Studies on Depth-First Mining Algorithms for Maximal Flock Patterns from Large Trajectory Data

Geng Xiaoliang

September, 2015

Division of Computer Science

Graduate School of Information Science and Technology

Hokkaido University

Abstract

As the period of big data comes, researchers focus their eyes on novel types of data, such as streaming data and spatio-temporal data. GPS trajectory data as one of these kinds of novel data becomes more attractive than before. Since mining of trajectory data has different characteristics from traditional transaction data mining research, trajectory mining has attracted a great deal of attention for recent years. Flock pattern mining is one of the trajectory data mining topic introduced by Gudmundsson *et al.* in 2006. Flock patterns are a class of spatio-temporal patterns that represent a groups of moving objects which are close to each other in a given time segment. They are useful in detecting a group of highly correlated entities combining spatio-temporal features.

In flock pattern mining, however, a huge number of redundant patterns with shorter lengths and fewer occurrences can be generated, which may degenerate the efficiency and comprehensiveness. One of the possible approach to this problem is to find *only* a subset of reprsentative patterns rather than *all* patterns. To overcome these difficulties, in this thesis, we introduce the classes of *length-maximal* and *size-maximal flock patterns*. Then, we devise efficient mining algorithms that find these classes of flock patterns from large trajectory data. Particularly, we present a variety of *depth-first mining algorithms* based on *pattern-growth approach* [66] for finding maximal flock patterns, which are efficient in terms of both time and space.

In Chapter 3, we focus on mining of rightward length-maximal flock patterns. A

rightward length-maximal flock pattern (RFP, for short) is such a maixmal flock pattern that cannot be extended rightward further without changing their occurrences. After introducing a basic algorithm FPM (Flock Pattern Miner), we present a depth-first algorithm RFPM (Rightward-maximal Flock Pattern Miner) for finding all RFPM patterns with given radius, and then a faster algorithm G-RFPM that uses a geometric index for data points to speed-up the search for RFPs.

In Chapter 4, we extended the results in Chapter 3 for the class of two-sided length-maximal flock patterns. Introduction of RFPs eliminates some redundant patterns with the same start points, while it may still include other shorter patterns with the different start points. To exclude these redundancy, we introduce the class of twosided length-maximal flock patterns (MFP, for short), which is a maixmal flock pattern that cannot be extended either leftward or rightward. Consequently, we see that the number of MFPs is no more than the number of RFPs in any trajectory data. For mining MFPs, we first present the algorithm FPM-M (FPM for Maximal Patterns) based on rejection of redundant patterns. Next, we present faster algorithms FPM-MP (FPM-M with Partition), which is based on recursive partition of patterns, and FPM-MG (FPM-M with Geometric Index) with a spatial index. All algorithms find all length-maximal flock patterns with given radius appearing in trajectory data.

In Chapter 5, we study size-maximal flock pattern mining problem. We present an algorithm **DFM** (Depth-First Miner) for mining a subclass of size-maximal envelope patterns, called size-maximal and rightward length-maximal flock patterns (RSFPs, for short), under given constraints from an input trajectory database. In reality, the algorithm works on equivalent representation of size-maximal and rightward length-maximal flock patterns, called *rightward length-maximal envelope patterns* (REVPs). The algo-

rithm combines depth-first search, efficient closure computation, and hash-based duplication check to achieve complete mining of all RSFPs.

Finally, we conclude this thesis and discuss future researches in Chapter 6. In summary, we studied the problem of mining various notions of maximal flock patterns, and presented efficient depth-first mining algorithm that find all maximal patterns in an input trajectory database for these classes of patterns. In experiments on artificial and real trajectory data, our algorithms were correctly and efficiently find all flock patterns.

Acknowledgments

First and foremost I would like to thank Professor Hiroki Arimura who supervised me from the beginning of my research activity. I would like to thank Associate Professor Takuya Kida. He supervised me very kindly, too.

I am deeply grateful to Professor Makoto Haraguchi, Professor Thomas Zeugmann and Professor Shin-ichi Minato for his guidance and strong support.

I wish to thank to Professor Takeaki Uno and Professor Yuzuru Tanaka, for their comments for improving and completing the thesis.

I am deeply indebted to many people who helped and supported me. I particularly appreciate to Associate Professor Masaharu Yoshioka and Dr. Satoshi Yoshida for their helpful discussions.

Finally, I would like to thank my family. My parents support me selflessly. My wife looks after me carefully during these years. I also would like to thank all my dear friends. I spent very happy time with them.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Scope of this thesis 2

1.3 Contributions of this thesis 5

1.4 Related work 6

1.4.1 Traditional frequent itemset data mining researches 6

1.4.2 Flock pattern mining 8

1.4.3 Other researches 10

2 Preliminaries **13**

2.1 Basic definitions 13

2.2 Trajectory databases 14

2.2.1 Two-dimensional discrete trajectories 14

2.2.2 Trajectory databases 15

2.2.3 Geometric indices 17

2.3 Classes of flock patterns 17

2.3.1 The class FP of flock patterns 17

2.3.2 The class RFP of rightward length-maximal flock patterns 19

2.3.3 The class MFP of two-sided length-maximal flock patterns 20

2.3.4 The class RSFP of rightward length-maximal, size-maximal flock patterns . . . 21

2.4 Equivalence between the class of envelope patterns and the class of flock patterns . . . 24

2.5 Flock pattern mining problems . . . 26

3 Depth-First Mining Algorithms for Rightward Length-Maximal Flock Patterns 29

3.1 A basic DFS algorithm **FPM** for FPs . . . 29

3.1.1 The algorithm . . . 29

3.1.2 Analysis of the algorithm . . . 30

3.2 A modified algorithm **RFPM** for RFPs . . . 31

3.2.1 Basic idea: Rightward horizontal closure . . . 32

3.2.2 The algorithm . . . 33

3.3 A faster algorithm **G-RFPM** with using geometric index . . . 34

3.3.1 The algorithm . . . 35

3.4 Experiments . . . 36

3.4.1 Data . . . 36

3.4.2 Methods . . . 37

3.4.3 Results . . . 38

3.5 Discussion . . . 42

4 Depth-First Mining Algorithms for Two-Sides Length-Maximal Flock Patterns 55

4.1 A basic mining algorithm **FPM-M** for MFPs based on rejection . . . 55

4.1.1 The first characterization of MFPs 55
4.1.2 The family forest for MFPs 60
4.1.3 The algorithm 62
4.2 An improved algorithm **FPM-MP** for MFP based on partitioning 63
4.2.1 A problem with the previous algorithm 63
4.2.2 The second characterization of MFPs 64
4.2.3 The algorithm 65
4.3 Speeding-up by spatial index 67
4.4 Experiments 68
4.4.1 Data 68
4.4.2 Methods 68
4.4.3 Results 70
4.5 Discussion 71

5 Table-Based Depth-First Mining Algorithm for Size-Maximal Flock Patterns 77

5.1 An algorithm **DFM** for RSFPs using tabulation 77
5.1.1 The algorithm 77
5.1.2 Conversion of REVPs to RSFPs 80
5.2 Analysis 82
5.3 Experiments 82
5.3.1 Data 84
5.3.2 Methods 84
5.3.3 Results 84
5.4 Discussion 85

6 Conclusion **89**

6.1 Summary of the results 89

6.2 Discussion and future researches 92

Bibliography **93**

Chapter1

Introduction

1.1 Background

Data mining has become so important today. It is developed well in the last twenty years. As the period of big data comes, researchers focus their eyes on novel types of data, such as streaming data and temporal-spatial data.

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 [32, 50, 51, 95].

Trajectory data mining is an rapidly developing topic in the areas of data mining and query processing. It aims at discovering groups of trajectories based on their proximity in either a spatial or a spatio-temporal sense.

1.2 Scope of this thesis

Since it is difficult to use traditional methods to mine patterns from trajectory data, we have introduced the definition of **flock pattern** (FP, for short) [35] and study a mining problem for closed and constrained flock patterns. FP is composed of a pair of a trajectory set and a time interval. Using the definition of FP, there can be potentially a huge number of similar patterns. Particularly, the shorter patterns may be included in longer patterns. There is the similar problem for flock pattern given by [91]. To overcome this problem, we first introduce the definition of rightward length-maximal flock pattern (RFP, for short), and study a mining problem for constrained RFPs.

In Chapter 3, we focus on pattern-growth approach for the mining the class of rightward length-maximal flock patterns (RFP), where our purpose is to make complete mining of all patterns that satisfy a given constraint in an input database. For the purpose, we have been developing our algorithm FPM for complete mining of flock patterns [10], the first pure pattern-growth style algorithm. Particularly, this chapter focuses on two extensions of FPM, called RFPM and G-RFPM. The former RFPM finds a class of closed patterns, called *rightward length-maximal flock patterns*, while the latter G-RFPM uses *speed-up technique using a geometric index* [10]. We implemented the basic and the improved algorithms above based on pattern-growth mining approach. To evaluate these extensions, we then ran experiments on implanted synthesis datasets. The experiments demonstrate that both of extensions significantly improve on the efficiency of the original algorithm FPM in a wide range of parameter settings.

In Chapter 4, we consider the mining problem for mining the class of two side length maximal flock pattern (MFP, for short). However, RFP still may includes other shorter patterns. Because the shorter patterns may share the same end position in the right

of pattern, the start position on the left may follows the start position of longer RFP. To reduce these redundant shorter patterns, we extend the definition of RFP to present two-sided extension length-maximal flock pattern. In this chapter, we present a variety of depth-first mining algorithms for the two-sided extension length-maximal flock pattern enumeration problem. This flock pattern is two-sided extension longest pattern. Thus, shorter patterns could not be included in the longest patterns. As a result, the number of patterns is fewer.

We introduce the algorithm FPM-MP. It finds all two-sided extension length-maximal flock patterns in $O(dkn)$ time per pattern P using $O(dm^2)$ additional space.

We also show that a modified algorithm FPM-MG with practical speed-up technique using spatial indexes. 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 $d = O(1)$[1].

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 $d \geq 2$.

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. [91]) 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 per-

[1]This is done by using linear time minimum enclosing ball computation by Megiddo [59].

formance, and that the proposed algorithms were much faster than BFE and our basic versions. We also ran experiments on real typhoon trajectory data. **FPM-R-G** could find all RFPs from 320 typhoon trajectory data, We affirm that it is useful for real trajectory data.

In Chapter 5, we introduce the rightward size-maximal flock patterns from trajectory data and study a mining problem for the class rightward size-maximal and length-maximal flock patterns (RSFPs). We also introduce its equivalent representation, called rightward length-maximal envelope patterns (REVP).

An REVP is composed of a list of minimum bounding rectangles and a time interval. We could compute RSFPs from REVPs. And REVP is convenient to find size-maximal flock patterns. With the definition of REVP, there can be potentially infinitely many similar patterns for the continuous nature of space domain. To overcome this problem, we present a procedure that recovers a rightward longest, and most-specific EVP from the current candidate of its cover set. As a main result, we present an efficient algorithm DFM that, given maximum width θ, minimum length ℓ, and minimum frequency σ, finds all length-maximal REVPs with width $\leq \theta$, length $\geq \ell$, and frequency $\geq \sigma$ in a trajectory database. In addition, we present an algorithm that converts a discovered REVP into the equivalent RSFP. Consequently, the algorithm combines depth-first search, efficient closure computation, and hash-based duplication check to achieve complete mining of all RSFPs.

Finally, we conclude this thesis and discuss future researches in Chapter 6.

1.3 Contributions of this thesis

In this thesis, we aim at finding all length-maximal closed flock patterns. To solve this problem, we present and extend the definitions of patterns. And we introduce the basic algorithm and improve it.

Our contributions are the following:

General data mining

Trajectory data mining is different from traditional data mining. We introduce the definition of rightward size-maximal flock pattern for this novel data mining problem. But we find that there are many redundant patterns included in longer RSFP patterns. To solve this problem, we first present rightward length-maximal closed pattern. Then we extend the definition to two sides extension pattern, two-sided extension length-maximal flock pattern.

Mining closed pattern problem

We first design DFM algorithm. It could find all longest closed rightward size-maximal flock patterns without duplication patterns. But the efficiency is low. To improve this algorithm, we design FPM and RFPM algorithms for mining RFP pattern. Using geometric index, we present G-RFPM to speed up RFPM algorithm. Then we design FPM-M and FPM-MP algorithms for mining two-sided extension length-maximal flock patterns. We also use geometric index to design FPM-MG. The efficiency of FPM-MP and FPM-MG algorithms are both high.

Finally, we note that the contents of this thesis was partly presented in the conference papers [28–30] and a journal paper [31].

1.4 Related work

In this section, we first introduce researches of traditional frequent itemset data mining. When we study trajectory data mining, we adopt some thinking from frequent itemset data mining, such as the depth first approach, the pattern-growth approach and pruning for search space reduction and so on. Then we introduce studies about trajectory data mining. Finally, we introduce researches about flock pattern mining.

1.4.1 Traditional frequent itemset data mining researches

The Apriori algorithm for frequent itemset mining was presented by [6]. It is important for association rule mining which was first proposed by [3]. Mannila *et al.* [58] developed a variation of the algorithm using a similar pruning heuristic. Agrawal *et al.* [4] combined these works later. A method for generating association rules from frequent itemsets was described in [6].

