Finding All Maximal Duration Flock Patterns in High-dimensional Trajectories

Hiroki Arimura¹, Takuya Takagi¹, Xiaoliang Geng¹, and Takeaki Uno²

¹ Hokkaido University, N14 W9, Sapporo 060-0814, Japan
{arim,tkg,gengxiaoliang}@ist.hokudai.ac.jp
² National Institute of Informatics, 2-1-2 Hitotsubashi, Tokyo 101-8430, Japan
uno@nii.jp

Abstract. In this paper, we study the problem of finding maximal duration flock patterns from trajectory data in dimension \(d\) = 2 or more, which is a class of spatio-temporal closed patterns introduced by (Gudmundsson and van Kreveld, 2006). The problem of the previous approaches to was the exponential dependency of the running time on the duration length \(k\) and the dimension \(d\). Also, they may miss some portion of patterns. For the problem, we present polynomial delay and space mining algorithms that finds all maximal duration flock patterns with specified radius and more than one entities appearing in an input collection of \(n\) trajectories in \(O(dkn)\) time per pattern using \(O(dm^2)\) extra space based on depth-first search, where \(m\) and \(k\) are the subset size and the length of a pattern to find, respectively. To the best of our knowledge, this is the first result on a polynomial delay and space algorithm for complete mining of all maximal patterns in the class for every \(d \geq 2\). We also describe a speed-up technique using spatial-index. Finally, we present some experimental results, where the improved version of proposed algorithms outperform the previous algorithm BFE (Vieira et al, 2009).

1 Introduction

Backgrounds. Recent advances of location tracking technologies, such as GPS and WIFI-positionning, produces a huge amount of trajectory data in the analysis of, e.g., car traffic, pedestrian data, wild animals, and sports videos. From collections of trajectories, there are demands for efficiently extracting interesting collective behavior or patterns, called trajectory mining. However, since such trajectory patterns are typically continuous objects in a higher-order spatio-temporal domain, not easy to handle, it is a challenging goal to establish methodologies to efficient handle pattern mining in trajectories.

A flock pattern is a kind of spatio-temporal patterns, introduced by Benkert et al. [4] as well as Laube et al. [9], which represents a set of entities moving together in a high-dimensional space close each other during a specified time span, such as a group of probe cars on the earth or a set of behaviors of correlated neurons. In this paper, we consider the problem of mining the complete set of flock patterns in a set of trajectories. A flock pattern with a set of entities is said to have maximal duration if its duration is maximally possible retaining the locations of entities within a specified radius (Gudmundsson and van Kreveld [8]).
Hiroki Arimura, Takuya Takagi, Xiaoliang Geng, and Takeaki Uno

Fig. 1. An example of a trajectory database $S_1 \subseteq \mathbb{D}^T$ on object ID set $ID$, time domain $T$ in the two dimensional space $\mathbb{D} = \mathbb{R}^2$, where $T = 7$ and each point $p$ and the associated number $t$ on a trajectory indicate the location and time stamp of a mobile object. We also show a flock pattern $P_1$ consisting of an ID subset $X = \{1, \ldots, 5\} \subseteq ID$ and a duration $I = [3, 5] \subseteq T$.

**Problems with previous approaches.** Unfortunately, the time complexities of previous algorithms depends exponentially in the length $k \geq 1$ of discovered patterns and the dimension $d \geq 1$ of the space. Thus, it cannot be applied to maximal duration pattern problem. In addition, they can output only a part of flock patterns, not the complete set of them. These can be limitaiton in applying flock pattern mining to real trajectory data.

**Goal of this paper.** In this paper, we consider the problem of complete mining of all $(\ast, \text{max } k, r)$-flock patterns in a set of trajectories in arbitrary dimension $d \geq 1$. We relax the problem by allowing an algorithm to find flock patterns with any maximal duration, not maximum, and with any number of entities. Furthermore, we study the problem of finding all maximal duration flock patterns, not some of them, in the framework of output-sensitive complexity, suitable to analysis of data mining algorithms, such that the efficiency of an enumeration algorithm is measured in the running time per solution, or delay, not the total running time.

As a main result, we present a depth-first mining algorithm that finds all maximal duration flock patterns with specified radius and more than one entities appearing in an input collection of trajectories in $O(dkn)$ time per pattern using $O(N + dm^2)$ space based on depth-first search, where $n$ is the number of entities, $N$ is the total length of input trajectories, $m$ and $k$ are the size and the length of a pattern to find, respectively. To the best of our knowledge, this is the first result on a polynomial delay and space algorithm for complete mining of all maximal duration flock patterns for every $d \geq 2$.

