Enumeration of Complete Set of Flock Patterns in Trajectories

Xiaoliang Geng
Graduate School of IST, Hokkaido University
N14 W9 Sapporo, Hokkaido, Japan
gengxiaoliang@ist.hokudai.ac.jp

Takuya Takagi
Graduate School of IST, Hokkaido University
N14 W9 Sapporo, Hokkaido, Japan
tkg@ist.hokudai.ac.jp

Hiroki Arimura
Graduate School of IST, Hokkaido University
N14 W9 Sapporo, Hokkaido, Japan
arim@ist.hokudai.ac.jp

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

ABSTRACT

In this paper, we consider the problem of mining the complete set of spatio-temporal patterns, called *maximal-duration flock patterns* (Gudmundsson and van Kreveld, Proc. ACM GIS 2006) from massive mobile GPS location streams. Such algorithms are useful for mining and analysis of real-time geographic streams in geographic information systems. Although a polynomial time algorithm exists for finding a maximal-duration flock pattern from a collection of trajectories, it has not been known whether it is possible to find *all* maximal-duration flock patterns with theoretical guarantee of its computational complexity. For this problem, we present efficient depth-first algorithms for finding all maximal-duration patterns in a collection of trajectories without duplicates that run in polynomial time per discovered pattern using polynomial space in the total size of input trajectories. To achieve the output-sensitive complexity above, our algorithms adopt depth-first search strategy to avoid the use of exponentially large memory. We also propose a speed-up technique using geometric indexes. Finally, we show experimental results on artificial data to evaluate the proposed algorithms.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management—*Data Mining*

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Data mining, geostream mining, GPS-trajectory, spatio-temporal data, pattern mining algorithms, knowledge discovery and databases

IWGS'14, November 04-07 2014, Dallas/Fort Worth, TX, USA

ACM 978-1-4503-3139-5/14/11· · · $15.00
http://dx.doi.org/10.1145/2676552.2676560 .

1. INTRODUCTION

By rapid increase of amount and species of rapid, unstructured, and heterogeneous data streams in the real world, there have been emerging interests in efficient mining methods for mining interesting patterns from such unstructured data streams. Especially, it is challenging to mine complex spatio-temporal patterns from massive geographic data streams due to their continuous nature, and has attracted much attention for the last decade in the area of data mining, computational geometry, and geographic information systems [6, 8, 9, 15].

The class of flock patterns in a trajectory database

In this paper, we consider the problem of mining the class of *maximal-duration flock patterns*, which is a variant of longest-duration flock patterns introduced by Gudmundsson and van Kreveld [7], defined as follows. Let $\mathcal{M} = (D, \delta)$ be the d-dimensional continuous space $D = \mathbb{R}^d$ associated with the L_∞-metric $\delta = \delta_\infty$, where $\delta_\infty(p, q) \geq 0$ is the L_∞-distance [1] between a pair of points p and q in D. For some integers n and $T \geq 1$, we denote by $\mathbb{O} = \{1, \ldots, n\}$ and $\mathbb{T} = [1..T] = \{1, \ldots, T\}$ the domains of all moving object IDs (or objects themselves) and all time points, respectively. Then, the input *trajectory database* (the *database*, for short) is a collection

$$S = \{\, s_i = s_i[1] \cdots s_i[T] \mid i \in \mathbb{O} \,\} \subseteq D^T,\ n \geq 1 \qquad (1)$$

of GPS-trajectories of n moving objects $o_1, \ldots, o_n$, where for every moving object ID $i \in \mathbb{O}$, the i-th *trajectory* $s_i = s_i[1] \cdots s_i[T] \in D^T$ is a polygonal line consisting of T vertices $s_i[1], \ldots, s_i[T] \in D$ that represents the locations of the moving object $i \in \mathbb{O}$ sampled at T consecutive time steps $\tau_1 < \ldots < \tau_T$.

The basic idea behind a flock pattern is that a highly correlated group of objects travel together within a small neighbor during a certain length of time [7].

Definition 1. **(flock patterns [7])** For any integers $m, k \geq 1$ and real number $r > 0$, an (m, k, r)-*flock pattern* (or a *flock pattern*, for short) [7] is a pair $P = (X, I)$ of a set $X \subseteq \mathbb{O}$ of moving object IDs and a time interval $I = [b, e] \subseteq \mathbb{T}$, $1 \leq b \leq e \leq T$ such that

(i) X consists of at least m moving objects,

(ii) the duration of I, $len(I) = e - b + 1$, is at least k, and

[1]All results in this paper can be generalized to the case of L_2-distance for any fixed dimension $d \geq 2$.

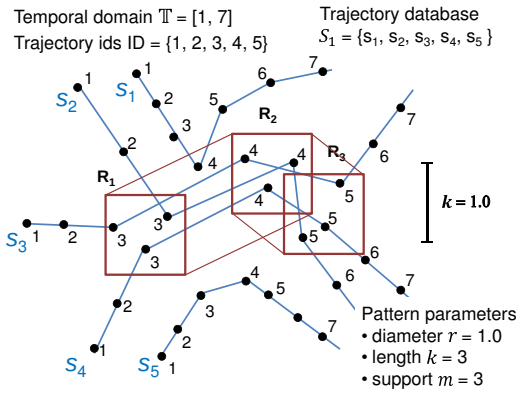


Figure 1: An example of a trajectory database $\mathcal{S}_1$ and an (m, k, r)-flock pattern in the plane. The pattern consists of $m = 3$ entities associated with an interval with length $k = 3$, and has radius $r = 1.0$ for enclosing disks at each sampling time. The points and numbers on each trajectory are the location of the entity at sampled points and the corresponding time stamps.

(iii) for every time point t within I, there exists some disk with radius r that contains the locations of all m objects.

In Fig. 1, we show an example of flock patterns in the two-dimensional space, $P_1 = (X_1, I_1)$ consisting of three moving objects $X_1 = \{3, 4, 5\}$ and duration $I_1 = [3, 5]$ of length three.

For a flock pattern $P = (X, I)$ with $I = [b, e]$, we denote by $P.set = X$ the *set of moving objects*, $P.span = I$ the *duration* or *span*, $P.start = b$ the *start time*, $P.end = e$ the *end time*, $len(P) = len(I) = e - b + 1$ the *length*, and $size(P) = |P.set|$ the *size* of P. The flock pattern obtained from P by simply adding object i is denoted by $P \cup \{i\}$ and called the *extension* of P by i.