There are variations of Apriori in the following. Park *et al.* [63] used hash tables to improve association mining efficiency. Savasere *et al.* [76] proposed the partitioning technique. Toivonen [88] discussed the sampling approach. A dynamic itemset counting approach was given in [18]. An efficient incremental updating of mined association rules was proposed by [26]. [5, 26, 64] studied parallel and distributed association data mining under the Apriori framework. Zaki *et al.* [98] proposed another parallel association mining method, which explores itemset clustering using a vertical database layout.

FP-growth [39], a pattern-growth approach for mining frequent itemsets without candidate generation, had been proposed as alternatives to the Apriori-based approach. Han *et al.* [67] proposed an exploration of hyper structure mining of frequent patterns, called H-Mine. A method that integrates top-down and bottom-up traversal of FP-trees in

pattern-growth mining was proposed [56]. Grahne and Zhu [34] proposed an array-based implementation of prefix-tree structure for efficient pattern growth mining. Eclat, an approach for mining frequent itemsets by exploring the vertical data format, was proposed by Zaki [99]. Agarwal *et al.* [1] proposed a depth-first generation of frequent itemsets by a tree projection technique. Sarawagi *et al.* [75] studied an integration of association mining with relational database systems.

Pasquier *et al.* [65] proposed the mining of frequent closed itemsets. They introduced an Apriori-based algorithm called A-Close for such mining. Pei *et al.* [68] proposed an efficient closed itemset mining algorithm, called CLOSET, based on the frequent pattern growth method. CHARM by Zaki and Hsiao [100] developed a compact vertical TID list structure called diffset. It records only the difference in the TID list of a candidate pattern from its prefix pattern. A fast hash-based approach was also used in CHARM to prune nonclosed patterns. CLOSET+ by Wang *et al.* [92] integrated previously proposed effective strategies as well as newly developed techniques such as hybrid tree-projection and item skipping. Liu *et al.* [55] introduced a method, called AFOPT, that explores a right push operation on FP-trees during the mining process. FPClose by Grahne and Zhu [34] was a prefix-treebased algorithm integrated with array representation, for mining closed itemsets using a pattern-growth approach.

Pan *et al.* [62] proposed CARPENTER, a method for finding closed patterns in long biological data sets, which integrates the advantages of vertical data formats and pattern growth methods. Bayardo [15] first studied mining max-patterns, where MaxMiner, an Apriori-based, level-wise, breadth-first search method, was proposed to find max-itemset by performing superset frequency pruning and subset infrequency pruning for search space reduction. MAFIA, another efficient method, developed by Burdick*et al.* [20], used

vertical bitmaps to compress TID lists, thus improving the counting efficiency. Goethals *et al.* [33] reported a FIMI (Frequent Itemset Mining Implementation) workshop dedicated to implementation methods for frequent itemset mining.

Mining interesting rules has been studied by many researchers. The interesting strong association rules was discussed in Chen *et al.* [23] ; Brin *et al.* [17]; and Aggarwal and Yu [2], which cover several interestingness measures, including lift. An efficient method for generalizing associations to correlations was given in Brin *et al.* [17]. Brin *et al.* [18] and Ahmed *et al.* [7] proposed other alternatives to the supportconfidence framework for assessing the interestingness of association rules.

Imielinski *et al.* [40] proposed a method for mining strong gradient relationships among itemsets. Silverstein *et al.* [77] studied the problem of mining causal structures over transaction databases. Some comparative studies of different interestingness measures were done by Hilderman and Hamilton [HH01]. Tan *et al.* [81] introduced the notion of null transaction invariance, together with a comparative analysis of interestingness measures. Omiecinski [61] and Lee *et al.* [52] studied the use of all confidence as a correlation measure for generating interesting association rules. Wu *et al.* [93] studied the Kulczynski measure for associative patterns and performed a comparative analysis of a set of measures for pattern evaluation.

1.4.2 Flock pattern mining

There are two lines of researches on trajectory mining: trajectory clustering [13, 51] and disk-based trajectory pattern mining [16, 32, 50]. The study of flock pattern mining started in the latter context [16, 36, 50]. Gudmundsson and van Kreveld [35] showed that the problem of finding at least one length-*maximum* (α, β, γ)-flock pattern is NP-

hard, while they gave an efficient 2-approximation algorithm, although it does not make complete enumeration of all flock patterns. Benkert *et al.* [16] proposed an $(2 + \varepsilon)$ approximation algorithm for fixed-length flock patterns, whose running time is polynomial in γ and $\frac{1}{\varepsilon}$, but exponential in the length β of a pattern, thus not polynomial delay. In addition, Cabrera *et al.* [72] analyzed the performance of flock pattern algorithms. Ulanbek *et al.* [89] developed an improved algorithm for mining such patterns following a frequent pattern discovery approach, a well-known task in traditional data mining.

Most closely related work is the work by Vieira, Bakakov, and Tsotras [91], who took pattern mining approach at the first time. They presented an efficient algorithm BFE with variations that finds all (α, β, γ)-flock patterns by systematically combining discovered clusters by depth-first search using the idea of intersection. This seems to be one of the first paper that take *pattern mining* approach for flock patterns. They also introduced the use of geometric constraints and spatial index for mining. However, it is not polynomial space algorithm since it uses tabulation to avoid duplicated patterns.

Recently, Romero [71] presented in his Master's thesis an interesting approach of solving a flock pattern mining by reduction to frequent pattern mining, where he used LCM algorithm [90] to solve the converted problem. However, its scalability seems to need improvement.

There have been a number of existing researches on finding flock patterns in a given trajectory database [16, 35, 50]. Unfortunately, however, most of these works deals with detection of one or more flock patterns as *pattern search* problem, not as *pattern mining* problem except few work (e.g., Vieira *et al.* [91]).

1.4.3 Other researches

Mining tasks on trajectories

The term trajectory pattern mining covers many different types of patterns. They can be mined from trajectory data [38]. Consequently, the concepts and techniques underlying trajectory pattern discovery are classified by according to a variety of aspects [79].

Two typical mining tasks on trajectories are clustering and join. Clustering aims at discovering groups of similar objects from a single trajectory collection. A join is a specific operation that computes pairs of similar objects from two trajectory collections [102].

Clustering of trajectories Clustering is the process of organizing objects into groups so that the members of a group are similar. It is a fundamental concept in data mining, due to its wide spectrum of applications.

Trajectory join Some trajectory patterns are defined and computed by means of join queries. Given two data sets, spatio-temporal joins find pairs of elements from the sets which satisfy the given predicate [24, 41]. For example, Tao *et al.* [86] show how join queries are processed by using the time-parameterized methods [78, 82, 85].

The close-pair join [87] finds all object pairs (o_1, o_2) from $P_1 \times P_2$ with distance $D\tau(o_1, o_2) \leq e$ within a time interval τ, where e is a user-specified distance. Planesweep techniques [73, 103] have been proposed for evaluating spatio-temporal joins. The trajectory join [14] aims at retrieving all pairs of similar trajectories in two data sets.

Granularity of trajectory patterns

The granularity of a trajectory pattern can be characterized by the time interval and the number of objects which are included in the pattern.

Individual patterns Individual trajectory pattern retrieval [37,42,74] relates to whether a set of individual trajectories are retrieved that satisfy a pattern specified in query.

Group patterns Group trajectory pattern discovery [19, 36, 43] also relates whether sets of trajectories are retrieved so that the trajectories in a set exhibit a similar pattern according to some specific notion of pattern. The discovery of group patterns may enable different forms of sharing among the objects who have similar trajectories. This class of patterns include concurrence [47], trend-setter [47], flock [16, 35, 36], leadership [8, 9], convergence [16], encounter [16], convoy [43], and swarm [53].

Relative motion patterns

It is challenging to explore and compare motion events of individuals and groups of individuals. To facilitate trajectory data analysis, the analysis concept RElative MOtion (REMO) by Laube *et al.* [46, 48–50] enables the identification of similar movements in a collection of trajectory datasets.

Temporal constraints: periodicity

In the our daily life, various type of objects periodic movement patterns. Periodic pattern mining of trajectory data concerns the discovery of periodic object behavior [54, 57]. For example, objects that follow the same routes over regular time intervals. These periodic patterns provide an insight into, and concise explanation of, periodic behaviors across long movement histories. Periodic patterns are also useful for compressing movement data [21].

Spatial constraints: movement on spatial networks

Many types of objects move in a spatial network, such as a road network. Since those objects are always located somewhere in the networks, raw trajectory data is typically modeled as or transformed to network-based trajectories, e.g., edge sequences in a road network graph [25, 44, 101].

Trajectory Indexes

In our study on trajectory mining algorithms, we use spatial index to speed up our basic algorithms. So in this section we introduce some trajectory index researches. There are mainly three index approaches. The first is to use any multidimensional access method like R-tree indexes with augmentation in temporary dimensions such as 3D R-tree [70], or STR-tree [70]. The second approach uses multiversion structures, such as MR-tree [94], HR-tree [60], HR+-tree [83], and MV3R-tree [84]. The third approach divides the spatial dimension into grids, and then builds a separate temporal index for each grid. This category includes SETI [22], and MTSB-tree [103]. Other indexes, like [80] used, to update a large number of moving objects in a system.

Chapter2

Preliminaries

In this chapter, we introduce basic definitions and notation that are used in this thesis. Based on these definitions, we introduce the class of flock patterns, and its extensions including the classes of length-maximal and size-maximal flock patterns. Finally, we define the flock pattern mining problems.

2.1 Basic definitions

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

A *point* in $\mathbb{R}^2$ is a pair $p = (x, y) = (p.x, p.y) \in \mathbb{R}^2$. For a rectangle $R = [x_0, x_1] \times [y_0, y_1]$ in $\mathbb{R}^2$, the *width* of R is given by $width(R) = \max\{|x_1 - x_0|, |y_1 - y_0|\}$. For a set $\mathbb{S} \subseteq \mathbb{R}^2$ of points, let $\mathbb{S}_x$ and $\mathbb{S}_y$ be the sets of all x- and y-coordinates of the points in $\mathbb{S}$. The *minimum bounding rectangle* (MBR) containing $\mathbb{S}$ is given by $MBR(\mathbb{S}) = [x_0, x_1] \times [y_0, y_1]$, where $z_0 = \min \mathbb{S}_z$ and $z_1 = \max \mathbb{S}_z$ for every $z \in \{x, y\}$. If $\mathbb{S} = \emptyset$, then

$MBR(\mathbb{S}) = \emptyset$, too.

Lemma 1. *Let $\mathbb{S}$ be any set of points. any rectangle R in $\mathbb{R}^2$,*

- *$MBR(R \cap \mathbb{S}) \subseteq R$,*
- *$X \subseteq Y$ implies $MBR(X) \subseteq MBR(Y)$,*
- *$MBR(\widehat{R} \cap \mathbb{S}) = \widehat{R}$ for $\widehat{R} = MBR(R \cap \mathbb{S})$.*

Let $(A, \leq)$ is a partially ordered set. The *greatest lower bound* (GLB) of a subset $X \subseteq A$ is the element $\hat{x} \in A$ that satisfies the following (i) and (ii):

i $\hat{x} \leq y$ for any $y \in X$ (a lowerbound of X), and

ii for any $x \in A$, if x is a lowerbound of X then $x \leq \hat{x}$ holds.

2.2 Trajectory databases

2.2.1 Two-dimensional discrete trajectories

Let $\mathbb{A} = \mathbb{R}^2$ and $\mathbb{T} = \mathbb{N}$ be the domains of 2-dim points and discrete time stamps, respectively. For a *2-dim point* (a *point*) $p = (x, y) \in \mathbb{A}$, we denote its x- and y-coordinates by $p.x = x$ and $p.y = y$. A *time stamp* is an element $t \in \mathbb{T}$.

In this section, we give our model of trajectory data according to Benkert *et al.* [16]. Let $m, n \geq 0$ be any non-negative integers. 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 objects and all time points, respectively. Let $\tau_1, \ldots, \tau_T$ be specified time steps with $\tau_{j-1} < \tau_j$ for every $j > 1$.

2.2.2 Trajectory databases

Let n and $T \geq 1$ are pre-determined nonnegative integers, which indicate the number of moving objects and the maximum value for discrete time stamps, respectively. Let $\mathbb{R}^2$ be the 2-dimensional continuous space, or the *plane*.

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

$$S = \{ s_i \,|\, i = 1, \ldots, n \} \subseteq (\mathbb{R}^2)^T \tag{2.1}$$

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

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

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

of T points on the 2-dimensional space $\mathbb{R}^2$ such that its t-th point is $s_i[t] = (p_t^i)_{i=1}^T = (x_t^i, y_t^i) \in \mathbb{R}^2$.

In the case for either an unevenly sampled sequence $\hat{s}_i$, or a continuous function $\hat{s}_i : [0, \infty) \to \mathbb{R}^2$ as in real trajectory data, we simply transform it to an evenly sampled trajectory $s_i = (p_j^i)_{i=1}^n$ by sampling n consecutive points from $\hat{s}$ at the specified time steps $1, \ldots, T$. We also assume that $\mathbb{O}$, S, n, and T are fixed, otherwise stated.

Example 1. In Fig. 2.1, we show an example of a trajectory database S, which consists of five trajectories of length $T = 7$.