To achieve efficient mining, the algorithms adopt depth-first search in the pattern growth framework by constructing tree-shaped search route over all maximal
duration flock patterns. We also present a speed-up technique using a spatial index in experimental section.

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. [13]) 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.

This paper is organized as follows. Sec. 2 introduce some definitions and notations on maximal duration flock pattern mining. Sec. 3 presents a basic polynomial algorithm for finding all maximal duration flock patterns. Then, Sec. 4 presents an improved algorithm for the problem. In Sec. 5 shows experimental results. Sec. 6 concludes this paper.

2 Preliminaries

In this section, we prepare some definitions and notations that are used in this paper. The definitions not defined here will be found in textbooks [5,7,11].

2.1 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 $\{x \in \mathbb{R} \mid a \leq x \leq b \}$, 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 $\varepsilon$ 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 $|S| = n$.

2.2 Metric Space

We introduce basic notions in computational geometry [7]. A metric space is a pair $\mathbb{D} = (D, \delta)$ of a set $D$ and a function $\delta : D \to [0, \infty)$ satisfying the following conditions: for any points $p, q, r$ in $D$, (i) $\delta(p, q) = 0$ iff $p = q$, (ii) $\delta(p, q) = \delta(q, p)$, and (iii) $\delta(p, q) + \delta(q, r) \geq \delta(p, r)$. The set $D$ and the function $\delta$ are called a spatial domain and a metric (or distance function).

For every positive integers $d$ and $\lambda \geq 1$, we introduce a family $\mathbb{D}_{d, \lambda} = (D_d, \delta_\lambda)$ of metric spaces defined on the $d$-dimensional continuous space $D_d = \mathbb{R}^d$ and a metric $\delta_\lambda$ based on the $L_\lambda$-norm defined as follows. For any integer $\lambda = 1, 2, \ldots, \infty$ and any point $p = (p_1, \ldots, p_k)$ in $D$, we define the $L_\lambda$-norm of $p$ by the nonnegative number

$$L_\lambda(p) = \left( \sum_{i=1}^{k} |p_i|^{\lambda} \right)^{1/\lambda}$$

for every $\lambda = 1, 2, \ldots$, and $L_\infty(p) = \max_{i=1}^{k} |p_i|$ for $\lambda = \infty$. Then, the associated metric $\delta$ is defined by $\delta(p, q) \triangleq L_\lambda(p - q) \geq 0$ for any points $p, q$ in $D$ [7]. If it is
clear from context, we will omit the subscripts $d$ and $\lambda$, and write, e.g. $p \in D$ or $\delta(p, q)$. The ball with radius $r > 0$ at center $c \in D$ is the closed region $Ball_r(c) \triangleq \{ p \in D | \delta(c, p) \leq r \}$.

Among many metric spaces, we focus on the metric spaces $\mathbb{D}_{d, \lambda} = (D_d, \delta_\lambda)$ for every $d \geq 1$ and $\lambda = 2, \infty$. Let $S \subseteq D_d$ be a finite point subset in $\mathbb{D}_{d, \lambda}$. Then, the radius of $S$ is defined as $r(S) \triangleq \min_{c \in D} \max_{p \in S} \delta(c, p) \geq 0$, and the diameter (or simply the width) of $S$ is defined to be $\text{width}(S) \triangleq 2 \cdot r(S) \geq 0$. Intuitively, the minimum bounding ball of $S$, denoted by $\text{MBB}_{d, \lambda}(S)$, is the ball $\text{MBB}_{d, \lambda}(S) \triangleq Ball_{\hat{r}}(c)$ containing all points of $S$ having minimum radius $\hat{r} = r(S)$. In what follows, we concentrate on the case of $\mathbb{D}_{d, \infty} = (\mathbb{R}^d, \delta_\infty)$, i.e., $d$-dimensional continuous space with $L_\infty$-norm for the ease of computation.

**Lemma 1.** In the space $\mathbb{D}_{d, \infty}$, the MBB of a set $S$ of $n$ points is given by

$$\text{MBB}_{d, \lambda}(S) = \prod_{1 \leq i \leq d} [a_i, b_i], \quad (1)$$

where $a_i = \min_{p \in S} p_i$ and $b_i = \max_{p \in S} p_i$ are the minimum and maximum $i$-th coordinates in $S$. Furthermore, it can be computed in $O(dn)$ time from $S$.