The *radius* of pattern P in an database is an important concept in flock pattern mining [7], defined as follows. In L_∞-metric, the *radius* of point set U, denoted by $radius(U)$, is the half of the length of longest side of the minimum bounding box (MBB) for U, while in the standard L_2-metric the radius of U is the radius of the minimum enclosing circle containing U (See [5]). For the trajectory database $S = \{s_1, \ldots, s_n\}$ and any $t \in I$, we denote the set of all location points at time point t for all moving objects in X by

$$S[X][t] \triangleq \{\, s_i[t] \,|\, i \in X \,\} \subseteq D. \tag{2}$$

We call this set the *snapshot* of P at time t. Then, the *radius* of P, denoted by $radius(P)$, is the maximum of the radius of the snapshot of P at time t over all $t \in I$. That is,

$$radius(P) \triangleq \min_{t \in I} radius(S[X][t]) \geq 0. \tag{3}$$

In the case of L_2-metric, we can also define the radius of P in the same way. By definition, we see that an (m, k, r)-flock pattern is a pair $P = (X, I)$ with $X \subseteq \mathbb{O}$ and $I \subseteq \mathbb{I}$ such that $size(P) \geq m$, $len(P) \geq m$, and $radius(P) \leq r$.

We also refer to the object set X as an (m, k, r)-*flock pattern* in a time interval I, If one of the parameters is $*$, this means that there is no constraint on the parameter. For example, a $(*, *, r)$-flock pattern represents those flock patterns with radius $\leq r$, but no restriction on their size and length. We refer to such a pattern as an r-*flock pattern*,

The problem: maximal-duration flock pattern mining

Gudmundsson and van Kreveld [7] introduced a variant of flock pattern, called the *longest-duration* flock pattern as follows, with the motivation to avoid generating huge number of patterns. Given $m, k \in \mathbb{N} \cup \{*\}$ and $r \in \mathbb{R}$, a *longest-duration* (m, k, r)-*flock pattern* in a database S is an (m, k, r)-flock pattern whose duration is longest among all (m, k, r)-flock patterns appearing in S. They showed that the problem of finding at least one longest-duration $(m, *, r)$-flock pattern in a given database S is NP-hard. This means that there will be no efficient algorithm to find longest-duration flock patterns in polynomial time in the input size unless $P = NP$ [7].

To overcome the computational difficulty associated to the discovery of longest-duration flock patterns above, we relax the problem by replacing the requirement for the existence for the global *maximum* with the local *maximal* of the duration. Now, we introduce the definition of maximal-duration flock patterns.

Definition 2. **(maximal-duration flock pattern)** For any real number $r > 0$, an r-flock pattern $F = (X, I)$ is said to be a *maximal duration r-flock pattern* in the database S if there is no other r-flock patterns with the same subset as F, but its duration properly includes the duration of F. Formally, F is a maximal-duration r-flock pattern if there is no r-flock pattern $F' = (X', I')$ in S that satisfies the conditions $X' = X$, $I \subseteq I'$, and $|I| < |I'|$.

In other words, a maximal-duration flock pattern is such a flock pattern that we cannot extend its span further in either side without violating the radius constraint.

In what follows, we denote by $\mathcal{FP}_r$ and $\mathcal{MFP}_r$ the classes of all r-flock patterns and all maximal-duration r-flock patterns in a given database S. Clearly, we see that the inclusion $\mathcal{MFP}_r \subseteq \mathcal{FP}_r$ holds, while the converse does not hold in general.

Since there are exponentially many maximal-duration flock patterns in the number m of their entities, it is natural to consider the enumeration of all of such patterns without duplicates.

Definition 3. **(maximal-duration flock pattern enumeration problem)** The enumeration problem for the class of maximal-duration flock patterns is the problem of listing all maximal duration r-flock patterns appearing in an input trajectory database S without duplicates.

Sometimes, it is useful to put a constraint on the minimum length of a pattern to the above pattern enumeration problem. We refer to such a variant as the maximal-duration flock pattern enumeration problem *with minimum length constraint*.

In general, there are exponentially many solutions for an enumeration problem. Thus, as a measure of efficiency of an enumeration algorithm $\mathcal{E}$, it is meaningless to measure the total running time of $\mathcal{E}$ because it is obviously exponential time even for $\mathcal{E}$ to just print out all outputs. Instead, we adopt *output-sensitive complexity* of enumeration algorithms, which has been used in analysis of algorithms in database and data mining producing large outputs [1, 13].

Our goal here is to devise an efficient enumeration algorithm that finds all maximal-duration flock patterns in polynomial time per pattern and polynomial space in the total size N of the input database S. Especially, we want to make the running time per pattern close to the description length $||P|| = dmk$ of each flock pattern P with size m and length k, which is usually much smaller than the total input size $N = ||S|| = dnT$.

We note that a straightforward breadth-first search algorithm with tabulation can solve the enumeration of maximal-duration flock patterns by memorizing all intermediate solutions in a table with

exponential size to avoid duplicated and non-maximal answers. However, it is a challenging task to achieve output-sensitive complexity without using such an exponential size table.

Main results

As a main result, we present a variety of depth-first mining algorithms for the maximal-duration r-flock pattern enumeration problem. For any d-dimensional continuous space $(\mathbb{R}^d, \delta_\infty)$ with dimension $d \geq 2$ and L_∞-metric, the algorithm FPM-MP finds all maximal-duration r-flock patterns (r-MFPs) appearing in an input trajectory database S in $O(dkn)$ time per pattern P using $O(dm^2)$ additional space, where n is the number of all entities, T is the trajectory length, $N = nT$ is the total input length, m and k are the size and the length of a pattern found, respectively.

We also show that a modified algorithm FPM-MG with practical speed-up technique using spatial indices. By simple average-case analysis, it finds all r-MFPs in $O(dk)$ time per pattern on average using $O(n \log^{d-1})n$ space and $O(N \log^d n)$ preprocessing for databases of uniform randomly generated transactions when the density of points and radius parameter are constant. We can extend the above results to L_2-metric for any fixed dimension $d = O(1)$. [2]

Since the time per solution is exact, but not amortized, the algorithms have polynomial delay and space complexity in the input size. To the best of our knowledge, this is the first result concerning to such polynomial delay and space algorithms for complete mining of maximal-duration flock patterns in a trajectory database for every $d \geq 2$.

Finally, we ran experiments on synthetic data to compare the efficiency of the proposed algorithms and the previous flock pattern mining algorithm BFE (Vieira et al. [14]) for non-maximal-duration flock patterns. The results showed that the introduction of maximal-duration patterns and speed-up techniques were effective to improve the performance, and that the proposed algorithms were much faster than BFE and our basic versions.

Organization

This paper is organized as follows. Sec. 2 gives related work. Sec. 3 introduces some definitions and notations. Sec. 4 presents a basic polynomial delay and space algorithm for finding all maximal-duration flock patterns. Then, Sec. 5 presents an improved algorithm for the problem. Sec. 6 describes a speed-up technique using spatial index. In Sec. 7 shows experimental results. Sec. 8 concludes this paper.

2. RELATED WORK

Since the seminal work of Laube *et al.* [8] on a variation of group patterns with duration length $k = 1$, called *flock, meet, diverge*, and *leadership*, there have been a number of efficient algorithms for mining flock patterns in an input collection of 2-dimensional trajectories [3, 7, 12, 14]. However, there seem no algorithms that simultaneously achive complete mining and polynomial delay and space complexity for subclasses r-flock patterns for arbitrary dimension $d \geq 2$ and duration length $k \geq 1$.