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

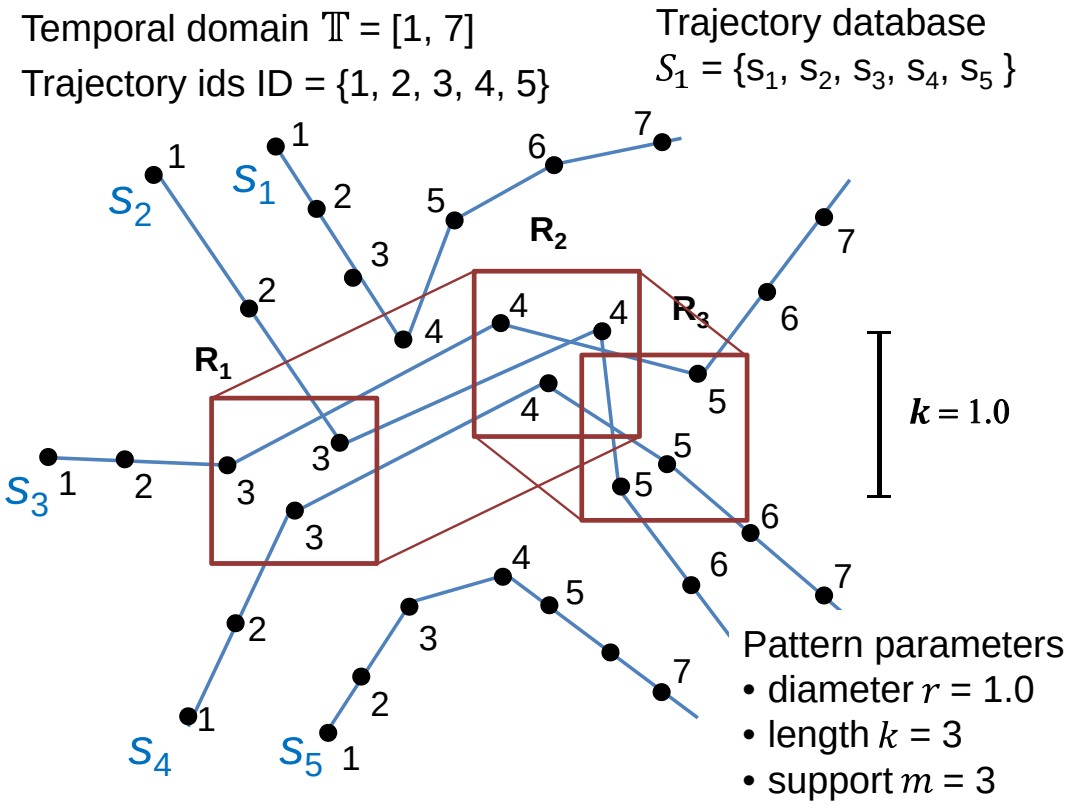

Figure 2.1: A 2-dim envelope pattern $P_1 = ((R_1, R_2, R_3), 3, 5)$ in $\mathbb{R}^2$ containing three moving objects $2, 3, 4$ at time steps t_3, t_4, t_5 in $I = [t_3..t_5]$ in a set of trajectories, where R_1, R_2, and R_3 are rectangles with width $\leq \theta$. This envelope pattern is not closed since the rectangular regions are not minimum bounding rectangles of their members.

2.2.3 Geometric indices

We can show that any trajectory i in ID must be contained in the rectangle $R = R(c, 2r)$ of size $2r \times 2r$ given by

$$R(c, 2r) = [x-r, x+r] \times [y-r, y+r] \subseteq \mathbb{R}^2, \tag{2.2}$$

where $x = p.x$, and $y = p.y$.

A *spatial index*, such as quad tree or range tree [27], compactly stores a set $U \subseteq D$ of points supporting the range query

$$U.Range(R) \triangleq U \cap R \ \subseteq U \tag{2.3}$$

in sub-linear time.

2.3 Classes of flock patterns

2.3.1 The class FP of flock patterns

For trajectory databases, we introduce the class $\mathcal{FP}$ of spatio-temporal patterns, called flock patterns. Formally, the class of flock patterns is defined as follows.

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 [35].

Definition 1 (FP)**.** For any integers $m, k \geq 1$ and real number $r > 0$, an (m, k, r)-*flock pattern* (or a *flock pattern*, for short) [35] is a pair $P = (X, I) = (X, [b, e])$ of a set $X \subseteq \mathbb{O}$ of moving objects 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

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

For a flock pattern $P = (X, I)$, we denote by $P.set = X$ the *set of moving objects*, by $P.span = I$ the *duration*, by $P.start = b$ the *start time*, by $P.end = e$ the *end time*, by $len(P) = len(I) = e - b + 1$ the *length*, and by $size(P) = |P.set|$ the *size* of P. We also define the *width* of P as follows, which plays an important role. 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

$$S[X][t] = \{\, s_i[t] \,|\, i \in X \,\} \subseteq D. \tag{2.4}$$

We call this set the *snapshot* of P at time t. For any $t \in I$, the minimum radius of such an enclosing disk that contains all points of X at time point t is called the *width of P at time t*. Then, the *width* of P, denoted by $width(P)$, is the maximum of the width of P over all $t \in I$. If $rad(U)$ denotes by the radius of point set U, we can write

$$width(P) = \max_{t \in I} rad(S[X][t]) \geq 0. \tag{2.5}$$

Sometimes, we denote $||P||_\infty^S$ the $width(P)$ of P. By definition, an (m, k, r)-flock pattern is a pair $P = (X, I)$ with $X \subseteq \mathbb{O}$ and $I \subseteq \mathbb{T}$ such that $size(P) \geq m$, $len(P) \geq k$, and $width(P) \leq r$.

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 width $\leq r$, but no restriction on their size and length. We refer to such a pattern as an *r-flock pattern*.

The *diameter* of a set $A = \{p_1, \ldots, p_n\}$ of points, denoted by $|| A ||_\infty$, is the maximum

L_∞-distance between any two points in A, defined by

$$|| A ||_\infty = \max_{p,p' \in A} L_\infty(p, p'). \tag{2.6}$$

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

Actually, we have the next lemma.

Lemma 2. *The width of P can be computed by Algorithm 1 in $O(m\ell)$ time, where $m = size(P)$ is the size of P and $\ell = len(P)$ is the length of P.*

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

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

2.3.2 The class RFP of rightward length-maximal flock patterns

For a given max-width parameter $r \geq 0$, it is often useful to find only (r, k)-flock patterns $P = (X, [b, e])$ whose time interval $[b, e]$ are extended rightward along time line as long as possible preserving the diameter r [35]. This idea of *length-maximal mining* is expected to reduce the number of solutions and running time than just finding all

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

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

Definition 2 (RFP)**.** A flock pattern $P = (X, [b, e])$ in S is a *rightward length-maximal* flock pattern (RFP, for short) if there is no other flock pattern $P' = (X, [b, e'])$ in S such that

(i) P' has the same ID set X as P, and

(ii) the right end of P' is strictly more larger than that of P.

By definition, any RFP in S is an FP. However, the converse does not hold in general. Thus, we have the inclusion $\mathcal{RFP}(r, k) \subseteq \mathcal{FP}(r, k)$.

2.3.3 The class MFP of two-sided length-maximal flock patterns

Gudmundsson and van Kreveld [35] 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 $m \in \mathbb{N} \cup \{*\}$ and $r \in \mathbb{R}$, an $(m, *, r)$-flock pattern P is said to be a *longest-duration* $(m, *, r)$*-flock pattern* in a database S if its duration is longest among all $(m, *, r)$-flock patterns appearing in the database S. They showed that the problem of finding at least one longest-duration flock pattern in a given S is, unfortunately, NP-hard. This indicates that there will be no efficient algorithm to find longest-duration flock patterns in polynomial time in the input size unless $P = NP$ [35].

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.

Below, we given the definition of the two-sided version of length-maximal flock patterns, called maximal-duration flock patterns.

Definition 3 (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* (MFP, for short) 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 Chapter 4, 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 the inclusion $\mathcal{MFP}_r \subseteq \mathcal{FP}_r$, while the converse does not hold in general.

2.3.4 The class RSFP of rightward length-maximal, size-maximal flock patterns

In this section, we introduce the class of size-maximal flock patterns (SFP, for short), and its subclass of rightward size-maximal flock patterns (RSFP, for short). Before introducing SFPs and RSFPs, we give the definition of the class of patterns, called envelope patterns (EVP, for short) in the following. Actually an EVP is yet another representation of a SFP.

A *2-dim trajectory envelope pattern* (or an *envelope pattern*, EVP) of length ℓ in S is a triple $P = (E, b, e)$, where $E = \langle R_1, \ldots, R_\ell \rangle$ is a ℓ-tuple of 2-dim rectangular regions

in $\mathbb{R}^2$, called an *envelope*, and b, e $(1 \leq b \leq e \leq T)$ are indices in $[1..n]$ such that $\ell = e - b + 1$, called the *start* and *end positions* of P, respectively. The *width* of P is $w = \max_{1 \leq j \leq \ell} width(R_j)$. We denote by $start(P) = b$, $end(P) = e$, $len(P) = \ell$, and $width(P) = w$. If $\ell = 0$, then $P_0 = (\langle\rangle, b, b-1)$ is the *empty pattern*, and we define $len(P_0) = 0$ and $width(P_0) = 0$.

Example 2. In Fig. 2.1, we show an example of an envelope pattern $P_1 = ((R_1, R_2, R_3), 3, 5)$ of length 3. We see that P_1 contains three moving objects $2, 3$, and 4 among five objects at time steps t_3, t_4, and t_5 in interval $I = [t_3, t_5]$.

Next, we define the semantics of envelope patterns by their cover sets. For every $1 \leq i \leq n$, the i-th trajectory s_i is *contained in* an envelope $P = (E, b, e)$ of length ℓ if for every $1 \leq j \leq \ell$, the $(b + j - 1)$-th point is contained in the j-th rectangle R_j, i.e., $p^i_{b+j-1} = s_i[b + j - 1] \in R_j$. In other words, s_i is contained in P iff the subsequence $(p^i_b, \ldots, p^i_e)$ is contained in the set $R_1 \times \cdots \times R_\ell$.

Definition 4 (cover set)**.** *For a rectangle $R \subseteq \mathbb{R}^2$, we define the cover set of R at time $1 \leq t \leq T$ by $\boldsymbol{Cov}_{\mathcal{S}}(R, t) = \{\, i \in \mathbb{O} \mid S_i[t] \in R \,\}$. Given an envelope pattern $P = (E, b, e)$, the* cover set *of P in S is the set $\boldsymbol{Cov}_{\mathcal{S}}(P)$ of objects whose trajectories belong to E within interval $[b, e]$, that is,*

$$\boldsymbol{Cov}_{\mathcal{S}}(P) = \{\, i \in \mathbb{O} \mid S_i[t_j] \in R_j \text{ for all } j \in [1, \ell] \,\} \tag{2.7}$$

$$= \bigcap_{j \in [1,\ell]} \boldsymbol{Cov}_{\mathcal{S}}(R_j, t_j), \tag{2.8}$$

where $t_j = b + j - 1$ for every $j \in [1, \ell]$.

Definition 5. *Given an envelope $P = (E, b, e)$, the flock pattern corresponding to P is defined as $P' = (X_P, I_P)$ such that*

- $X_P = \boldsymbol{Cov}_S(P)$
- $I_P = [b, e]$.

We use $EV(P) = P'$ to denote the envelope pattern of P. Now, we define the class of size-maximal flock pattern, SFP, in terms of envelope patterns, EVP.

Definition 6. *A flock pattern $P = (X, I)$ is a* size-maximal flock pattern *(SFP, for short) in a database S if there is no other flock pattern $P' = (X', I')$ such that*

- $EV(P) = EV(P')$,
- $I = I'$,
- $X \subset X'$.

In the above definition, each SFP has the unique EVP. Therefore, we identify an SFP and equivalent EVP.

Next, we define the class of rightward length-maximal, size-maximal flock pattern, RSFP, using envelope patterns, EVP.

Definition 7. *A flock pattern $P = (X, I) = (X, [b, e])$ with length ℓ is a* rightward length-maximal, size-maximal flock pattern *(RSFP, for short) in S if there is no other flock pattern $P' = (X', I') = (X', [b', e'])$ with length ℓ' such that*

- *$EV(P) = (E, [b, e])$, $EV(P') = EV(E', [b', e'])$, $\ell' \geq \ell$, and $R_i = R'_i$ hold for every $i = 1, \ldots, \ell$, where $E = \langle R_1, \ldots, R_\ell \rangle$ and $E' = \langle R'_1, \ldots, R'_\ell \rangle$.*
- *$b = b'$ and $e \leq e'$,*
- *$X \subseteq X'$.*

Similar to RSFP defined above, we can also introduce such an EVP that cannot be extended rightward. Now, we give such a rightward length-maximal counterpart of an EVP, called a rightward length-maximal EVP as follows.

Definition 8 (rightward length-maximal EVPs)**.** *Let $P = (E, b, e)$ be an EVP, where $E = \langle R_1, \ldots, R_\ell \rangle$ is an envelope of length ℓ. Then, P is a* rightward length-maximal envelope pattern *(REVP, for short) if there is no strictly longer EVP $P' = (E', b', e')$ with length $\ell' > \ell$ and the same cover set, that is, P satisfies the following condition (i) and (ii):*

(i) $\boldsymbol{Cov}_{\mathcal{S}}(P) = \boldsymbol{Cov}_{\mathcal{S}}(P')$.