In the following sections, the MBB will be used to efficient incremental computation of geometric objects. As a note, we can drop the $O(d)$ term in our results if the dimension $d$ is fixed.

### 2.3 Maximal trajectory motif discovery problem

In what follows, we consider only the metric space $\mathbb{D}_{d, \infty} = (\mathbb{R}^d, \delta_\lambda)$ associated with $L_\infty$-norm. However, we can extend our theory for any metric $\mathbb{D}$ if it has the unique and linear time computable MBB for any finite point set.

Let $ID = \{1, \ldots, n\}$, $n \geq 1$, be a set of moving objects (objects or object IDs), $T = [1, T]$ be an interval called the time domain, and $D = D_d$ be a metric space called the space domain. A trajectory with length $T$ is a sequence $s = s[1] \cdots s[T] \in D^T$ of $T$ points in $D_d$. The trajectory can be regarded as a function $s : \text{dom}(s) \to D$ from times to locations, where $\text{dom}(s) = T = [1, T]$ is the domain of $s$. A trajectory database (or database) is a collection

$$S = \{s_1, \ldots, s_h\} \subseteq D^T, \quad h \geq 1 \quad (2)$$

of trajectories, where for every $i = 1, \ldots, h$, $s_i = s_i[1] \cdots s_i[T] \in D^T$ is a trajectory of length $T$.

**Definition 1 (flock pattern).** A flock pattern (FP) is $S$ is a pair $P = (X, I)$ of a subset $X \subseteq ID$ of moving objects and a time interval $I = [b, e] \subseteq T$.

---

3 For $L_2$-norm, Megiddo [10] showed that the minimum enclosing ball $\text{MBB}_{d,2}(S)$ in any fixed dimension $d \geq 1$ can be computed in linear time by solving linear programming in fixed dimension in linear time.
As notation, for a flock pattern \( P = (X, I = [b, e]) \), we then denote by \( P.set = X \), \( P.span = I \), \( P.start = b \), and \( P.end = e \) and call them the subset, duration, start time, and end time of \( P \), respectively. The subset size of \( P \) is defined by \(|P.set| \geq 0\) and its duration length is defined by \( \text{len}(P) = |I| = e - b + 1\).

The semantics of a flock pattern is defined through its width in a database \( S \) as follows. We need some technical notions. For a subset \( X \subseteq ID \) of moving objects and time \( t \in T \), the cross section of \( S \) w.r.t. \( X \) at time \( t \) is given by

\[
S[X][t] \triangleq \{ s_i[t] \in D \mid i \in X \}.
\]

Then, the width of pattern \( P \) at time \( t \) in \( S \) is defined to be \( \text{width}(S[X][t]) \geq 0 \). A radius parameter is any positive number \( r > 0 \).

**Definition 2 (r-flock pattern).** A flock pattern \( P \) is an width-\( r \)-flock pattern (or \( r \)-flock pattern) in \( S \) if \( \text{width}(S[X][t]) \leq r \) for every \( t \in P.span \).

We denote by an \((m, k, r)\)-flock pattern any flock pattern in \( S \) with subset size at least \( m \), duration length at least \( k \), and width at most \( r \). We denote the class of such patterns by \( \mathcal{FP}(m, k, r) \). We sometimes use the symbol \( * \) at some parameter, say \( k \), to denote the corresponding parameter \( k \) can have arbitrary value.

**Definition 3.** An \( r \)-flock pattern \( P \) in \( S \) is said to be a maximal duration \( r \)-flock pattern (MFP) in \( S \) if there is no \( r \)-flock pattern \( Q \) in \( S \) such that (i) \( Q.set = P.set \), and (ii) \( Q.span \supset P.span \), i.e., the duration \( Q.span \) properly contains the duration \( P.span \).

Next, we introduce a one-sided version of maximal duration \( r \)-flock patterns, RFPs, below as a theoretical tool for analyzing the enumeration of MFPs in Sec. 3.

**Definition 4.** An \( r \)-flock pattern \( P \) in \( S \) is said to be of rightward maximal duration (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., \( Q \supset P \).

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

Now, we state our data mining problems for \( \mathcal{MFP}_r \) as follows.