Benkert *et al.* [3] studied mining of the class of (m, k, r)-flock patterns with arbitrary duration length $k \geq 1$. They presented approximation algorithms for finding a subset of (m, k, cr)-flock patterns for some constant $c > 1$ whose running time is polynomial in n, T and $\frac{1}{\varepsilon}$, but exponential in the duration length k. Moreover, these algorithm can find only a part of solution patterns whose centers locates on some trajectories, while there are at most nT such patterns.

The complexity of mining of the class of *longest-duration* $(m, *, r)$-flock patterns was first studied by Gudmundsson and van Kreveld [7]. They proved that the search problem for a *longest*-duration flock pattern, not *maximal* ones, in a given database becomes NP hard even if we allow $(2 + \varepsilon)$-approximation of radius r.

For complete mining of flock patterns, Vieira, Bakakov, and Tsotras [14] studied first time such an enumeration problem for flock patterns in slightly different setting, and presented a series of algorithms including BFE algorithm. Although the BFE algorithm seems to achieve polynomial time per patterns as ours, it requires exponential space due to table look-up in order to avoid duplicated answers. Turdukulova, Romero, Huismanc, and Retsiosa [12] study the efficient implementation of complete mining of flock patterns based on frequent itemset mining algorithm, such as LCM [13].

3. PRELIMINARIES

In this section, we prepare some technical definitions that are not explained in the introduction. Other definitions not defined here will be found in textbooks [4, 5, 11].

Basic definitions

We denote by $\mathbb{R}$, $\mathbb{Z} = \{0, -1, +1, \ldots\}$ and $\mathbb{N} = \{0, 1, 2, \ldots\}$, the sets of all *real numbers*, all *integers*, and all *non-negative integers*, respectively. For real numbers $a, b \in \mathbb{R}$ and integers $i, j \in \mathbb{Z}$, the notation $[a, b]$ denotes the *continuous interval* $\{\, x \in \mathbb{R} \,|\, a \leq x \leq b \,\}$, and $[i, j]$ denotes the *discrete interval* $\{i, i + 1, \ldots, j\}$. For a set A, $|A|$ denotes its *cardinality*. We also denote by A^* the *set of all sequences* of zero or more elements on A, and by ε the *empty sequence* with length zero. For a sequence $S = a_1 \cdots a_n \in A^*$ of n elements in A, and indices $i \leq j \leq n$, we define the i-th element by $S[i] = a_i$, the *consecutive subsequence of S from indices i to j* by $S[i..j] = a_i a_{i+1} \cdots a_j$, and the *length* of S by $|S| = n$.

We introduce terms and notations in geometry according to Overmars *et al.* [5]. Let $\mathcal{M} = (D, \delta)$ be a metric space, the d-dimensional continuous space, where D is the set of all d-dimensional points, $\mathbb{R}^d$, and $\delta : D \to [0, \infty)$ is a metric function that assigns the distance $\delta(p, q) \geq 0$ for any pair of points p and q. For simplicity, we consider the metric δ_∞ based on the L_∞-norm defined as: $\delta_\infty(p, q) = \max_i |p.i - q.i|$, where $p.i = x_i$ for $p = (x_1, \ldots, x_d) \in \mathbb{R}^d$. However, all results in this paper can be extended to metric δ_2 based on L_2-norm for fixed dimension d. We assume the space domain is always the metric space $\mathcal{M} = (D, \delta) = (\mathbb{R}^d, \delta_\infty)$.

Enumeration algorithms

According to the literature in enumeration algorithms [2], an enumeration algorithm $\mathcal{E}$ is called *output-polynomial time* (OUT-POLY) if its total running time is bounded by polynomial of both input size N and output size M. $\mathcal{E}$ is called *of polynomial enumeration* (POLY-ENUM) if the time per solution is bounded by polynomial in input size N only. It means that the total running time is in the form $M \cdot poly(N)$. $\mathcal{E}$ is called *of polynomial delay* (POLY-DELAY) if the maximum time between consecutive solution, called the *delay*, is bounded by polynomial in input size N only. This is the worst case version of POLY-ENUM, and is most restricted among all definitions above. All results in this paper are POLY-DELAY. It is also called *polynomial space* (POLY-SPACE) if the maximum space used is bounded by some polynomial in N.

4. BASIC MINING ALGORITHM

In this subsection, we present a basic mining algorithm, called FPM-M (Maximum Duration Flock Pattern Miner with rejection)

[2] This is done by using linear time minimum enclosing ball computation by Megiddo [10].

that solves the maximal-duration flock pattern enumeration problem with minimum length constraint. In Algorithm 2, we show the algorithm FPM-M and subprocedure RecFPM-M. To solve the problem without minimum length constraint, we set the minimum length parameter k to be 1.

In what follows, we denote by $\mathcal{FP}_r$ and $\mathcal{MFP}_r$ the classes of all r-flock patterns all maximal-duration r-flock patterns in a given database S. Clearly, we see the inclusion $\mathcal{MFP}_r \subseteq \mathcal{FP}_r$, while the converse does not hold in general.

4.1 The first characterization of $\mathcal{MFP}_r$

From the view of closed pattern mining, a maximal-duration pattern can be regarded as a kind of *closure operation* for flock patterns. Now, we introduce a closure operation for maximal-duration flock patterns.

Definition 4. **(rightward closure)** For any number $r > 0$, the *r-rightward closure* of an r-flock pattern P is the unique flock pattern $\mathsf{RClosure}(P, S, r) = Q$ satisfying conditions (1)–(3) below:

1. $Q.set = P.set$.
2. $Q.start = P.start$.
3. $Q.end = e_{\max}$, where $e_{\max}$ is the maximum $e \in [P.start, T]$ such that $radius(S[P.set][t]) \leq r$ for every $t \in [P.start, e]$.

Algorithm 1 below computes $Q = \mathsf{RClosure}(P, S, r)$ in $O(dmk) = O(dmT)$ time, where $m = size(P)$ and $k = len(P)$.

Algorithm 1 The algorithm for computing the rightward closure of a given possibly non-maximal flock pattern $P = (X, [t_0, e_0])$, that is the rightward maximal-duration pattern that has the same subset and start time with P.

```
1: procedure RClosure(P = (X, [t_0, e_0]), S, r)
2:     t ← t_0;
3:     while radius(S[X][t]) ≤ r do t ← t + 1;
4:     b ← t_0; e ← t − 1;
5:     return P_max(X, [b, e]);
```

The next lemma tells us how we can accelerate this computation regardless of m by using incremental computation of MBB (minimum bounding box) as follows.

LEMMA 1 (FAST COMPUTATION OF RClosure). *Let $o_1, \ldots, o_m$ be any sequence of moving object IDs for some $m \geq 1$. Starting from the empty pattern $P_0 = (\emptyset, [t, T])$ for some start time t, we can iteratively compute for every $i = 1, \ldots, m$, the i-th pattern*

$$P_i = \mathsf{RClosure}(P_{i-1} \cup \{o_i\}, S, r) \tag{4}$$

in $O(dk)$ time and $O(dm)$ additional space.