(ii) $b = b'$, $e < e'$, *and thus,* $\ell < \ell'$.

In the next section, we consider the relationship between SFPs and EVPs.

2.4 Equivalence between the class of envelope patterns and the class of flock patterns

In this section, we show the equivalent relationship between the class of trajectory envelope patterns (EVP) and the class of flock patterns. The following lemma is essential to the discussion in this subsection.

Lemma 3. *For any flock pattern $P = (X, I)$, the following conditions (a) and (b) are equivalent each other:*

(a) P is a size-maximal flock pattern in S, and

(b) $X = \boldsymbol{Cov}_{\mathcal{S}}(EV(P))$.

First, we start with the conversion from an EVP to a SFP as follows.

Lemma 4. *Given any EVP $P = (E, b, e)$ with length ℓ in S, where $E = \langle R_1, \ldots, R_\ell \rangle$ and $freq_{\mathcal{S}}(P) > 0$, we could compute a flock pattern $Q = (X, [b, e])$, which satisfies the following conditions:*

1. $X = \bigcap_{j \in [1,\ell]} \boldsymbol{Cov}_{\mathcal{S}}(R_j, b + j - 1)$,
2. $\boldsymbol{Cov}_{\mathcal{S}}(P) = ID(X)$,
3. $width(Q) = width(P)$,
4. $len(Q) = len(P)$.

Proof. By the definition of cover set in Definition 2.4, the cover set of the EVP P equals the set of IDs $X = \bigcap_{j \in [1,\ell]} \mathbf{Cov}_{\mathcal{S}}(R_j, b + j - 1)$. This satisfies conditions 1 and 2. According to definitions of width and length of EVP and flock pattern respectively, we have condition 3 and 4. Then we would show Q is size-maximal flock pattern for P. First, we assume that there is a flock pattern $P' = (X', [b, e])$ that all the trajectories in P' are contained by EVP P and $X' \supset X$. Then we could find a trajectory s_i contained in X' but not contained in X for some ID i. According to our assumption and condition 2, s_i should be included by EVP P. By assumption, we observe that its ID i is included in X. This contradicts the assumption. By contradiction, this shows the lemma. □

Corollary 1. *The flock pattern Q which is computed from an EVP P in the previous lemma 4 is also a sized-maximal flock pattern in S'.*

Conversely, we show the conversion from a SFP to an EVP as follows.

Lemma 5. *Given any flock pattern $P = (X, [b, e])$ in S, we could compute EVP $Q = (E, b, e)$, which $E = \langle R_b, \ldots, R_e \rangle, \ell = e - b + 1$ and Q satisfies the following conditions:*

1. $R_i = MBR(S[X][i]), i = [b, e]$,

2. $\boldsymbol{Cov_S}(Q) \supseteq ID(X)$,

3. $width(Q) = width(P)$,

4. $len(Q) = len(P)$.

Proof. According to the definition of $MBR(\mathbb{S})$ in section 2.1, we could compute R_i in condition 1. Then by the definition of cover set in section , every trajectory data included in X is also contained in P'. But some trajectory data included in P' may be not in X. Condition 2 is granted. According to definitions of width and length of EVP and flock pattern respectively, we have condition 3 and 4. □

Based on the above lemma, in Algorithm 14, we present the algorithm EVPTOFP that computes the equivalent size-maximal flock pattern from a given EVP. For the classes of RSFPs and REVPs, we can show an equivalence between them similar to the above lemmas. However, we omit the details since it is similar to the above lemmas.

2.5 Flock pattern mining problems

Let $\mathcal{C} \in \{\mathcal{FP}, \mathcal{RFP}, \mathcal{MFP}, \mathcal{RSFP}, \ldots\}$ be any class of flock patterns and its extensions, and let $\Theta = (r, k, m, \ldots)$ be a tuple of any parameter values on constraints. Then, we denote by $\mathcal{C}(\Theta)$ the class of all flock patterns within the class $\mathcal{C}$ with constraint Θ, that is, the class of all Θ-flock patterns in $\mathcal{C}$. For example, for any positive real number $r > 0$, non-negative integers k and m, $\mathcal{FP}(r, k)$, $\mathcal{RFP}(r, k)$, and $\mathcal{MFP}(r, k)$ denote the classes of all (r, k)-flock patterns, all (r, k)-rightward length-maximal flock patterns, and all r-maximal duration flock patterns (or maximal duration r-flock patterns), re-

spectively. Similarly, $\mathcal{FP}(r, k, m)$ and $\mathcal{RSFP}(r)$ denote the classes of all (r, k, m)-flock patterns and all size-maximal r-flock patterns, respectively.

Now, we state our data mining problem as follows.

Definition 9 (flock pattern mining problem for pattern class $\mathcal{C}$)**.** Let $\mathcal{C}$ be any class of flock patterns and let Θ be a tuple of any parameter values on constraints. An *input* to the problem is a tuple (S, Θ) of an input trajectory database S, and a tuple Θ of parameter values. Then, the *task* is to find all flock patterns P in S within class $\mathcal{C}$ without repetition that satisfy the constraint specified with given Θ.

In general, there are exponentially many solutions for an enumeration problem. 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. 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 [11, 90].

Specifically, we evaluate the performance of a flock pattern mining algorithm $\mathcal{A}$ in terms of *enumeration algorithms* [12]. Let N and M be the input size and the number of patterns as solutions.

- A pattern mining algorithm $\mathcal{A}$ is said to have *polynomial delay* (poly-delay) if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial $p(N)$ in N.

- $\mathcal{A}$ is of *polynomial space* (poly-space) if the maximum size of its working space, in

addition to that of output stream O, is bounded by a polynomial $p(N)$.

Our goal here is to devise an efficient enumeration algorithm that finds all flock patterns within a given class $\mathcal{C}$ and constraint Θ in polynomial delay and polynomial space in the total size N of the input database S.

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.

Chapter3

Depth-First Mining Algorithms for Rightward Length-Maximal Flock Patterns

In this chapter, we present our pattern mining algorithms for flock patterns and rightward length-maximal flock patterns. We also give a speed-up technique using geometric indexes to prune redundant candidates. Then we run experiments on a trajectory dataset to examine its performance.

3.1 A basic DFS algorithm FPM for FPs

3.1.1 The algorithm

We first present a basic mining algorithm **FPM** (basic flock pattern miner) for FPs. In Algorithm 2 we present the algorithm **FPM** with its subprocedure **RecFPM** for mining (r, k)-FPs. In the overall design of our algorithm **FPM**, we employ DFS (depth-first search) procedure according to pattern growth approach, as in PrefixSpan [69], Eclat [96] and LCM [90].

In DFS (or pattern growth) approach, a recursive mining procedure searches for all descendant of the current pattern from smaller to larger in depth-first manner using

backtracking. The advantage of DFS approach is that DFS miners are proven fast in main memory environment and can be easily implemented as a simple recursive procedure.

At the top-level of **FPM**, for each possible length $k \in [k, T]$ no less than k, it invokes the recursive subprocedure **RecFPM** given as arguments an initial pattern $P_0 = (X_0, [b_0, e_0])$ consisting of a singleton ID set $X_0 = \{i_0\}$ and an interval $[b_0, e_0]$ for every possible combination of $i_0 \in ID$ and $b_0 \in [0, T]$. The end time e_0 is calculated by b_0 and k.

The recursive subprocedure **RecFPM** is a DFS algorithm (or a backtracking algorithm) that searches the hypothesis space of all r-flock patterns with length exactly k as follow. Starting from the initial pattern $P_0 = (X_0, [b_0, e_0])$ consisting of a singleton ID set $X_0 = \{i_0\}$, the procedure enumerates all subsets X of ID using a backtracking algorithm similar to depth-first search algorithms for frequent itemset mining, such as Eclat [96] and LCM [90]. For each generated subset X, the procedure forms a candidate (r, k)-flock pattern $P = (X, [b, e])$ with a specified interval $[b, e]$. Then, the algorithm computes the width $|| P ||_\infty$ of the pattern P by accessing the trajectories in S, and checks if P satisfies the width constraint $|| P ||_\infty \leq r$. If the condition is violated, then it prunes the search for P and all of its descendants.

3.1.2 Analysis of the algorithm

This *width-based pruning rule* is justified by the following lemma, which says the class of (r, k)-patterns has the anti-monotonicity w.r.t. set inclusion of their ID sets.

Lemma 6 (anti-monotonicity)**.** *Let $P_i = (X_i, I_i)$ are two flock patterns, where $i = 1, 2$. If P_2 is an (r, k)-flock pattern in S and if $X_1 \subseteq X_2$ and $I_1 \subseteq I_2$ hold, then P_1 is also an (r, k)-flock pattern in S.*

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

On the running time and space of the algorithm **FPM**, the following proposition is easily derived from our manuscript [10].

Proposition 2. *[10] Let S be an input trajectory database S of n trajectories with length T. Then, the algorithm **FPM** in Algorithm 2 solves the flock pattern mining problem for the class $\mathcal{FPM}(r, k)$ of (r, k)-flock patterns in S. It uses $O(knT^2)$ time per pattern and $O(k^2)$ words of space, respectively, where $k = supp(X) = |X|$ is the support of the pattern X being enumerated.*

From the practical view, $O(T^2)$ term in the time complexity of **FPM** is too large to apply it to long trajectories with large T. We point our that this $O(T^2)$ term come from the doubly nested for-loop in Lines 2 and 3 of Algorithm 2. In the next subsection, we will see how we can remove this $O(T^2)$ term by focusing on mining of RFPMs.

3.2 A modified algorithm RFPM for RFPs

Next, we present a modified mining algorithm **RFPM** (rightward flock pattern miner) for RFPs (rightward length-maximal flock patterns), the class of rightward length-maximal flock patterns. In Algorithm 7, we present the algorithm **RFPM** with its subprocedure **RecRFPM** for mining (r, k)-RFPs.

3.2.1 Basic idea: Rightward horizontal closure

From the view of frequent pattern mining, RFPs in a trajectory database are a sort of *closed patterns*, which have been extensively studied in frequent itemset mining (FIM) field [90, 97] as well as formal concept analysis (FCA) field. Many efficient closed pattern mining algorithms use a class of operation, called *closure* operation, which enlarge a given, possibly non-closed pattern to obtain its *closed* version.

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

Definition 10 (rightward horizontal closure)**.** Let $P = (X, I = [b, e])$ be any flock pattern in a database S. Then, the *rightward horizontal closure* of P in S, denoted by **RH_Closure**$(P; S, r)$, is the unique flock pattern $P_{max} = (X, I = [b, e_{max}])$ such that $e_{max} \in [0, T]$ is the maximum value of end position e' satisfying the equality

$$|| P'(X, [b, e']) ||_\infty = || P ||_\infty. \tag{3.1}$$

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

In Algorithm 6, we show the procedure **RH_Closure** that computes the rightward horizontal closure of non-RFP P in $O(m\ell)$ time, where $m = supp(P) = |X| = O(n)$ and $\ell = len(P_{max}) = O(T)$.

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

Lemma 7 (characterization)**.** *Let $P = (X, [b, e])$ be an (r, k)-flock pattern in S. Then, P is rightward length-maximal if and only if*

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

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

From the above lemma, we have the correctness below.

Lemma 8. *[10] The rightward horizontal closure* P_{max} *of a possibly non-rightward length-maximal r-FP P is the unique longest r-RFP such that the ID sets and the start time are identical to those of P.*

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

3.2.2 The algorithm

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

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

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

- Receiving a partial RFP $P_* = (X, b, *)$ as arguments, the recursive procedure **RecFPM** computes the rightward horizontal closure $P = (X, [b, e])$ from P_* by the procedure **RH_Closure** with max-width r.

- Next, if the obtained RFP P satisfies (r, k)-constraints, then output it. Otherwise, we safely prune all descendants as before.

- Finally, **RecFPM** recursively calls its copy with an extended pattern $P_1 = (X \cup \{i\}, [b, *])$. To avoid duplicated generation of patterns, the id i is removed from the universe ID.

From a similar argument to [90] based on reverse search technique of [12], we have the following time and space complexities of **RFPM**.

Theorem 3. *[10] Let S be an input trajectory database S of n trajectories with length T. Then, the algorithm **RFPM** in Algorithm 7 solves the flock pattern mining problem for the class $\mathcal{RFPM}(r, k)$ of (r, k)-flock patterns in S. It uses $O(knT)$ time per pattern and $O(k^2)$ words of space, respectively, where $k = supp(X) = |X|$ is the support of the pattern X being enumerated.*

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

3.3 A faster algorithm G-RFPM with using geometric index

In this subsection, we present a speed-up technique using geometric index in $\mathbb{R}^2$, called *geometric database reduction*, which achieve order of magnitude acceleration of both of

FPM and **RFPM** algorithms, which is orthogonal to the rightward horizontal closure technique.

3.3.1 The algorithm

In Algorithm 5, we present our modified mining algorithm **G-RFPM** (grid-based flock pattern miner) based on **RFPM** using geometric constraint on the 2-dimensional plane for (r, k)-patterns. The algorithm uses **RecFPM** in Algorithm 5 as subprocedure.