**Definition 5.** The maximal duration flock pattern discovery problem: Given a positive number \( r > 0 \) and a set \( S \in D^T \) of \( n \) input trajectories with length \( T \), the task is to find all maximal duration \( r \)-flock patterns appearing in \( S \) without duplicates within class \( \mathcal{MFP}_r \).

Similarly, we can define the pattern discovery problems for \( \mathcal{RFP}_r \) and \( \mathcal{FP}_r \).
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 $\text{RClosure}(P = (X, [t_0, e_0]), S, r)$
2: $t \leftarrow t_0$
3: while $\text{width}(S[X][t]) \leq r$ do $t \leftarrow t + 1$
4: $b \leftarrow t_0$; $e \leftarrow t - 1$
5: return $P_{\text{max}}(X, [b, e])$

2.4 Model of computation

The goal of this paper is to devise efficient pattern mining algorithms for the above mining problems with high-throughput and of small memory footprint. To formalize this notion of efficiency, we employ the output-sensitive complexity in the theory of enumeration algorithms [3], becoming popular in the last decades [1, 2].

An enumeration algorithm $\mathcal{A}$ receives an input $I$ with size $N$, and outputs all of $M$ solutions, namely, all maximal duration flock patterns in our case, without duplicates [3, 12]. $\mathcal{A}$ is said to be of output-polynomial time if the total running time is bounded by some polynomial in the number $M$ of solutions, i.e., $t_{\text{total}} = O(poly(M, N))$. $\mathcal{A}$ is of polynomial time enumeration if the amortized running time per solution, namely $t_{\text{total}}/M$, is bounded by some polynomial in $N$, i.e., $t_{\text{total}} = O(M \cdot poly(N))$. $\mathcal{A}$ is of polynomial delay if the delay, i.e. the maximum computation time $t_{\text{delay}}$ between any consecutive outputs, is bounded by some polynomial in $N$ alone, i.e., $t_{\text{delay}} = poly(N)$.

Among all definitions above, polynomial delay is most strict, then (amortized) polynomial time enumeration follows, and finally output-polynomial time is most relaxed. As a computation model, we use the standard RAM [6].

3 Basic Mining Algorithm

In this subsection, we present a basic mining algorithm, called FPM-M (Maximum Duration Flock Pattern Miner with rejection), for the class $\mathcal{MFP}(*, r, k)$ of maximal duration flock patterns trajectory database. In Algorithm 2, we show the algorithm FPM-M and subprocedure $\text{RecFPM-M}$.

3.1 The first characterization of maximal duration flock patterns

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 6 (rightward closure). For any number $r > 0$, the $r$-rightward closure of a $r$-flock pattern $P$ is the unique flock pattern $\text{RClosure}(P, S, r) = Q$ satisfying the following conditions (1)–(3):

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

Algorithm 1 computes $Q = \text{RClosure}(P, S, r)$ in $O(dm) = O(dmT)$ time, where $m = |P.set|$ and $k = \text{len}(P)$. The next lemma tells us how we can accelerate this computation regardless of $m$ by using the idea of MBB (minimum bounding box) of Lemma 1 incrementally.

**Lemma 2** (fast computation of $\text{RClosure}$). Starting from a singleton pattern $P_0$, we can iteratively compute $P_i$ as $P_i = \text{RClosure}(P_{i-1} \cup \{i\}, S, r)$ in $O(dk)$ time from $P_{i-1}$ and ID $i$.

**Proof.** Let $k = \text{len}(P.span)$. We maintain the $k$-vector of MBBs defined by $\text{envelope}_S(P) = (\text{MBB}(S[P.set][t]))_{t \in P.span}$. When we add a new trajectory to $P.set$, we can update all MBBs in $O(k)$ time in total, and thus can compute the width of $P$ in the same time complexity.

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

**Lemma 3** (characterization of $\mathcal{RM}_r$). An $r$-flock pattern $P$ is said to be a rightward maximal duration $r$-flock pattern in $S$ if $P = \text{RClosure}(P, S, r)$.

**Lemma 4** (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 of rightward maximal duration in $S$.
Algorithm 2 The rejection-based polynomial amortized delay and space algorithm for finding all maximal duration \((\ast, \max k, r)\)-flock patterns in a trajectory database \(S\) in dimension \(d \geq 1\), where \(r\) is a radius parameter and the operator \(\text{delete}\min(X)\) deletes and then returns \(\min(X)\) from \(X\).