PROOF. Let $k = len(P.span)$. We maintain the *envelope* of P, the k-vector of the minimum bounding boxes (MBBs) of snapshots of P defined by

$$envelope_S(P) \triangleq (MBB(S[P.set][t]))_{t \in P.span}. \tag{5}$$

When we add a new trajectory to $P.set$, we can update all MBBs in $O(dk)$ time in total, and thus can compute the radius of P in the same time complexity. □

Next, we introduce a *one-sided version* of maximal-duration r-flock patterns, r-RFPs, as a theoretical tool for analyzing the enumeration of MFPs.

Algorithm 2 The rejection-based polynomial amortized delay and space algorithm for finding all maximal-duration r-flock patterns with minimum length k_{min} in a trajectory database S in dimension $d \geq 1$, where r is a radius parameter and the operator $deletemin(X)$ deletes and then returns $\min(X)$ from X.

```
1: procedure FPM-M(ID = {1, ..., n}, S, r, k_min)
2:     for b ← 1, ..., T do                          ▷ each start time in 𝕋
3:         while ID ≠ ∅ do                           ▷ each id in ID
4:             i ← deletemin(ID);
5:             P ← ({i}, [b, ∗]);                    ▷ initial maximal pattern
6:             RecFPM-M(P, ID, S, r, k_min);

7: procedure RecFPM-M(P = ({i}, [b, ∗]), ID, S, r, k_min)
8:     P_∗ ← RClosure(P, S, r);
9:     if len(P_∗) < k_min then return ;             ▷ backtrack
10:    if radius(S[X][b − 1]) > r then               ▷ maximality test
11:        output P_∗;
12:    ID_1 ← ID;
13:    while ID_1 ≠ ∅ do
14:        i ← deletemin(ID_1);
15:        Create Q; Q.set ← P_∗.set ∪ {i};
16:        Q.start ← P.start; Q.end ← P_∗.end;
17:        RecFPM-M(Q, ID_1, S, r, k_min);           ▷ recursive call
```

Definition 5. An r-flock pattern P in a database S is said to be a *rightward maximal-duration r-flock pattern* (an *r-RFP*) in S if there exists no r-flock pattern Q such that that (i) $Q.set = P.set$, (ii) $Q.start = P.start$, and (iii) the duration of Q is strictly longer than that of P, i.e., $Q \supset P$.

We denote by $\mathcal{RFP}_r$ the classes of all r-RFPs in a given database S. Again, we can easily see the inclusion $\mathcal{MFP}_r \subseteq \mathcal{RFP}_r \subseteq \mathcal{FP}_r$. It is not hard to see that the same inclusion holds even if we add the restriction of minimum pattern length k.

Next lemma gives the characterization of the maximal-duration r-flock patterns in terms of rightward closure, where we define $radius(S[X][t]) = \infty$ for any time outside of the domain $\mathbb{T} = [1, T]$, i.e., $t < 1$ or $t > T$.

LEMMA 2 (CHARACTERIZATION OF $\mathcal{RM}_r$). *An r-flock pattern P is an r-RFP in S if and only if $P = \mathsf{RClosure}(P, S, r)$.*

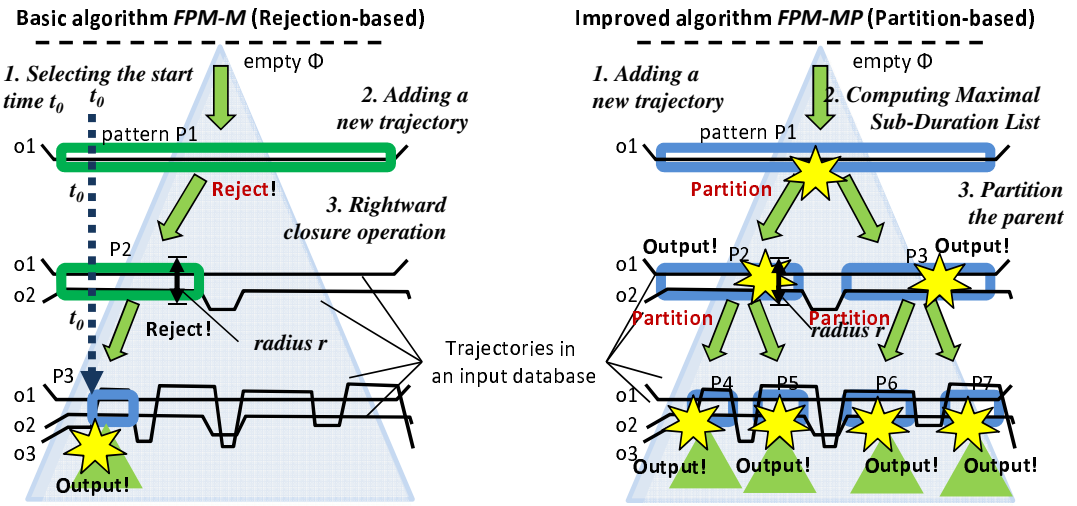
LEMMA 3 (CHARACTERIZATION OF $\mathcal{MFP}_r$). *An r-flock pattern P is a maximal-duration r-flock pattern in S if and only if the following conditions 1–2 hold:*

1. *P is an r-RFP in S.*
2. *$radius(S[P.set][P.start - 1]) > r$. (leftward extension test (*))*

PROOF. For any time t, let w_t be the radius of P at t. By definition, an r-flock pattern P is of maximal-duration if and only if (i) w_t is at most r within the duration $I = [P.start, P.end]$, (ii) $w_t > r$ at $t = P.start - 1$, and (iii) $w_t > r$ at $t = P.end + 1$. We can easily see that above conditions (i), (ii), and (iii) are equivalent to conditions 1 and 2 in the lemma. This shows the lemma. □

4.2 The family forest for $\mathcal{MFP}_r$

In this subsection, we introduce the tree-shaped search space $\mathcal{F}_r^{\mathrm{rm}}$ for all r-RFPs. From the above Lemma 3, in order to find all maximal-duration patterns, we see that it is sufficient to (i) first enumerate each r-RFP P in a trajectory database S, (ii) test if P



Rejection-based approach: application of the extension operation to the parent pattern sometimes prodeces non-maximal patterns such as P2. Then, we eliminate these non-maximal patterns by the left-expansion test as shown in Lemma 6.

Partition-based approach: application of the extension and partitioning operations to the parent pattern always produces maximal-duration flock patterns only based on the computation of the maximal sub-duration list as shown in Lemma 11.

Figure 2: Illustration of Rejection-based approach FPM-M (left, Sec. 4) and Partition-based approach FPM-MP (right, Sec. 5). Both approaches achieve polynomial delay and space enumeration of maximal-duration flock patterns, but have different complexities.

can be extended leftward by the test (*), and (iii) output P if the test succeeds and reject it otherwise.