Given a trajectory database S, maximum width $r > 0$ and minimum length k as arguments, the algorithm **G-RFPM** starts with selecting a combination of a trajectory id i_0 in ID and a starting time b_0 in $\mathbb{T} = [1, T]$ as in the original **FPM** or **RFPM**.

Let $i_0 \in ID$ and $b_0 \in [1, T]$ be any pair of trajectory ID and start time. Then, we know that the trajectory s_{i_0} starts from the point $c = s_{i_0}[b_0]$ in the database. Now, we assume to find any (r, k)-flock pattern of the form $P = (X, [b_0, *])$ be any (r, k)-pattern such that $i_0 \in X$ in the database.

Consider the $t = b_0$ time slice U of S, that is, the set U of all points with the specified time $t = b_0$, given by

$$U = \{\, p = s_i[t] \,|\, s_i \in S, i \in ID, t = b_0 \,\}. \tag{3.2}$$

We have defined the rectangle $R = R(c, 2r)$ in section 2.2.3. Let $P = (X, [b, e])$ with $b = b_0$ be any target (r, k)-flock patterns in S. For any trajectory ID i, if the ID i belongs to X then the corresponding trajectory s_i starts from any point in $U \cap R(c, 2r)$. Therefore, we can reduce the original domain ID of candidate IDs for X to the following smaller sub-domain

$$ID(R) = \{\, i \in ID \,|\, p_i = s_i[b_0] \in U \cap R \,\}. \tag{3.3}$$

By using an appropriate geometric index, such as quad trees or range trees, we can compute $ID(R)$ by making the range query

$$ID(R) = U.Range(R) \tag{3.4}$$

in $q = O(\log^2 m)$ time by quad trees, or $O(\log^2 n)$ time by range trees using $O(n \log n)$ time preprocessing of S, where $n = |U| = |S|$ and $m = || U ||_\infty$ are the L_∞-diameter of points in U. Then, the total overhead becomes $O(N \log^2 n)$ time [10], which is linear in input size N with poly-logarithmic factor.

As shown in Sec. 5.3, the above modification on **RFPM** to obtain **G-RFPM** greatly reduces the time complexity of the algorithm.

3.4 Experiments

We ran experiments on synthesis datasets to evaluate the efficiency of our algorithms. Finally, we ran the experiment on real typhoon dataset to show that RFPs are meaningful for real applications.

3.4.1 Data

Synthesis data

We generated a sets of implanted synthesis trajectory datasets using our data generator implemented in C++ as follows. Let $n = 200$ and $T = 200$. Our data set is a collection of random trajectories in which C copies of random patterns are implanted as follows. We first fixed $a \times a$ area $\mathbb{A}$ in the plane, where $a = 40.0$, and then generated a set of n trajectories of length T by uniform distribution on $\mathbb{A}$. Then, we embed C copies of each of K random short trajectories of length L_* are implanted in some of generated

Table 3.1: Parameters of the real data experiment

category	parameter	notion	value
trajectory data	amount of trajectories	n	322
	length of every trajectory	T	50
	total number of points	$N = nT$	16,100

trajectories, where location of the copies are randomly perturbed within width r_*. In our experiments, we set $C = 5$, $K = 6$, $L_* = 20$, and $r_* = 1.0$. The other parameters are varied in experiments.

Real data

From the website of digital-typhoon [45][1], we download 1668 files of the typhoon trajectory data happened from 1951 to October, 2014. We choose 322 files. These files contain the trajectory data which are no shorter than 50 points (about 300 hours). Then we choose the front part of these trajectory data from the head to the 50th point of the trajectory data. In this way, the length of our real typhoon trajectory data which we choose to run experiments is 50. These trajectories happened in different time. But we use the time interval between two adjacent points as relative time interval.

3.4.2 Methods

We implemented our algorithms **FPM** (BFPM), **RFPM** (BFPM R), and **G-RFPM** (GFPM R) of Sec. 3 in C++. We also implemented a simple grid-based geometric index in C++, where the plane is divided into $b \times b$ grid cells, and cells are looked up by constant time random access followed by sequential scan of a point list, where $b = 5$

[1] `http://agora.ex.nii.ac.jp/digital-typhoon/`

most time.

We compiled the above programs by g++ of GNU, version 4.6.3. We used a PC with Intel(R) Xeon(R) CPU E5-1620, 3.60GHz with 32GB of memory on OS Ubuntu Linux, version 12.04. We used the following default parameters otherwise stated: our algorithms use width $r = 1.0$, length $k = 20$, and min-sup is $m = 5$ for patterns.

In the experiments, we varied as data parameters, the number n and length T of input trajectories, and as mining parameters, the minlen k, minsup m, and minwid r. We used default values for other values. We note that in Exp 1a and Exp 1b, only the number of false random trajectories is varied, while the numbers C and K of the copies and the true patterns are kept constant. In plots below, each line indicates the running time, while the number attached to each mark indicates the number of solutions. In real data experiment, we ran the experiment on the real data of Section 3.4.1 with the parameters as Table 3.1.

3.4.3 Results

Results A: the speed-up by rightward length-maximal flock patterns

In this subsection, we examine the effect of mining of RFPs (rightward length-maximal flock patterns) introduced in Sec. 3.2, compared to mining of FPs (original flock patterns). For the purpose, we measure the number of solutions and the running time by running **RFPM** (BFPM R, in plots) of Sec. 3.2 for mining all RFPs with length $\geq k$, compared to basic **FPM** (BFPM) of Sec. 3.1 for mining all FPs with length $\geq k$.

Exp 1a: In Fig. 3.1, we show the running time and the number of patterns of by varying the number of points of input size n from 60 to 100 trajectories.

Exp 2a: In Fig. 3.2, we show the running time by varying the length of input

trajectory database from 100 to 200 trajectories.

From Exp 1b and Exp 2 above, we see that the algorithm **RFPM** exactly detect the number $K = 6$ of true patterns, while **FPM** detects the larger numbers depending on n.

Exp 3a: In Fig. 3.3, we show the running time and the number of patterns by varying the minlen k of mining parameter from 20 to 100.

For example, in the case of minlen is $k = 20$ points, the running times for **FPM** and **RFPM** are 72.92 (sec) and 1.44 (sec), respectively, resulting around 50 times speed-up, while the numbers of solutions are 594 and 6 patterns, resulting around 100 times reduction.

Exp 4a: In Fig. 3.4, we show the running time and the number of patterns by varying the minsup m of mining parameter from 6 to 10. For this experiment, we generate patterns with the support of 10.

Exp 5a: In Fig. 3.9, we show the running time and the number of patterns by varying the maxwidth r of mining parameter from 1 to 5. For this experiment, we fix $a \times a$ area $\mathbb{A}$ to $a = 500.0$.

Summary of results A: Overall, **RFPM** with RFPs is around 50 times faster than **FPM** with FPs as well as the number of RFPs is around 100 times smaller than that of FPs at maximum in our experiments. Specifically, we obtain the larger speed-up by RFP, the longer the input trajectories, or the smaller the minlen of flock patterns, as expected by theory [10].

Results B: the speed-up by geometric database reduction

In this subsection, we examine the speed-up by geometric database reduction technique introduced in Sec. 3.3. The task is mining all RFPs in a database. We compared

two algorithms **RFPM** (BFPM R, in plots) of Sec. 3.1 and **G-RFPM** (GFPM R, in plots) of Sec. 3.3, without and with geometric database reduction, respectively. Note that the numbers of solutions are same between two algorithms since they solve the same task.

Exp 1b: In Fig. 3.5, we show the running time and the number of patterns by varying the number n of input points from 20K to 200K points, where $T = 200$. For example, in the case of the input with 200K points, the running times for **RFPM** and **G-RFPM** are 61.61 (sec) and 0.96 (sec), respectively, resulting around 70 times speed-up.

Exp 2b: In Fig. 3.6, we show the running time and the number of patterns by varying the length T of input trajectory database from 0.2K to 1K points, where $n = 200$.

Exp 3b: In Fig. 3.7, we show the running time and the number of patterns by varying the minlen k of mining parameter from 20 to 100.

Exp 4b: In Fig. 3.8, we show the running time and the number of patterns by varying the minsup m of mining parameter from 6 to 10.

Exp 5b: In Fig. 3.10, we show the running time and the number of patterns by varying the maxwidth r of mining parameter from 1 to 5.

Summary of results B

Overall, in the task of mining RFPs, the modified algorithm **G-RFPM** with geometric database reduction improves the performance of **RFPM** more than ten to 70 times on the basic algorithm **FPM**. In actual running time, **G-FPM** found all RFPs in less than a second on a PC from an input database consisting of totally 0.2 million points, which seems enough for actual applications.

Summary of results A and B

From Fig. 3.1 of Exp 1a and Fig. 3.5 of Exp 1b, in the case of the input with 200K points (small dataset), the running times for **FPM**, **RFPM**, and **G-RFPM** are 61.61, 0.96, 0.03 (sec), respectively. From these timing, the speed-ups from **FPM** to **RFPM** and **RFPM** to **G-RFPM** were around 64 times and 32 times. Overall, we obtained the total speed-up of around 2,000 times from the basic **FPM** to most advanced **G-RFPM**.

Results C: the application on typhoon trajectory data

In order to test our algorithms with real data, we choose 320 typhoon trajectory data which includes 16000 points as the input dataset. We use the algorithm **G-RFPM** with the parameters of $k = 15$ points (about 90 hours) and $m = 3$ to find RFPs. The unit of measurement of the pattern length and width is longitude and latitude. As a result, when the maxwidth is $r = 2$ (about 200km), **G-RFPM** spends 0.08 seconds finding 10 RFPs. In addition, when $k = 15, 30, 35$ and $r = 3.0$, it spends less than 0.2 seconds finding 1442, 9 and 4 RFPs respectively.

In Table 3.2, we show that there are three typhoon traces included in one of the ten RFPs which are found by **G-RFPM**. We show them in Fig. 3.11. According to the birth place of typhoon traces in the bottom of the figure, the ID number of them is 200603, 195504 and 198712, respectively[2]. Every point represents a sample point in the passed 6 hours. We use deep blue to show the RFP in this figure. The length of movement of No. 200603 typhoon is 3872km. And maximum diameter is 280km [45].

According to this figure, the length of the RFP is about 1000km and the width is about 2.0 longitude and latitude (about 200km). This RFP is from the Celebes Sea in

[2]No. 195504 typhoon is the 4th typhoon happened in 1955.

Table 3.2: The information of the pattern P_2.

Num	Birth	Lifttime	Length of movement(km)	Average speed (km/h)
195504	1955/04/17	10.0	2705	11.0
198712	1987/08/22	9.5	4966	21.8
200603	2006/06/30	9.8	3872	16.6

the southeast of the city of Manila extending to westward about 1000km. It shows the common feature of the three typhoon traces. Because this pattern is RFP, there is no longer pattern which includes the same start time and typhoon trajectory data.

Summary of results C

There are $\binom{320}{3} \times 35^3 = 1.392 \times 10^{12}$ combinations which include more than 3 trajectory data from total 320 trajectory data. According to the completeness of the algorithm **G-RFPM** and the result that there are 10 RFPs found from all combinations of all trajectory data, we could affirm that there are only 10 RFPs which the width is no more than 200km and the length is longer than 90 hours. Since **G-RFPM** could find all RFPs from 320 typhoon trajectory data, it is useful for real trajectory data.

3.5 Discussion

In this chapter, we showed empirical study of trajectory mining algorithms, called **FPM**, from trajectory data. We implemented two of recent theoretical progress of depth-first algorithms for mining flock patterns [10] based on the *pattern-growth approach* [69]. The experimental results demonstrated that both of extensions, RFPs and geometric database reduction, improve on the speed of **FPM** by orders of magnitude.

To scale out flock pattern mining to bigdata in cloud environment, it will be interesting

future research to develop efficient implementation of our **FPM** in massively parallel environment, such as *map-reduce* or *hadoop*, on cloud environments

Algorithm 2 A basic DFS algorithm **FPM** for finding all (r, k)-flock patterns in an input trajectory database S given maximum width r and minimum length k.

1: **procedure FPM**(ID, S, r, k)
2: **for** $k \leftarrow k, \ldots, T$ **do** ▷ Every length
3: **for** $b_0 \leftarrow 1, \ldots, T$ **do** ▷ Each start time in $\mathbb{T}$
4: $k_0 \leftarrow b_0 + k - 1$;
5: $ID_1 \leftarrow ID$;
6: **while** $ID_1 \neq \emptyset$ **do** ▷ Each id in ID
7: $i_0 = deletemin(ID_1)$;
8: $P_0 \leftarrow (\{i_0\}, [b_0, k_0])$; ▷ Initial pattern
9: **RecFPM**(P_0, ID_1, S, r, k);
10: **end while**
11: **end for**
12: **end for**
13: **end procedure**
14: **procedure RecFPM**$(P = (X, [b, e]), ID, S, r, k)$
15: **if** $||\, P \,||_\infty{}^S > r$ **then**
16: **return** ; ▷ P is too wide
17: **end if**
18: **output** P;
19: $ID_1 \leftarrow ID$;
20: **while** $ID_1 \neq \emptyset$ **do**
21: $i = deletemin(ID_1)$;
22: **RecFPM**$(Q = (X \cup \{i\}, [b, e]), ID_1, S, r, k)$;
23: **end while**
24: **end procedure**

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

1: **procedure RH_Closure**$((X, [b_0, e_0]); S, r)$

2: $t \leftarrow b_0$;

3: **while** $|| S[X][t] ||_\infty \leq r$ **do**

4: $t \leftarrow t + 1$;

5: **end while**

6: $b \leftarrow b_0$; $e \leftarrow t - 1$;

7: **return** $(X, [b, e])$;

8: **end procedure**

Algorithm 4 An algorithm **FPM** for finding all length-maximal (r, k)-flock patterns appearing in a given trajectory database S with ID for maximum width r and minimum length k.