1: procedure FPM-M\((ID = \{1, \ldots, n\}, S, r, k)\)
2: \hspace{1em} for \(t \leftarrow 1, \ldots, T\) do \(\triangleright\) each start time in \(\mathbb{T}\)
3: \hspace{2em} while \(ID \neq \emptyset\) do \(\triangleright\) each id in \(ID\)
4: \hspace{3em} \(i \leftarrow \text{delete}\min(ID);\)
5: \hspace{3em} \(P \leftarrow \{i, [t, \ast]\};\) \(\triangleright\) initial maximal pattern
6: \hspace{3em} RecFPM-M\((P, ID, S, r, k)\);
7: procedure RecFPM-M\((P, ID, S, r, k)\)
8: \hspace{1em} \(P \leftarrow \text{RClosure}(P, S, r);\)
9: \hspace{2em} if \(|P_*| < k\) then return \(\triangleright\) backtrack
10: \hspace{2em} if \(\text{width}(S[X][t]) > r\) then \(\triangleright\) maximality test
11: \hspace{3em} output \(P_*;\)
12: \hspace{2em} \(ID_1 \leftarrow ID;\)
13: \hspace{2em} while \(ID_1 \neq \emptyset\) do
14: \hspace{3em} \hspace{1em} \(i \leftarrow \text{delete}\min(ID_1);\)
15: \hspace{3em} \hspace{1em} Create \(Q; Q.set \leftarrow P_*\cdot set \cup \{i\};\)
16: \hspace{3em} \hspace{1em} \(Q\cdot start \leftarrow P_*\cdot start; Q\cdot end \leftarrow P_*\cdot end;\)
17: \hspace{3em} \hspace{1em} RecFPM-M\((Q, ID_1, S, r, k);\) \(\triangleright\) recursive call

2. \(\text{width}(S[P.set][P.start - 1]) > 2r\). (leftward extension test \((\ast)\))

Proof. For any time \(t\), let \(w_t\) be the width 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 \(2r\) within the duration \(I = [P\cdot start, P\cdot end]\), (ii) \(w_t > 2r\) at \(t = P\cdot start - 1\), and (iii) \(w_t > 2r\) at \(t = P\cdot end + 1\). 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. \(\square\)

3.2 The family forest for \(\mathcal{MFP}_r\)

From the above Lemma 4, in order to find all maximal duration patterns, we see that it is sufficient to (i) first enumerate each rightward maximal duration \(r\)-flock patterns \(P\) in a trajectory database \(S\), (ii) test if \(P\) can be extended leftward by the test \((\ast)\), and (iii) output \(P\) if the test succeeds and reject it otherwise.

The remaining problem is how to enumerate all rightward maximal duration \(r\)-flock patterns without duplicates. We associate the parent \(P(Q)\) to any rightward maximal duration \(r\)-flock pattern \(Q\) in \(S\) as follows.

Definition 7 (parent for \(\mathcal{RFP}_r\)). Let \(Q\) be any rightward maximal duration flock pattern such that \(|Q\cdot set| \geq 2\). Then, we define the parent of \(Q\), denoted by \(P(Q)\), by \(P(Q) \overset{\Delta}{=} \text{RClosure}(R)\) such that \(R\cdot set = Q\cdot set - \{i_{\text{max}}\}\), i.e., \(P(Q)\) is the closure of the pattern \(R\) obtained from \(Q\) by removing the maximum ID \(i_{\text{max}} = \max(Q\cdot set)\) from \(Q\cdot set\). Then, \(Q\) is called a child of \(P\).
The next lemma immediately follows from the construction.

**Lemma 5.** For any rightward maximal duration $r$-flock pattern $Q$ such that $|Q.set| \geq 2$, (i) $P(Q)$ 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 a rightward maximal duration $r$-flock pattern in $S$, too.

Recall that $\mathcal{RFP}_r$ is the class of all rightward maximal duration $r$-flock patterns in $S$. Now, we introduce a directed graph $\mathcal{T}^{\text{rm}} = (V, P, I)$, called the family forest for $\mathcal{RFP}_r$, where:

- $V = \mathcal{RFP}_r$ is the set of all rightward maximal duration $r$-flock patterns as the vertex set.
- $P : \mathcal{RFP}_r \setminus I \rightarrow \mathcal{RFP}_r$ is the parent function representing the reverse edges $(P(Q), Q)$ for any $Q$ such that $|Q.set| \geq 2$.
- $I \subseteq V$ is the set of root nodes that are singleton patterns $P \in \mathcal{RFP}_r$ with $|P.set| = 1$.