The remaining problem is how to enumerate all r-RFPs without duplicates. A flock pattern P is called a *singleton pattern* if it contains only one moving object, i.e., $size(P) = 1$, and a *non-singleton pattern* otherwise. The singleton patterns form the initial vertices of our search space $\mathcal{F}_r^{\mathrm{rm}}$. We associate the parent $\mathcal{P}(Q)$ to any non-singleton r-RFP Q in S as follows.

Definition 6. **(parent for $\mathcal{RFP}_r$)** Let Q be any non-singleton r-RFP. Then, we define the parent of Q, denoted by $\mathcal{P}(Q)$, by $\mathcal{P}(Q) \triangleq \mathsf{RClosure}(R)$ such that $R.set = Q.set - \{i_{\max}\}$, i.e., $\mathcal{P}(Q)$ is the closure of the pattern R obtained from Q by removing the maximum ID $i_{\max} = \max(Q.set)$ from $Q.set$. Then, Q is called a *child* of P.

The next lemma immediately follows from the construction.

LEMMA 4. *For any r-RFP Q such that $size(Q) \geq 2$, (i) $\mathcal{P}(Q)$ is well-defined, (ii) is unique, (iii) has a properly smaller subset than Q, (iv) has duration length longer or equal to Q, and (v) is an r-RFP in S, too.*

Recall that $\mathcal{RFP}_r$ is the class of all r-RFPs in the input database S. Now, we introduce a directed graph $\mathcal{F}_r^{\mathrm{rm}} = (\mathcal{V}, \mathcal{P}, \mathcal{I})$, called the *family forest* for all r-RFPs, where:

- $\mathcal{V} = \mathcal{RFP}_r$ is the vertex set consisting of all r-RFPs appearing in the database S.
- $\mathcal{P} : \mathcal{RFP}_r \setminus \mathcal{I} \rightarrow \mathcal{RFP}_r$ is the parent function for the reverse edges $(\mathcal{P}(Q), Q)$ that assigns the parent $\mathcal{P}(Q)$ to any non-singleton RFP Q with $size(Q) \geq 2$.
- $\mathcal{I} \subseteq \mathcal{V}$ is the set of root nodes that are singleton patterns $P \in \mathcal{RFP}_r$ with $size(P) = 1$.

LEMMA 5 (FAMILY FOREST FOR $\mathcal{RFP}_r$). *The* family forest $\mathcal{F}_r^{\mathrm{rm}}$ *is a spanning forest over the class $\mathcal{RFP}_r$ of all r-RFPs in S.*

PROOF. We will show that the directed graph $\mathcal{F}_r^{\mathrm{rm}}$ is connected and acyclic. From condition (iii) of Lemma 4 and the termination property of sizes, we see that starting from any non-singleton maximal pattern Q, any ascending path $Q, \mathcal{P}^1(Q), \mathcal{P}^2(Q), \ldots$ eventually terminates at some singleton maximal pattern P in $\mathcal{I}$ in the graph $\mathcal{F}_r^{\mathrm{rm}}$. Thus, $\mathcal{T}^{\mathrm{rm}}$ is acyclic and contains every maximal pattern in $\mathcal{RFP}_r$ as its node. This completes the proof. □

4.3 A rejection-based polynomial delay and space algorithm for $\mathcal{MFP}_r$

For complete enumeration of all RFPs, we can perform depth-first search (DFS) of the tree-shaped search space $\mathcal{F}_r^{\mathrm{rm}}$ that we have just introduced. Then, we can filter out all non-maximal RFPs by the characterization of MFPs in Lema 3 to obtain MFPs.

For DFS of the search space, the remaining thing is how to efficiently generate all children Q of a given P. Q is an *expansion* of P if $Q.set = P.set \cup \{i\}$ for some ID $i \notin P.set$. Then, we write $Q = P \cup \{i\}$. An expansion Q of P is a *tail expansion* of P if $Q = P \cup \{i\}$ for some $i > \max(P.set)$ [16]. Using the tail expansion, it is well-known that we can systematically generate all subsets X of a set ID in backtracking [13, 16]. The following lemma answers the above question.

LEMMA 6 (GENERATION OF A CHILD). *For any r-RFP $P, Q \in \mathcal{RFP}_r$, (a) P is the parent of Q if and only if (b) Q is obtained from P by $Q = \mathsf{RClosure}(P \cup \{i\}, S, r)$ for some ID $i \in ID$ such that $i > \max(P.set)$.*

By the above lemma, we can perform DFS of $\mathcal{F}_r^{\rm rm}$ based on backtracking using a stack of parent patterns. we analyze an upper-bound of the size of the family of all r-RFPs in terms of the size of the family of all r-MFPs.

LEMMA 7. *In any trajectory database S with the time domain $[1, T]$, the inequality $|\mathcal{RFP}_r| \leq |\mathcal{MFP}_r| \cdot T$ holds.*

Combining the above arguments, we obtain Algorithm 2. In order to give the estimation of an upper bound of the amortized delay of Algorithm 2, Now, we have the main theorem of this section.

THEOREM 1. (BASIC POLYNOMIAL DELAY AND SPACE ALGORITHM) *For any trajectory database S, number $r > 0$, and dimension $d \geq 1$, the algorithm* FPM-M *shown in Algorithm 2 finds all maximal-duration r-flock patterns with length at least k_{min} in S without duplicates in $O(dnkT) = O(dnT^2)$ amortized time per pattern and $O(dm^2)$ additional space, where k and m are the length and subset size of a discovered maximal-duration pattern P.*

PROOF. We first show the correctness. From Lemma 5 and Lemma 6, we can easily see that the algorithm FPM-M visits all r-RFP in S without duplicates at Line 8. From Lemma 3, the algorithm correctly outputs only and all maximal r-flock patterns at Line 8. For complexity, we process an input pattern P to obtain the corresponding maximal pattern P_* if it exists. This can be done in $O(dk)$ incremental time when we add a new object ID i by maintaining the vector $(MBB(S[P.set][t]))_{t \in P.span}$ of the MBBs of the snapshot of P at time t. Since each r-RFP P_* can have at most $n = |ID|$ children and all of them can be failed in the worst case, the worst case time per r-RFP, i.e. the delay, is bounded by $O(dkn)$ time. Thus, Lemma 7 adds extra $O(T)$ factor to the amortized delay. The space complexity follows from that the path from the root to a leaf Q with subset size m has depth at most m. Hence, this completes the proof. □

5. FASTER MINING ALGORITHM

In this subsection, we present an improved mining algorithm, called FPM-MP (Maximum Duration Flock Pattern Miner with Partitioning) that solves the maximal-duration flock pattern enumeration problem with minimum length constraint.