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

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

Algorithm 5 An algorithm **G-RFPM** for finding all length-maximal (r, k)-flock patterns appearing in a given trajectory database S with ID for maximum width r and minimum length k.

1: **procedure G-RFPM**(X, b, k, ID, S, r, k)
2: Let $S = \{ s_i \mid i = 1, \ldots, n \}$;
3: **for** $b_0 \leftarrow 1, \ldots, T$ **do** ▷ Each start time in $\mathbb{T}$
4: Build a grid index for point set $U \leftarrow S[b_0]$;
5: ▷ The time slice at time b_0
6: **for** $i_0 \leftarrow 1, \ldots, n$ **do** ▷ Each id in ID
7: $p \leftarrow s_{i_0}[b_0]$; $\delta \leftarrow r$; ▷ initial point p
8: $R \leftarrow [p.x-\delta, p.x+\delta] \times [p.y-\delta, p.y+\delta]$;
9: ▷ $2r \times 2r$-query rectangle at center p
10: $ID_0 \leftarrow U.Range(R)$; $P_0 \leftarrow (\{i_0\}, [b_0, *])$;
11: **RecRFPM**(P_0, ID_0, S, r, k);
12: **end for**
13: **end**
14: **end for**
15: **end**
16: **end procedure**

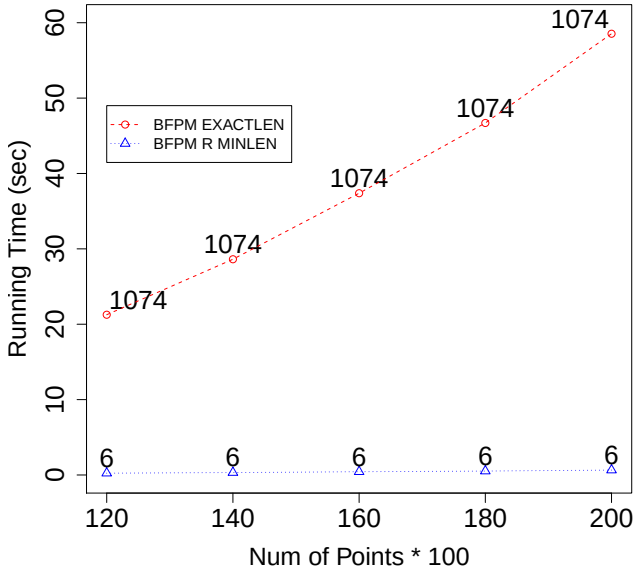

Figure 3.1: Exp 1a: The running time (and the number of patterns by mark) by algorithms **FPM** (BFPM) for FPs and **RFPM** (BFPM R) for RFPs by varying the the total number n of input points from 12K to 20K points, where a number attached to each mark indicates the number of solutions.

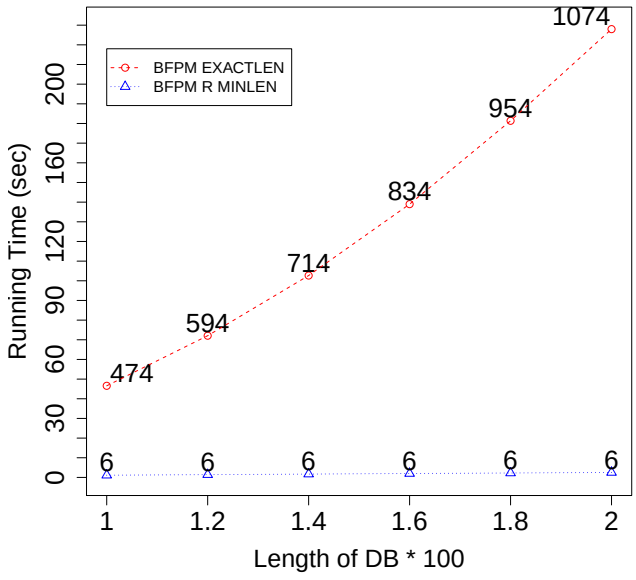

Figure 3.2: Exp 2a: The running time (and the number of patterns by mark) by algorithm **FPM** (BFPM) for FPs and **RFPM** (BFPM R) for RFPs varying the length T of input trajectories from 100 to 200 points, where a number attached to each mark indicates the number of solutions.

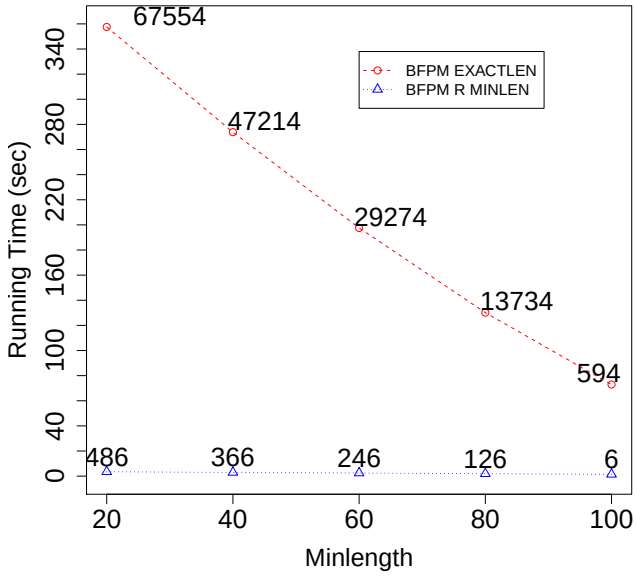

Figure 3.3: Exp 3a: The running time (and the number of patterns by mark) by algorithm **FPM** (BFPM) for FPs and **RFPM** (BFPM R) for RFPs by varying the min-len k from 20 to 100 points.

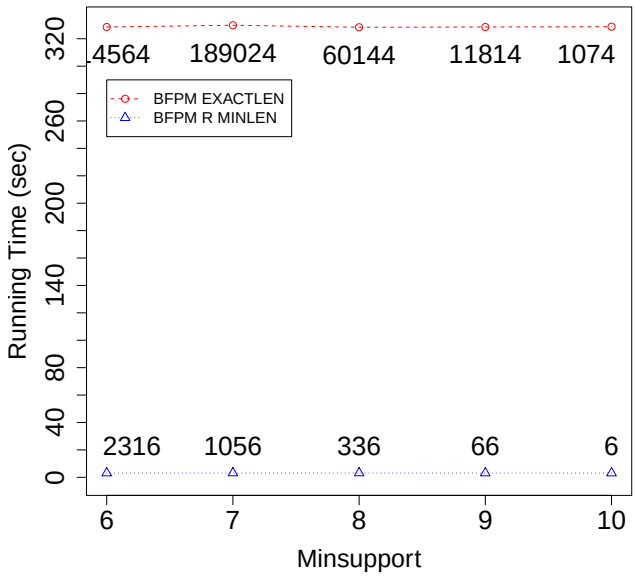

Figure 3.4: Exp 4a: The running time (and the number of patterns by mark) by algorithm **FPM** (BFPM) for FPs and **RFPM** (BFPM R) for RFPs by varying the min-sup m from 7 to 10.

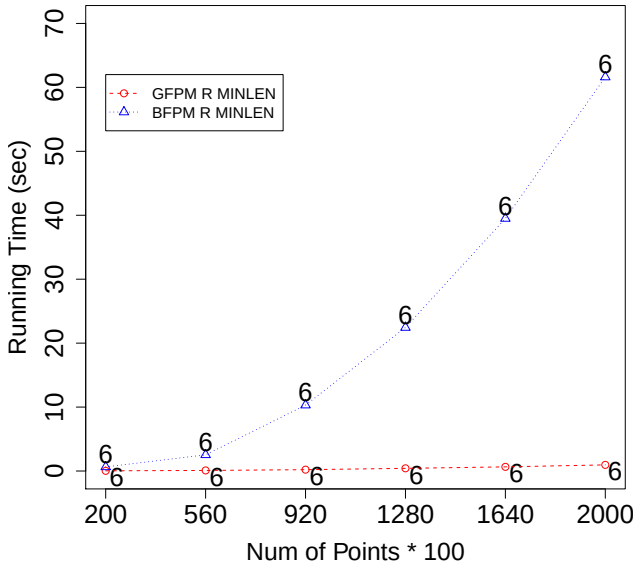

Figure 3.5: Exp 1b: The running time (and the number of patterns by mark) by algorithms **RFPM** (BFPM R) and **G-RFPM** (GFPM R) for RFPs by varying the the total number n of input points from 20K to 200K points.

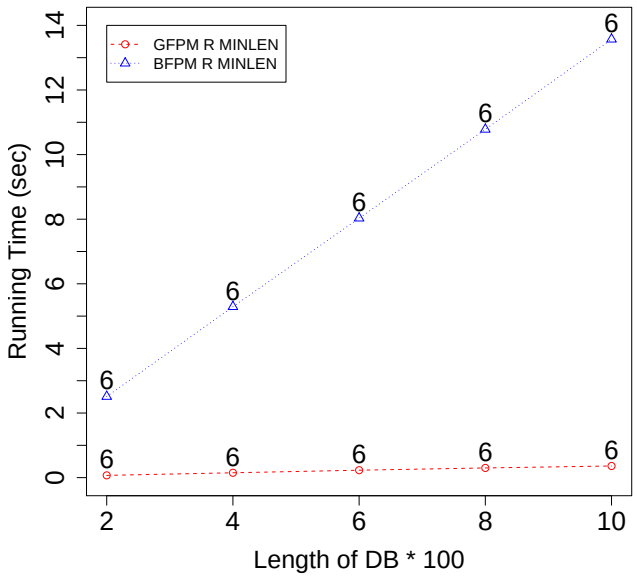

Figure 3.6: Exp 2b: The running time (and the number of patterns by mark) by algorithms **RFPM** (BFPM R) and **G-RFPM** (GFPM R) for RFPs by varying the length T of input trajectories from 200 to 1000 points.

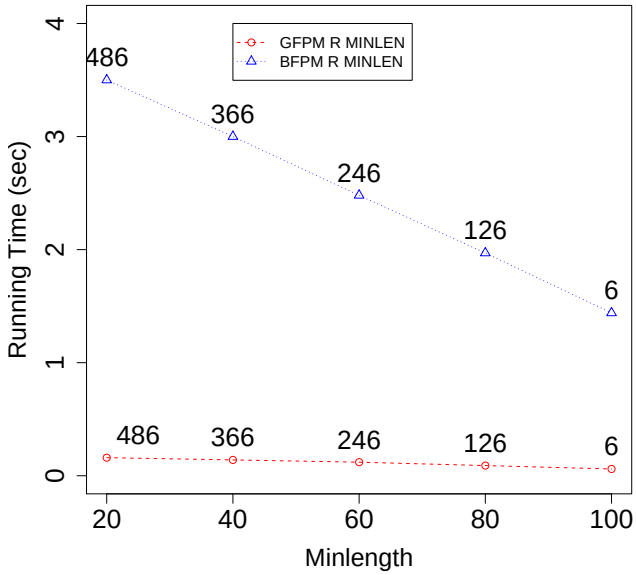

Figure 3.7: Exp 3b: The running time (and the number of patterns by mark) by algorithms **RFPM** (BFPM R) and **G-RFPM** (GFPM R) for RFPs by varying the min-len k from 20 to 100 points.

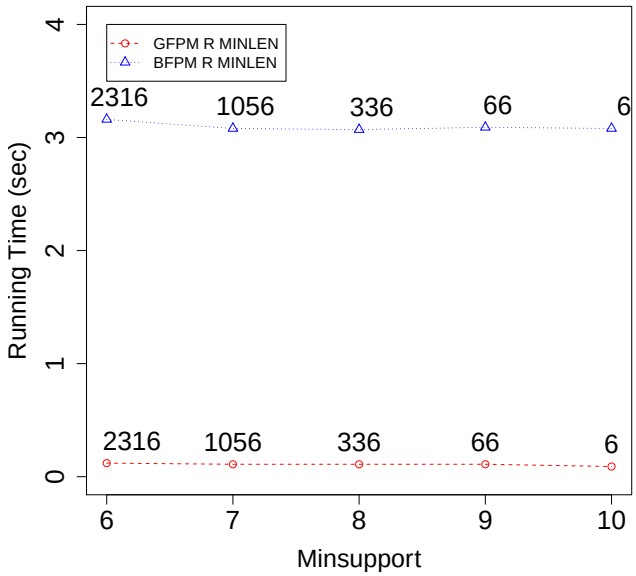

Figure 3.8: Exp 4b: The running time (and the number of patterns by mark) by algorithms **RFPM** (BFPM R) and **G-RFPM** (GFPM R) for RFPs by varying the min-sup m from 7 to 10.