**Lemma 6 (family forest for $\mathcal{RFP}_r$).** The family forest $\mathcal{T}^{\text{rm}}$ is a spanning forest over the class $\mathcal{RFP}_r$ of all rightward maximal duration $r$-flock patterns in $S$.

*Proof.* We will show that the directed graph $\mathcal{T}^{\text{rm}}$ is connected and acyclic. From condition (iii) of Lemma 5 and the termination property of sizes, we see that starting from any non-singleton maximal pattern $Q$, any ascending path $Q, P_1(Q), P_2(Q), \ldots$ eventually terminates at some singleton maximal pattern $P$ in $I$ in the graph $\mathcal{T}^{\text{rm}}$. Thus, $\mathcal{T}^{\text{rm}}$ is acyclic and contains every maximal pattern in $\mathcal{RFP}_r$ as its node. This completes the proof. \qed

### 3.3 A rejection-based polynomial delay algorithm for $\mathcal{MFP}_r$

The remaining thing is how to efficiently generate all children $Q$ of a given $P$. The following lemma answers this question.

**Lemma 7 (generation of a child).** For any rightward maximal duration $r$-flock patterns $P, Q \in \mathcal{RFP}_r$, (a) $P$ is the parent of $Q$ if and only if (b) $Q$ is obtained from $P$ by $Q = \text{RClosure}(P \cup \{i\}, S, r)$ for some ID $i \in ID$ such that $i > \max(P.set)$.

From the above discussion, we obtain Algorithm 2. In order to give the estimation of an upperbound of the amortized delay of Algorithm 2, we finally give the ratio of the number $|\mathcal{RFP}_r|$ of all rightward maximal duration patterns to the number $|\mathcal{MFP}_r|$ of all maximal duration patterns in $S$.

**Lemma 8.** For any trajectory database $S$ on time domain $\mathbb{T} = [1, T]$ in dimension $d \geq 1$, and radius parameter $r > 0$, we have $|\mathcal{RFP}_r| \leq |\mathcal{MFP}_r| \cdot T$.

We now have the following theorem.
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 $(*, \max k, r)$-flock patterns 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 6 and Lemma 7, we can easily see that the algorithm FPM-M visits all rightward maximal $r$-flock patterns in $S$ without duplicates at Line 8. From Lemma 4, 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 $(MBB(S[P.set][t]))_{t \in P.duration}$ of the MBBs of the sections of $P$. Since each rightward maximal pattern $P_*$ can have at most $n = |ID|$ children and all of them can be failed in the worst case, the worst case time per rightward maximal pattern, i.e. the delay, is bounded by $O(dkn)$ time. Thus, Lemma 8 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. □

4 Faster Algorithm

In this subsection, we present an improved mining algorithm, called FPM-MP (Maximum Duration Flock Pattern Miner with Partitioning), for the class $MF(*, r, k)$ of maximal duration flock patterns in trajectory database $S$.

4.1 A problem in 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 4 (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.

4.2 The second characterization by the left and rightward closure

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 8 (left and rightward closure). Let $P$ be a $r$-flock pattern. The $r$-left-rigth-closure of $P$ is the unique flock pattern $\text{LRClosure}(P, S, r) = Q$ satisfying
Algorithm 3 The partition-based polynomial delay and space algorithm for finding all maximal duration \((\ast, \max k, r)-flock\) patterns in a trajectory database \(S\) in dimension \(d \geq 1\), where \(r\) is a radius parameter and the operator \(\textit{deletemin}(X)\) deletes and then returns \(\min(X)\) from \(X\). For \(\textit{span} = [b, e]\), we denote by \(\textit{span.start} = b\) and \(\textit{span.end} = e\).