5.1 A problem with the previous algorithm

The basic algorithm FPM-M in Algorithm 2 introduced in the previous section has quadratic time complexity, i.e., $O(nkT) = O(nT^2)$ time, in the length T of input trajectories. One reason of this quadratic complexity is the use of Lemma 3 (characterization of maximal-duration flock patterns by left expansion test). We can use this rule to decide if a given rightward maximal-duration flock pattern is of maximal-duration, but cannot use it for pruning the whole descendants in the family forest even when the test is failed because the property is not monotone. FPM-M incurs extra $O(T)$ term from this fact. To overcome this problem, we give the second characterization of maximal-duration flock patterns using "Merging and Split" operation introduced below.

5.2 The second characterization of $\mathcal{MFP}_r$

For intervals I and $J \subseteq \mathbb{T}$, we say that I is a *sub-interval* of I' if $I'.start \leq I.start$ and $I.end \leq I'.end$ hold. If $I.end < I'.start$, we say that I *precedes* I' and write $I < I'$.

Definition 7. **(left and rightward closure)** Let P be a r-flock pattern. The r-left-rigth-closure of P is the unique flock pattern $\mathsf{LRClosure}(P, S, r) = Q$ satisfying (1) $Q.set = P.set$, and (2) $Q.start \leq P.start$ and $Q.end \geq P.end$, respectively, are the smallest and largest time points satisfying that $radius(S[Q.set][t]) \leq r$ for every $t \in [Q.start, Q.end]$.

Algorithm 3 The partition-based polynomial delay and space algorithm for finding all maximal-duration r-flock patterns with minimum length k_{min} in a trajectory database S in dimension $d \geq 1$, where r is a radius parameter and the operator $deletemin(X)$ deletes and then returns $\min(X)$ from X. For $span = [b, e]$, we denote by $span.start = b$ and $span.end = e$.

```
1: procedure FPM-MP(ID = {1,...,n}, S, r, k_min)
2:     while ID ≠ ∅ do                                  ▷ each id in ID
3:         i ← deletemin(ID);
4:         P ← ({i}, [1, T]);                           ▷ initial maximal pattern
5:         RecFPM-MP(P, ID, S, r, k_min);

6: procedure RecFPM-MP(P, ID, S, r, k_min)
7:     if len(P) < k_min then return ;                  ▷ backtrack
8:     output P;
9:     ID_1 ← ID;
10:    while ID_1 ≠ ∅ do                                ▷ first loop
11:        i ← deletemin(ID_1);
12:        Create Q; Q.set ← P.set ∪ {i};
13:        Q.start ← P.start; Q.end ← P.end;
14:        MSD ← MaxSubDuration(Q, S, r);               ▷ all maximal sub-durations
15:        for each maxspan ∈ MSD do                    ▷ second loop
16:            Create R; R.set ← Q.set;
17:            R.start ← maxspan.start;
18:            R.end ← maxspan.end;
19:            RecFPM-MP(R, ID_1, S, r, k_min);         ▷ recursion
```

LEMMA 8. $Q = \mathsf{LRClosure}(P, S, r)$ *always exists, is unique, and can be computed in $O(mk)$ time, where $m = size(P) = size(Q)$ and $k = len(Q)$.*

As in the previous section, we construct a spanning forest $\mathcal{T}^{\rm m}$ over the class $\mathcal{M}_r$ of all maximal-duration flock patterns. First, we define the parent function.

Definition 8. Let Q be any maximal-duration flock pattern in S. Then, we define the *parent* of Q, denoted $\mathcal{P}(Q)$, by $\mathcal{P}(Q) \triangleq \mathsf{LRClosure}(Q - \{i_{\max}\})$, where $i_{\max} = \max(Q.set)$, and $Q - \{i_{\max}\}$ is the flock pattern obtained from Q by removing $i_{\max}$.

Now, we define the *family forest* for the class $\mathcal{M}_r$ as the directed graph $\mathcal{T}^{\rm rm} = (\mathcal{V}, \mathcal{P}, \mathcal{I})$, where $\mathcal{V} = \mathcal{M}_r$ is the vertex set, $\mathcal{P}$ is the set of reverse edges from children to parents, and $\mathcal{I}$ is the set of roots. Similar to Lemma 5 in Sec. 4, we have the following lemma for the family forest.

LEMMA 9 (FAMILY FOREST FOR $\mathcal{M}_r$). *The* family forest $\mathcal{F}^{\rm m}$ *is a spanning forest over the class $\mathcal{M}_r$ of all maximal-duration r-flock patterns in S.*

To invert the parent-child relationship, we need some technical definitions below. We introduce the notion of $\mathcal{MSD}$ (the maximal r-sub-duration list) of a pattern P as follows. Let P be a flock pattern with arbitrary radius. A sub-duration $I \subseteq P.span$ is said to have radius r if $radius(S[P.span][t]) \leq r$ holds for every $t \in I$. An r-sub-duration I is *maximal* in P if there is no other r-sub-duration I' in P that properly contains I as sub-duration. For intervals $I, J \subseteq \mathbb{T}$, $I < J$ if $I.end < J.start$.

Algorithm 4 The algorithm for computing the set of all maximal r-sub-durations for a given flock pattern Q possibly with radius more than r in trajectory database S, where $r > 0$ is a radius parameter.

```
1: procedure MaxSubDuration(Q, S, r)
2:     MSD ← ∅;
3:     b ← Q.start; X ← Q.set;
4:     while b ≤ Q.end do
5:         Increment b while radius(S[X][b]) > r;        ▷ skip inter-duration time
6:         e ← b;
7:         Increment e while radius(S[X][e]) ≤ r;    ▷ expand a duration rightwards
8:         MSD ← MSD ∪ {[b, e]};
9:         b ← e + 1;
10:    return MSD;
```

Definition 9. **(maximal sub-duration list)** The maximal r-sub-duration list for P, denoted by $\mathcal{MSD}(P, S, r)$, is a sorted list $\mathcal{M} = (I_1, \ldots, I_\ell)$, $I_1 < \cdots < I_\ell$, of non-empty sub-intervals of $\mathbb{T} = [1, T]$ in the preceding order such that

(i) $\max_{t \in I_i} radius(S[P.set][t]) \leq r$ for every $i = 1, \ldots, \ell$.

(ii) any interval of $\mathcal{MSD}(P, S, r)$ cannot be extended in both direction without violating property (i) above.

In Algorithm 4, we show the algorithm for computes $\mathcal{MSD}(P, S, r)$. From the construction of this algorithm, we can show the following lemma.

LEMMA 10. *For any flock pattern P with arbitrary radius, the set $\mathcal{MSD}(P, S, r)$ is uniquely determined, and computable in $O(mk)$ time and $O(m + \ell)$ additional space, where $m = size(P)$, $k = len(P)$, and ℓ is the number of maximal sub-durations in $\mathcal{MSD}(P, S, r)$.*

5.3 An improved partition-based polynomial delay and space algorithm for MFP

Using the maximal sub-duration list, we can efficiently computes any child Q of a given parent maximal-duration flock pattern P as follows.