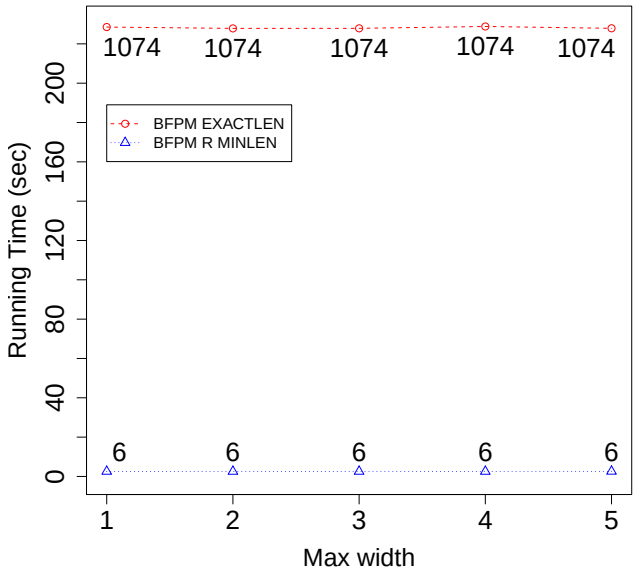

Figure 3.9: Exp 5a: The running time (and the number of patterns by mark) by algorithm **FPM** (BFPM) for FPs and **RFPM** (BFPM R) for RFPs by varying the number of max-width r from 1 to 5.

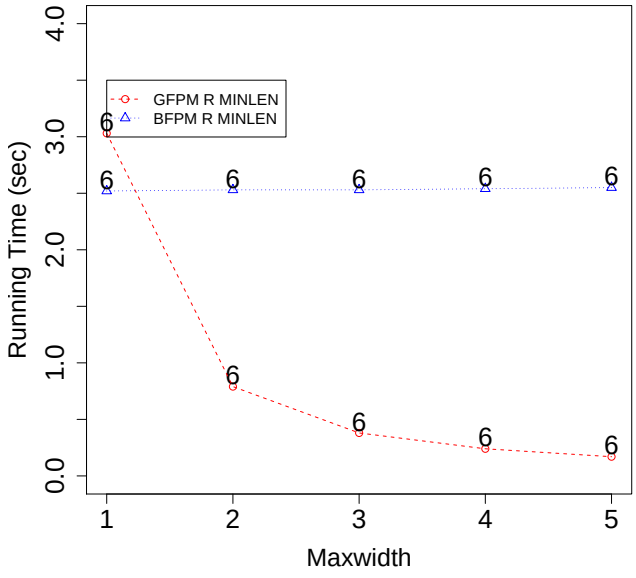

Figure 3.10: Exp 5b: The running time (and the number of patterns by mark) by algorithm **RFPM** (BFPM R) and **G-RFPM** (GFPM R) for RFPs by varying the number of max-width r from 1 to 5.

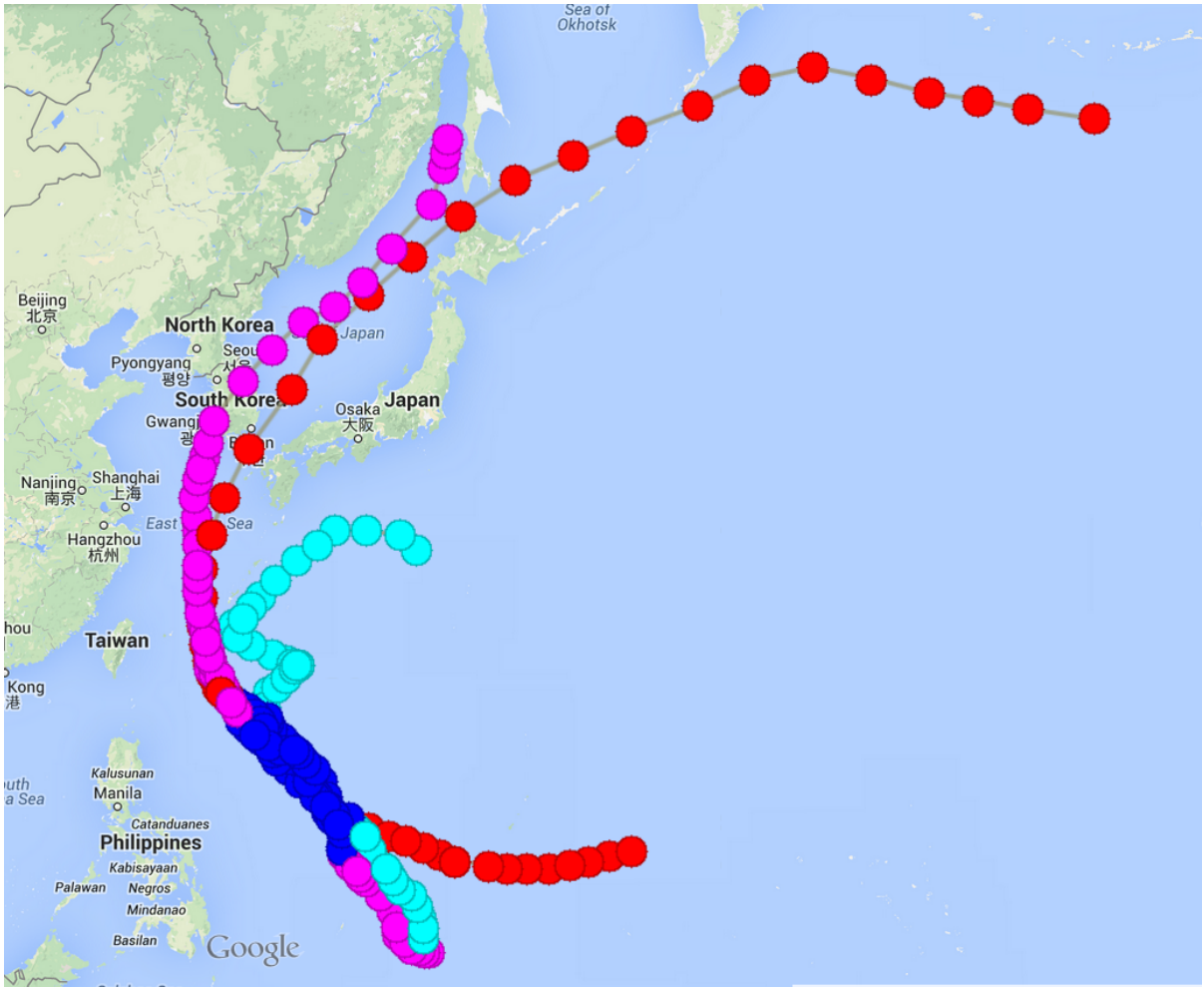

Figure 3.11: There are 3 trajectory data in the pattern P_2. The P_2 is tinted by deep blue. Its width is 2.0 latitude and length is 15 points.

Chapter4

Depth-First Mining Algorithms for Two-Sides Length-Maximal Flock Patterns

In this chapter, we present a basic mining algorithm, called **FPM-M** (Maximum Duration Flock Pattern Miner with rejection) that solves the maximal-duration flock pattern enumeration problem with minimum length constraint. In Algorithm 7, 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 A basic mining algorithm FPM-M for MFPs based on rejection

4.1.1 The first characterization of MFPs

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

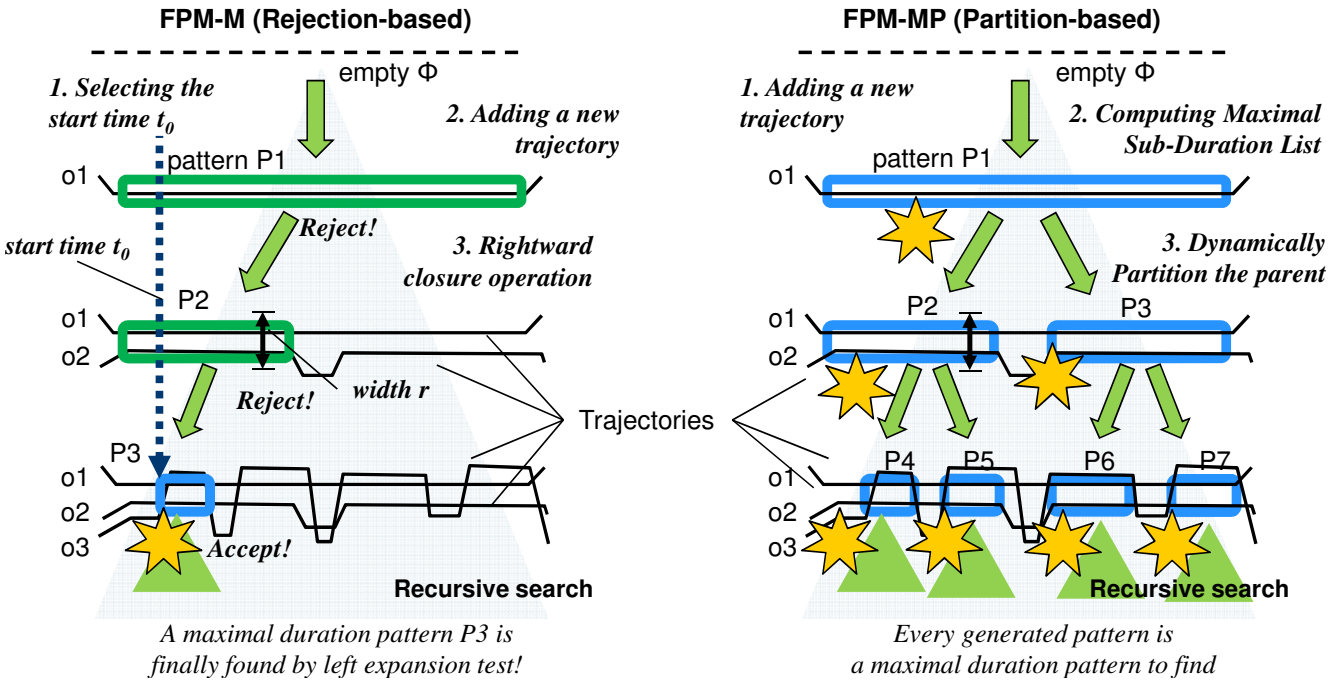

Figure 4.1: Illustration of Rejection-based approach **FPM-M** (left, Sec. 4.1) and Partition-based approach **FPM-MP** (right, Sec. 4.2). Both approaches achieve polynomial delay and space enumeration of maximal-duration flock patterns, but have different complexities.

for maximal-duration flock patterns.

Definition 11 (rightward closure)**.** For any number $r > 0$, the *r-rightward closure* of an r-flock pattern P is the unique flock pattern $\mathbf{RClosure}(P, S, r) = Q$ 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 $|| S[P.set][t] ||_\infty \leq 2r$ for every $t \in [P.start, e]$.

Algorithm 6 below computes $Q = \mathbf{RClosure}(P, S, r)$ in $O(dmk) = O(dmT)$ time,

where $m = |P.set|$ and $k = len(P)$.

Algorithm 6 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: $t \leftarrow t_0$;
3: **while** $|| S[X][t] ||_\infty \leq r$ **do** $t \leftarrow t + 1$;
4: $b \leftarrow t_0$; $e \leftarrow t - 1$;
5: **return** $P_{max}(X, [b, e])$;
6: **end procedure**

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

Proof. Let $k = len(P.span)$. We maintain the k-vector of MBBs defined by $envelope_S(P) = (MBB(S[P.set][t]))_{t \in P.span}$. 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, we introduce a *one-sided version* of maximal-duration r-flock patterns, r-RFPs, as a theoretical tool for analyzing the enumeration of MFPs.

Definition 12. 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

Algorithm 7 The FPM-M 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 $\min(X)$.

```
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:         end while
8:     end for
9: end procedure
```

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 $|| S[X][t] ||_\infty = \infty$ for any time out of $\mathbb{T} = [1, T]$, i.e., $t < 1$ or $t > T$.

Lemma 10 (characterization of $\mathcal{RM}_r$)**.** *An r-flock pattern P is an r-RFP in S if and only if $P =$ **RClosure**(P, S, r).*

Algorithm 8 RecFPM-M algorithm for finding all maximal-duration r-flock patterns recursively.

procedure RecFPM-M$(P = (\{i\}, [b, *]), ID, S, r, k_{min})$

$P_* \leftarrow$ **RClosure**(P, S, r);

if $len(P_*) < k_{min}$ **then return** ; ▷ backtrack

if $|| S[X][b-1] ||_\infty > r$ **then** ▷ maximality test

output P_*;

end if

$ID_1 \leftarrow ID$;

while $ID_1 \neq \emptyset$ **do**

$i \leftarrow deletemin(ID_1)$;

Create Q; $Q.set \leftarrow P_*.set \cup \{i\}$;

$Q.start \leftarrow P.start$; $Q.end \leftarrow P_*.end$;

RecFPM-M(Q, ID_1, S, r, k_{min}); ▷ recursive call

end while

end procedure

Lemma 11 (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. *$|| S[P.set][P.start-1] ||_\infty > 2r$. (leftward extension test (*))*

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 $I = [P.start, P.end]$,

ii $w_t > 2r$ at $t = P.start - 1$, and

iii $w_t > 2r$ 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.

□

4.1.2 The family forest for MFPs

From the above Lemma 11, 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 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-RFP without duplicates. We associate the parent $\mathcal{P}(Q)$ to any r-RFP Q in S as follows.