1: procedure \(\text{FPM-MP}(ID = \{1, \ldots, n\}, S, r, k)\)
2: while \(ID \neq \emptyset\) do \(\triangleright\) each id in \(ID\)
3: \(i \leftarrow \text{deletemin}(ID)\);
4: \(P \leftarrow (\{i\}, \text{dom}(s_i))\); \(\triangleright\) initial maximal pattern
5: \(\text{RecFPM-MP}(P, ID, S, r, k)\);

6: procedure \(\text{RecFPM-MP}(P, ID, S, r, k)\)
7: if \(\text{len}(P) < k\) then \text{return} \(; \) \(\triangleright\) backtrack
8: output \(P\);
9: \(ID_1 \leftarrow ID\);
10: while \(ID_1 \neq \emptyset\) do \(\triangleright\) first loop
11: \(i \leftarrow \text{deletemin}(ID_1)\);
12: Create \(Q\); \(Q.set \leftarrow P.set \cup \{i\}\);
13: \(Q.start \leftarrow P.start; Q.end \leftarrow P.end\);
14: \(\text{MSD} \leftarrow \text{MaxSubDuration}(Q, S, r)\); \(\triangleright\) all maximal sub-durations
15: for each \(\text{maxspan} \in \text{MSD}\) do \(\triangleright\) second loop
16: Create \(R\); \(R.set \leftarrow Q.set\);
17: \(R.start \leftarrow \text{maxspan.start}; R.end \leftarrow \text{maxspan.end}\);
18: \(\text{RecFPM-MP}(R, ID_1, S, r, k)\); \(\triangleright\) recursive call

\(1\) \(Q.set = P.set\), and \(2\) \(Q.start \leq P.start\) and \(Q.end \geq P.end\), respectively, are the smallest and largest time points satisfying that \(\text{width}(S[Q.set][t]) \leq 2r\) for every \(t \in [Q.start, Q.end]\).

Lemma 9. \(Q = \text{LR Closure}(P, S, r)\) always exists, is unique, and can be computed in \(O(mk^*\ast)\) time, where \(m = |P.set| = |Q.set|\) and \(k^* = |Q.span|\).

As in the previous section, we construct a spanning forest \(T^m\) over the class \(M_r\) of all maximal duration flock patterns. First, we define the parent function.

Definition 9. Let \(Q\) be any maximal duration flock pattern in \(S\). Then, we define the \textit{parent} of \(Q\), denoted \(P(Q)\), by \(P(Q) \triangleq \text{LR Closure}(Q - \{i_{\max}\})\), 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 \textit{family forest} for the class \(M_r\) as the directed graph \(T^m = (V, P, I)\), where \(V = M_r\) is the vertex set, \(P\) is the set of reverse edges from children to parents, and \(I\) is the set of roots. Similar to Lemma 6 in Sec. 3, we have the following lemma for the family forest.

Lemma 10 (family forest for \(M_r\)). The family forest \(F^m\) is a spanning forest over the class \(M_r\) of all maximal duration \(r\)-flock patterns in \(S\).
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** \texttt{MaxSubDuration}(Q, S, r)
2: \hspace{1em} MSD $\leftarrow \emptyset$;
3: \hspace{1em} $b \leftarrow Q.start$;
4: \hspace{1em} \textbf{while} $b \leq Q.end$ \textbf{do}
5: \hspace{2em} Increment $b$ while $width(S[X][b]) > r$; $\triangleright$ skip inter-duration time
6: \hspace{2em} $e \leftarrow b$;
7: \hspace{2em} Increment $e$ while $width(S[X][e]) \leq r$; $\triangleright$ expand a duration rightwards
8: \hspace{2em} MSD $\leftarrow$ MSD $\cup \{[b,e]\}$;
9: \hspace{1em} $b \leftarrow e + 1$;
10: **return** MSD;

To invert the parent-child relationship, we need some technical definitions below. We introduce the notion of MSD (the maximal $r$-sub-duration list) of a pattern $P$ as follows. Let $P$ be a flock pattern with arbitrary width. A sub-duration $I \subseteq P.span$ is said to have width $r$ if $width(S[P.set][t]) \leq r$ 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.

**Definition 10 (maximal sub-duration list).** The maximal $r$-sub-duration list for $P$, denoted by $\text{MSD}(P, S, r)$, is a sorted list $M = (I_1, \ldots, I_\ell)$, $I_1 < \cdots < I_\ell$, of non-empty sub-interval of $T = [1, T]$ in the preceding order such that

1. $\max_{t \in I_i} width(S[P.set][t]) \leq r$ for every $i = 1, \ldots, \ell$.
2. any interval of $\text{MSD}(P, S, r)$ cannot be extended in both direction without violating property (i) above.

4.3 A faster polynomial delay algorithm for $MFP_r$

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

**Lemma 11.** For any flock pattern $P$ with arbitrary width, $\text{MSD}(P, S, r)$ is unique, and computable in $O(mk)$ time, where $m$ and $k$ are the size of subset and the length of duration of $P$, respectively.