LEMMA 11 (GENERATION OF A CHILD). *For any maximal-duration r-flock patterns $P, Q \in \mathcal{RM}_r$, (a) P is the parent of Q if and only if (b) $Q.set = P.set$ and $Q.span \in \mathcal{MSD}(P, S, r)$. Furthermore, if $i \neq j$ then the corresponding children are mutually distinct.*

Now, we present the depth-first mining algorithm FPM-MP and subprocedure RecFPM-MP in Algorithm 3. The proposed algorithm RecFPM-MP is a backtracking algorithm similar to the previous algorithm RecFPM-M. The algorithm starts from each initial singleton maximal-duration pattern in $\mathcal{I}$. It then recursively expands the parent maximal-duration flock pattern P by adding a new object ID $i > tail(P)$ to obtain another maximal-duration flock pattern Q with larger subset size as its child. This expansion is done by Lemma 11.

From the above discussion, we have the main theorem on an improved polynomial delay and space algorithm for MFPs.

THEOREM 2 (MAIN RESULT). *For any trajectory database S, number $r > 0$, and dimension $d \geq 1$, the algorithm* FPM-MP *shown in Algorithm 3 finds all maximal-duration r-flock patterns with length at least k_{min} in S without duplicates in $O(dkn) = O(dnT)$ amortized time per pattern and $O(dm^2)$ additional space, where k and m are the length and subset size of a discovered maximal-duration pattern P.*

PROOF. The correctness of the algorithm immediately follows from Lemma 9 and Lemma 11. The analyses of time and space complexities can be done in a similar manner as in Theorem 2 except that it now contains no anti-monotone pruning , and Lemma 11 makes it possible to directly generate maximal-duration patterns from its parent maximal-duration pattern. The work at each generated child with length k can be done in $O(dk)$ time and $O(k)$ space by using incremental computation similar to one in Lemma 1. This is repeated for all $O(n)$ possible children. By charging the cost to all of $O(n)$ children to each solution MFP, the cost for the solution becomes $O(dkn)$. Hence, the result follows. □

6. SPEED-UP USING SPATIAL INDICES

In this section, we present an modified algorithm FPM-MG with a speed-up technique using a spatial index obtained from the basic FPM-M.

The basic algorithm FPM-M works as follows. After selecting initial object ID $i_0 \in ID = \mathbb{O}$ and a start time $t \in \mathbb{T}$, it invokes the subprocedure RecFPM-M with a singleton pattern $P_0 = (\{i_0\}, [t, *])$ to start DFS. At each iteration with pattern P, the subprocedure first generates the intermediate pattern, denoted by $P \cup \{i\}$, obtained from P with adding any object $i \in ID_1$ to P such that (Condition 1) $i > \max(P.set)$ and (Condition 2) $radius(\mathsf{RClosure}(P \cup \{i\})) \leq r$. We note that the latter condition implies $\delta(pos(o, t), pos(c, t)) \leq r$ for any $o \in Q.set$, where $pos(o, t) = S[o][t] \in D$ denotes the location of object o at start time t. The left-expansion test $radius(S[X][Q.start - 1]) > r$ should be also tested, but this is not relevant with discussion here.

A *spatial index* such as range tree [5] compactly stores a set $U \subseteq D$ of points supporting the range query $U.Range(R_{c,r}) \triangleq U \cap R$ in sub-linear time, where $R_{c,r}$ is the axis-parallel box with center c and radius r. For instance, the d-dimensinal range tree structure stores n points in $O(dn \log^{d-1} n)$ space and $O(dn \log^d n)$ preprocessing supporting $O(\log^d n)$ time range query operation [5].

Using spatial indices, we modify the original algorithm FPM-M to obtain the revised version FPM-MG as follows. In preprocessing, for each start time $t \in \mathbb{T}$, the index U_t stores the locations $p_o = pos(o, t)$ with the pointer to o for all objects o in $\mathbb{O}$. In runtime, after selecting start time $t \in \mathbb{T}$ and initial object ID $i_0 \in \mathbb{O}$, the top-level procedure FPM-MG computes the subset

$$ID_1^{c,r} \leftarrow \{\, i \in U.Range(R_{c,r}) \mid i > i_0 \,\} \tag{6}$$

of ID, where $c = pos(i_0, t)$ is the location of i_0 at t. Then, we call the recursive procedure RecFPM-M with this subset as argument. Actually, we can easily see that any object ID $i \in ID_1^{c,r}$ satisfies Conditions 1 and 2 above. From above discussion, we see that the modified algoirthm FPM-MG correctly finds all r-MFPs in a given database S without duplicates. By simple average case analysis on uniformly random trajectories, we have the following result.

THEOREM 3. (AVERAGE-CASE ANALYSIS WITH SPATIAL INDEX) *Suppose that the database S consists of n random trajectories of length T each, where all locations are uniformly distribted over some area $A = [0, u]^2$ of size $u > 0$ with average density $\rho = nT/u^2$. Then, the modified algorithm* RecFPM-MG *finds all r-MFPs in S in $O(dk\rho r^2)$ delay using $O(dn \log^{d-1} n)$ additional space and $O(N \log^d n)$ preprocessing, where $d \geq 2$ is the dimension of the space, and $N = dnT$ is the total input size.*

The above theorem says that the delay of FPM-MG becomes $O(dk)$ time when the density ρ and radius r are constant. This is much smaller than the delay $O(dkn)$ of the improved algorithm FPM-MP for a large value of $n = |\mathbb{O}|$. Since dk is the size of the difference between consecutive answers, the delay seems to be nearly optimal assuming the use of the family tree $\mathcal{F}_r^{\mathrm{rm}}$.

7. EXPERIMENTAL RESULTS

To have insight into practical performance of the proposed algorithms in the previous section, we ran preliminary experiments on synthesis trajectory datasets.

7.1 Method

In the experiments, the following programs are implemented in C++, compiled by GNU g++ ver.4.6.3, and used in experiments, where FP, RFP, and MFP denote a flock, rightward maximal duration, and maximal duration flock patterns, respectively. The implementations FPM-R-G and BFE used a spatial/geometric index implemented in C++ inside.

- BFE: A previous algorithm for r-FPs based on breadth-first search for sequences of r-disks (Vieira *et al.* [14]).
- FPM-E: the naive depth-first search (DFS) algorithm for r-FPs using exhaustive search over FPs with length $k, k+1, \ldots, T$.
- FPM-M: the proposed basic DFS algorithm for r-MFPs (Sec. 2).
- FPM-MP: the proposed improved DFS algorithm for r-MFPs (Sec. 3).
- FPM-MG: the proposed improved version for r-MFPs with speed-up technique using geometric index (Sec. 6).

From Theorem 1 and Theorem 2, it is theoretically ensured that our algorithms FPM-M, FPM-MP, and FPM-MG output all patterns without duplicates. To verify this fact, we confirm that these programs correctly outputs more patterns than implanted. On the contrary, BFE [14] has a possibility that it may output the same flock patterns more than once if we do not use tabulation. [3]

As an experimental environment, for all experiments except Fig. 6, we used a PC with Intel Core i5, 1.7GHz, Memory 4GB running Mac OS X, ver.10.9.2. For Fig. 6, only this experiment was run on a PC server (CPU Intel Xeon 3.6GHz, 32GB, Debian GNU/Linux v.7.4) since larger memory was required for BFE.

