The running time of algorithms BasicFPM (BFPM) and GridFPM (GFPM) by varying the number n of input trajectories from 50 to 1000 up to totally 250,000 points.

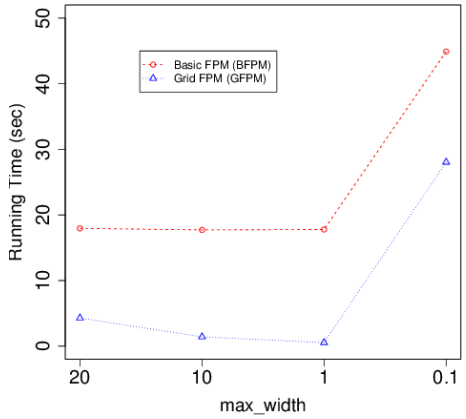

Figure 4: Exp 2: The running time of BasicFPM (BFPM) and GridFPM (GFPM) by varying the maximum width m of patterns from 0.1 to 20.0, that is, 1% to 20% in ratio to the width of the whole area.

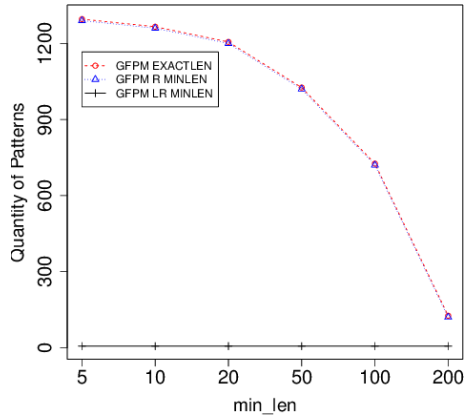

Figure 5: Exp 3: The number of patterns found in GridFPM by varying the minimum length of patterns for three tasks GFPM EXACTLEN, GFPM R MINLEN and GFP LR MINLEN.

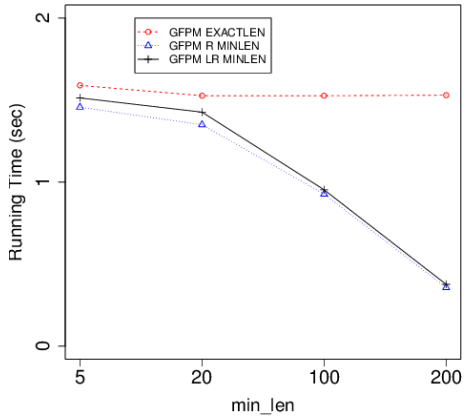

Figure 6: Exp 3: The running time of GridFPM by varying the minimum length of patterns for three tasks GFPM EXACTLEN, GFPM R MINLEN and GFP LR MINLEN.

experiments, the data generator used a longer embedded pattern length parameter $k* = 220$, and a mining algorithm ran with parameter k from 5 to 200 time points. We designed three diffrent tasks with given parameter k:

- GFPM EXACTLEN: finding all FPs with length exactly k.
- GFPM R MINLEN: finding all RFPs with length $\geq k$.
- GFP LR MINLEN: finding all UFPs with length $\geq k$.

In Fig. 5, we show the number of patterns and the running time against exact or minimum pattern length k. Then, the number of patterns is around a small constant value for UFPs since it correctly extracts one UFP of length $k = 200$ per embedded patterns of length $k = 220$, while the numbers patterns increase for FPs and RFPs as k gets smaller since many patterns appear in each embedded pattern. The running time behave in a similar way: the algorithm rans slow for FPs, and quite faster for RFPs and UFPs.

It is interesting to note that the running times for RFPs and UFPs in Fig. 6 are almost identical, inspite of the large difference between the numbers of patterns for RFPs and UFPs in Fig. 5. This is because the pruning condition of Lemma 8 (filtering lemma) in Sec. 3.2 is not anti-monotone, and thus, this method cannot prune the whole subtree of descendants. Hence, it is an interesting future research to devise more efficient pruning method for miningn UFPs.

4.3.3 *Summary of Experiments*

Overall, both of algorithms BasicFPM and GridFPM efficiently finds RFPs and UFPs. Paricularly, the geometric pruning technique introduced in Sec. 3.3 improves the performance of GridFPM more than ten times than that of BasicFPM. In actual running time, GridFPM finds all RFPs in a trajectory database with totally 1/4 million points in less than 10 seconds on a PC.

5. 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. We also give a speed-up technique using geometric pruning of search space, which improves the performance of the algorithm more than ten times faster.

As seen in Corollary 2, 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 [14], 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, 16]. 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.

6. REFERENCES

[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] A. O. C. Romero. Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach. Master's thesis, University of Twente, March 2011.

[14] 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.

[15] 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.

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

[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.