From the next lemma, we can efficiently computes any child $Q$ of a given parent maximal duration flock pattern $P$.

**Lemma 12 (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 $Q.span \in \text{MSD}(P, S, r)$. 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 \( I \). It then recursively expands the parent maximal duration flock pattern \( P \) by adding a new object ID \( i > \text{tail}(P) \) to obtain another maximal duration flock pattern \( Q \) with larger subset size as its child. This expansion is done by Lemma 12. From the discussion above, we have the main theorem of this paper.

**Theorem 2 (improved polynomial delay and space algorithm).** 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 \((\ast, \text{max } k, r)\)-flock patterns in \( S \) without duplicates in \( O(dnk) = O(dnT) \) 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 10 and Lemma 12. 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 12 makes it possible to directly generate maximal duration patterns from its parent maximal duration pattern. The work at the parent is \( O(k_\ast) \) incremental time if we economically maintain the list of MBBs of all sections of \( P \) in \( S \) in \( O(k_\ast) \) space. Hence, the result follows. \( \square \)

## 5 Experimental Results

We ran experiments on a set of synthesis datasets in \( d = 2 \) dimensional space to evaluate the proposed algorithms under various parameter settings.

### 5.1 Data

In Table 1, we show the experimental parameters and their default values, which are grouped into database, true patterns, and mining patterns. We used synthesis datasets randomly generated by implanted attern method as follows. Each data set is a collection of \( n \) random walks of length \( T \) in the \( a \times a \) plain in which \( f \) parturbed copies of \( h \) true patterns with length \( k_\ast \) and width \( r_\ast \) are implanted.

<table>
<thead>
<tr>
<th>category</th>
<th>parameter</th>
<th>symbol</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectory domain size</td>
<td>( a )</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td>( n )</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>( T )</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>num points</td>
<td>( N = nT )</td>
<td>10,000</td>
<td></td>
</tr>
<tr>
<td>True number</td>
<td>( h )</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>pattern length</td>
<td>( k_\ast)</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>( r_\ast)</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>subset size</td>
<td>( m_\ast)</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Mining min. len</td>
<td>( k )</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>( r )</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>subset size</td>
<td>( m )</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Fig. 3. Exp 1: The running time of the algorithms varying the number of input points $N = nT$. This experiment was run on PC with CPU Intel Core i5, 1.7GHz, 4GB.

Fig. 4. Exp 2: Running time of the algorithms varying true pattern length $k^*$ and with fixed $k = 5$. This experiment was run on PC with CPU Intel Core i5, 1.7GHz, 4GB.

In the following, we denote by FP, RFP, and MFP a flock, rightward maximal duration, and maximal duration patterns, respectively.

5.2 Method

In the experiments, we implemented the following programs in C++ compiled by GNU g++ ver.4.6.3, and used in experiments, where FPM-R-G and BFE used a spatial/geometric index implemented in C++ inside.

- **BFE**: a breadth-first algorithm for FP based on combination of $r$-disks (Vieira et al. [13]).
- **FPM-E**: a naive depth-first algorithm for FP using exhaustive search over FPs with length $k, k + 1, \ldots, T$.
- **FPM-M**: a basic depth-first algorithm for MFP (Sec. 2).
- **FPM-MP**: an improved depth-first algorithm for MFP (Sec. 3).
- **FPM-MG**: FPM-M for MFP augmented with the speed-up technique based on geometric index (Due to the space, we omit the detail).

Note that all three algorithms of FPM family output all patterns without duplicates, while BFE can output the same pattern more than once [13]. 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 experiments of Fig. 6, we used a PC with Intel Xeon E5-1620, 3.6GHz, Memory 32GB running Debian GNU/Linux, ver.7.4 since we needed CPU power and memory to run BFE. In the preliminary experiments, we confirm that all implementations correctly found more number of patterns than implanted. We stop an experiment
if the running time exceeds the predetermined time limit, approx. one hour for BFE and FPM-E.

5.3 Results
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 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.

6 Conclusion
In this paper, we studied the problem of complete mining the class \( \mathcal{MFP}_r \) of all maximal duration flock patterns in a given trajectory database. We present two polynomial delay and space algorithms for \( \mathcal{MFP}_r \) based on new characterization of the class. Overall, the proposed algorithms seem theoretically as well as practically efficient solutions for mining large trajectories.
References