7.2 Data sets

For experiments on synthesis datasets, we implemented a simple random trajectory generator, which implants random patterns into a specified fraction of randomly generated trajectories computed by random walk on the 2-d plane of specified domain size $a \times a$. Each data set contains n random walks of length T in the $a \times a$ plain in which f perturbed copies of h true patterns with length k_* and radius r_* are implanted. We show the default values in Table 1.

7.3 Results

[3]Precisely speaking, as Theorem 1 of [14] said, BFE correctly enumerates flock patterns in the form of k-vectors of radius-r disks as *syntex*, but some of them can have identical set of trajectories contained as *denotation*. Actually, their algorithm seemed to find a certain kind of representatives, called *closed patterns*, for flock patterns in a database. For detailed discussions on the potential problems in enumerating such *closed* patterns, see, e.g. [13, 17]

Table 1: Experimental parameters and their default values

category	parameter	symbol	default value
Trajectory	domain size	a	100.0
	number	n	100
	length	T	100
	num points	$N = nT$	10,000
True pattern	number	h	6
	length	k_*	50
	radius	r_*	1.0
	subset size	m_*	5
Mining pattern	min. len	k	40
	radius	r	1.0
	subset size	m	5

From Fig. 3 to Fig. 6, we show the results. In all figures, we observed that the proposed FPM-M (FPM Max) family of algorithms based on closure operators are one or two order of magnitudes faster than the naive algorithm FPM-E using exhaustive search. Precisely, the basic algorithm FPM-M is ten times faster, and the improved algorithms FPM-MP with $\mathcal{MSD}$ computation and FPM-MG with geometric index are hundreds times faster than the baseline method FPM-E.

In Fig. 6, we show comparison of FPM family and the previous algorithm BFE, where we skip $m = 14$ due to time out. From this plot, we see that our algorithm outperforms BFE in the speed and scalability to the input size. In summary, for most parameter values examined in this experiments, FPM-M family algorithms demonstrated stable performance as expected from theoretical analysis.

8. CONCLUSION

In this paper, we studied the enumeration version of maximal-duration flock pattern mining problem, which asks to systematically list all solution patterns without duplicates. Then, we presented two polynomial delay and space algorithms for maximal-duration r-flock patterns (r-MFPs) based on two characterizations. Overall, the proposed algorithms seem theoretically as well as practically efficient solutions for mining r-MFPs from large trajectories. For future work, we plan evaluation of the proposed algorithms on real data sets in meteorology and urban traffic.

Acknowledgements.

The authors would like thank anonymous reviewers for their comments which improved the correctness and the presentation of this paper very much. They would also like to thank Shin-ichi Minato, Shin-ichi Nakano, Kunihiko Sadakane, and Tetsuji Kuboyama for their comments on this work. This research was partly supported by MEXT Grant-in-Aid for Scientific Research (A), 24240021, FY2012–2015.

9. REFERENCES

[1] H. Arimura and T. Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In *Proc. SDM 2009, SIAM*, pages 1087–1098, 2009.

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

[3] M. Benkert, J. Gudmundsson, F. Hubner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41:111–125, 2008. Also appeared in Proc. ESA 2006.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

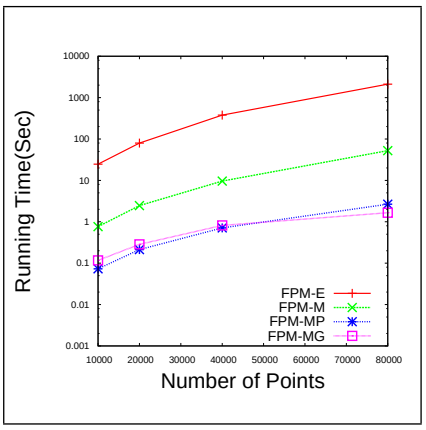


Figure 3: Exp 1: The running time of the algorithms varying the number of input points $N = nT$ on a PC

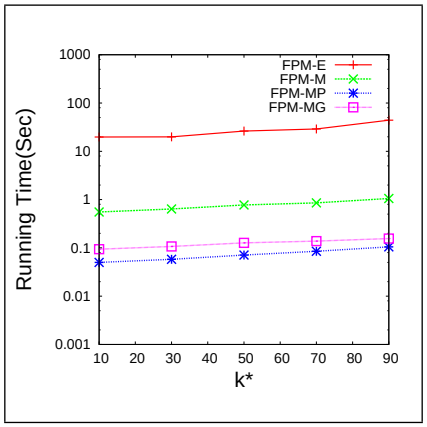


Figure 4: Exp 2: Running time of the algorithms varying true pattern length k_* and with fixed $k = 5$ on a PC

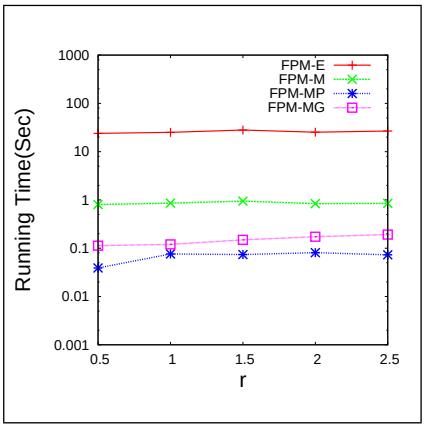


Figure 5: Exp 3: Running time varying the true and mining pattern radius r_* and r with $r = r_*$ on a PC

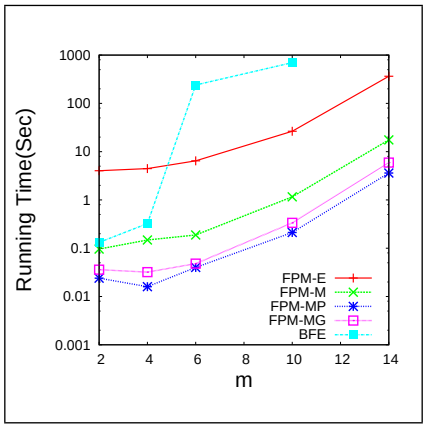


Figure 6: Exp 4: Running time varying the true and mining minimum subset size $m_* = m$ on a PC server.

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

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

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

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

[9] 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.

[10] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

[11] P. Revesz. *Moving Objects Databases, Introduction to Databases: From Biological to Spatio-Temporal, Chapter 7, Texts in Computer Science, vol.111*. Springer, 2010.

[12] U. Turdukulova, A. O. C. Romerob, O. Huismanc, and V. Retsiosa. Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach. *International Journal of Geographical Information Science*, 28(10):2013–2029, 2014.

[13] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Proc. DS'04, LNCS 3245*, pages 16–31, 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. Yang and M. Hu. Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. In *Proc. EDBT'06, LNCS 3896*, pages 664–681, 2006.

[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 TKDE*, 17(4):462–478, 2005.