Definition 13 (parent for $\mathcal{RFP}_r$)**.** Let Q be any r-RFP such that $|Q.set| \geq 2$. Then, we define the parent of Q, denoted by $\mathcal{P}(Q)$, by $\mathcal{P}(Q) \stackrel{\triangle}{=} \textbf{RClosure}(R)$ 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 12. *For any r-RFP Q such that $|Q.set| \geq 2$,*

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-RFP in S. Now, we introduce a directed graph $\mathcal{F}^{\mathrm{rm}} = (\mathcal{V}, \mathcal{P}, \mathcal{I})$, called the *family forest* for $\mathcal{RFP}_r$, where:

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

Lemma 13 (family forest for $\mathcal{RFP}_r$)**.** *The* family forest $\mathcal{F}^{\mathrm{rm}}$ *is a spanning forest over the class $\mathcal{RFP}_r$ of all r-RFP in S.*

Proof. We will show that the directed graph $\mathcal{T}^{\mathrm{rm}}$ is connected and acyclic. From condition (iii) of Lemma 12 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 $\mathcal{T}^{\mathrm{rm}}$. Thus, $\mathcal{T}^{\mathrm{rm}}$ is acyclic and contains every maximal pattern in $\mathcal{RFP}_r$ as its node. This completes the proof. □

4.1.3 The algorithm

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 $i > \max(P.set)$ [96]. Using the tail expansion, it is well-known that we can systematically generate all subsets X of a set ID in backtracking [90, 96]. The following lemma answers the above question.

Lemma 14 (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 $Q =$ **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 7. In order to give the estimation of an upper bound of the amortized delay of Algorithm 7, 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 15. *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 4. (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 7 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 13 and Lemma 14, we can easily see that the algorithm **FPM-M** visits all r-RFP in S without duplicates at Line 2. From Lemma 11, the algorithm correctly outputs only and all maximal r-flock patterns at Line 10. 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 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 15 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. □

4.2 An improved algorithm FPM-MP for MFP based on partitioning

In this section, 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.

4.2.1 A problem with the previous algorithm

The basic algorithm **FPM-M** in Algorithm 7 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 11 (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.

4.2.2 The second characterization of MFPs

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 14 (left and rightward closure)**.** Let P be a r-flock pattern. The r-left-right-closure of P is the unique flock pattern $\mathbf{LRClosure}(P, S, r) = Q$ 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 $|| S[Q.set][t] ||_\infty \leq 2r$ for every $t \in [Q.start, Q.end]$.

Lemma 16. *$Q = \boldsymbol{LRClosure}(P, S, r)$ always exists, is unique, and can be computed in $O(mk_*)$ time, where $m = |P.set| = |Q.set|$ and $k_* = |Q.span|$.*

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

Definition 15. Let Q be any maximal-duration flock pattern in S. Then, we define the *parent* of Q, denoted $\mathcal{P}(Q)$, by $\mathcal{P}(Q) \triangleq \mathbf{LRClosure}(Q - \{i_{\max}\})$, 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 $\mathcal{T}^{\mathrm{rm}} = (\mathcal{V}, \mathcal{P}, \mathcal{I})$, 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 13 in Sec. 4.1, we have the following lemma for the family forest.

Lemma 17 (family forest for $\mathcal{M}_r$)**.** *The* family forest $\mathcal{F}^{\mathrm{m}}$ *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 width. A sub-duration $I \subseteq P.span$ is said to have width r if $|| S[P.span][t] ||_\infty \leq r$ 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 16 (maximal sub-duration list)**.** The maximal r-sub-duration list for P, denoted by $\mathcal{MSD}(P, S, r)$, is a sorted list $\mathcal{M} = (I_1, \ldots, I_k)$, $I_1 < \cdots < I_k$, of non-empty sub-interval of $\mathbb{T} = [1, T]$ in the preceding order such that

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

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

4.2.3 The algorithm

In Algorithm 10, we show the algorithm for computes $\mathcal{MSD}(P, S, r)$. From the construction of this algorithm, we can show the following lemma.

Lemma 18. *For any flock pattern* P *with arbitrary width, the set* $\mathcal{MSD}(P, S, r)$ *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 19 (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 9. 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 19.

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

Theorem 5 (Main result)**.** *For any trajectory database* S*, number* $r > 0$*, and dimension* $d \geq 1$*, the algorithm* ***FPM-MP*** *shown in Algorithm 9 finds all maximal-duration r-flock patterns with length at least* k_{min} *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 17 and Lemma 19. The analyzes of time and space complexities can be done in a similar manner as in Theorem 5 except that it now contains no anti-monotone pruning , and Lemma 19 makes it possible to directly generate maximal-duration patterns from its parent maximal-duration pattern. The work at the parent is $O(k_*)$ incremental time if we economically maintain the list of MBBs of all sections of P in S in $O(k_*)$ space. Hence, the result follows.

□

4.3 Speeding-up by spatial index

In this section, we present a revised algorithm **FPM-MG** based on a speed-up technique using a spatial index.

In preprocessing of **FPM-M**, we prepare spatial indices for T subsets $U_1, \ldots, U_T$ of points in S, where for every $t \in \mathbb{T}$, the index U_i stores all points in the t-th cross section $S[t] = \{\, s_i[t] \,|\, i \in ID \,\}$ of a database.

Our speed-up technique using this index is described as follows. Suppose that we start the top-level search in **FPM-M** of Algorithm 7. Selecting a start time $t \in \mathbb{T}$ and an initial ID $i \in ID$, we start the search with initial pattern $P_0 = (\{i_0\}, [t, *])$ over the candidate ID set $ID_0 = ID$.

During the depth-first search of **RecFPM-M**, we generate a new child pattern $Q = P \cup \{i\}$ by tail expansion of a parent P with a new ID $i > \max(P)$ $(i \in ID)$. In trajectory mining, the expanded flock pattern Q must satisfy an extra constraint: the width of Q must not exceed maximum width $w = 2r$ (**).

By this geometric constraint (**), we can narrow the range of candidates in ID_0 as follows. Here we use the definition of U.Range(R) in section 2.2.3.

Lemma 20. *For the algorithm **RecFPM-M** to find all $\mathcal{MFP}_r$ in S without duplicates, it is sufficient for the tail expansion $Q = P \cup \{i\}$ to add only ID $i \in ID$ satisfying conditions (i) and (ii) below:*

- *(i)* $i \in U.Range(R_{c,r})$.
- *(ii)* $i > i_0$.

To implement this mechanism, after selecting t and i, the top-level procedure **FPM-**

M computes the subset

$$ID_0 \leftarrow \{\, i \in U.Range(R_{c,r}) \,|\, i > i_0 \,\} \tag{4.1}$$

of ID, and uses this subset in the recursive procedure **RecFPM-M**. We denote by **FPM-MG** the revised version of **FPM-M** augmented with this speed-up techniques.

4.4 Experiments

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

4.4.1 Data

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 width r_* are implanted. We show the default values in Table 4.1.

4.4.2 Methods

In the experiments, we implemented the following programs 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-MG** and **BFE** used a spatial/geometric index implemented in C++ inside.

Table 4.1: 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	number	h	6
pattern	length	k_*	50
	width	r_*	1.0
	subset size	m_*	5
Mining	min. len	k	40
pattern	width	r	1.0
	subset size	m	5

- **BFE**: A previous algorithm for FP based on breadth-first search for sequences of r-disks (Vieira *et al.* [91]).
- **FPM-E**: the naive depth-first algorithm for FP using exhaustive search over FPs with length $k, k+1, \ldots, T$.
- **FPM-M**: the proposed basic depth-first algorithm for MFP (Sec. 7).
- **FPM-MP**: the proposed improved depth-first algorithm for MFP (Sec. 9).
- **FPM-MG**: the proposed improved version for MFP augmented with a practical speed-up technique using geometric index (Sec. 4.3).

From Theorem 4 and Theorem 5, 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, in the sense of entity subset, **BFE** [91] has a possibility that it may output the same entity set, i.e., our flock patterns, more than once. [1]

As an experimental environment, for all experiments except Fig. 4.5, we used a PC with Intel Core i5, 1.7GHz, Memory 4GB running Mac OS X, ver.10.9.2. For experiments of Fig. 4.5, 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.

4.4.3 Results

From Fig. 4.2 to Fig. 4.5, 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. 4.5, 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.

[1] Precisely speaking, as Theorem 1 of [91] 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 *semantics*. 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. [90, 97]

4.5 Discussion

In this chapter, 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 $\mathcal{MFP}_r$ based on the characterizations. Overall, the proposed algorithms seem theoretically as well as practically efficient solutions for mining maximal-duration flock patterns from large trajectories. For future work, we plan evaluation of the proposed algorithms on real data sets.

Algorithm 9 The FPM-MP algorithm for finding all r-flock patterns with minimum length k_{min} in a 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-MP**$(ID = \{1, \ldots, n\}, S, r, k_{min})$
2: **while** $ID \neq \emptyset$ **do** ▷ each id in ID
3: $i \leftarrow deletemin(ID)$;
4: $P \leftarrow (\{i\}, dom(s_i))$; ▷ initial maximal pattern
5: **RecFPM-MP**(P, ID, S, r, k_{min});
6: **end while**
7: **end procedure**
8: **procedure RecFPM-MP**(P, ID, S, r, k_{min})
9: **if** $len(P_*) < k_{min}$ **then return** ; ▷ backtrack
10: **output** P;
11: $ID_1 \leftarrow ID$;
12: **while** $ID_1 \neq \emptyset$ **do** ▷ first loop
13: $i \leftarrow deletemin(ID_1)$;
14: Create Q; $Q.set \leftarrow P.set \cup \{i\}$;
15: $Q.start \leftarrow P.start$; $Q.end \leftarrow P.end$;
16: $\mathcal{MSD} \leftarrow$ **MaxSubDuration**(Q, S, r); ▷ all maximal sub-durations
17: **for each** $maxspan \in \mathcal{MSD}$ **do** ▷ second loop
18: Create R; $R.set \leftarrow Q.set$;
19: $R.start \leftarrow maxspan.start$; $R.end \leftarrow maxspan.end$;
20: **RecFPM-MP**(R, ID_1, S, r, k_{min}); ▷ recursive call
21: **end for**
22: **end while**
23: **end procedure**

Algorithm 10 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: $\mathcal{MSD} \leftarrow \emptyset$;
3: $b \leftarrow Q.start$;
4: **while** $b \leq Q.end$ **do**
5: Increment b while $|| S[X][b] ||_\infty > r$; ▷ skip inter-duration time
6: $e \leftarrow b$;
7: Increment e while $|| S[X][e] ||_\infty \leq r$; ▷ expand a duration rightwards
8: $\mathcal{MSD} \leftarrow \mathcal{MSD} \cup \{[b, e]\}$;
9: $b \leftarrow e + 1$;
10: **end while**
11: **return** $\mathcal{MSD}$;
12: **end procedure**

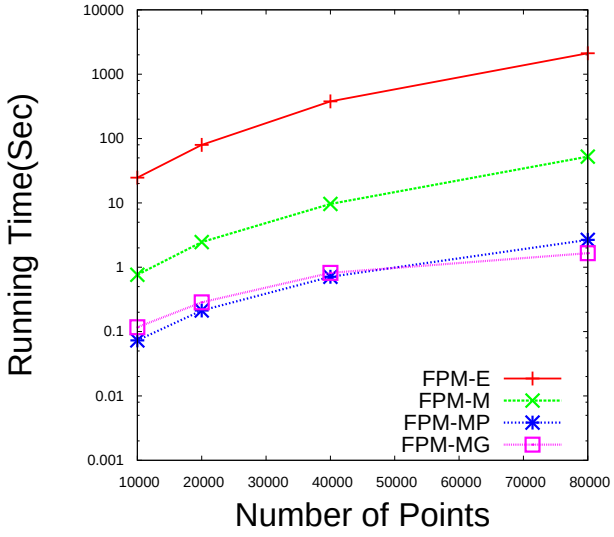

Figure 4.2: Exp 1: The running time of the algorithms varying the number of input points $N = nT$.

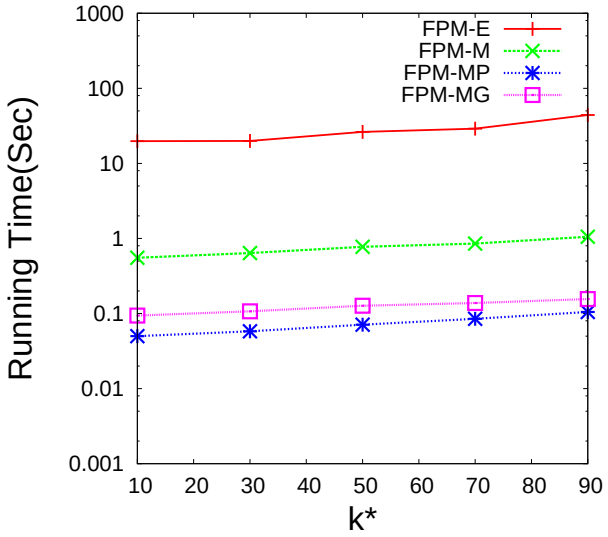

Figure 4.3: Exp 2: Running time of the algorithms varying true pattern length k_* and with fixed $k = 5$.

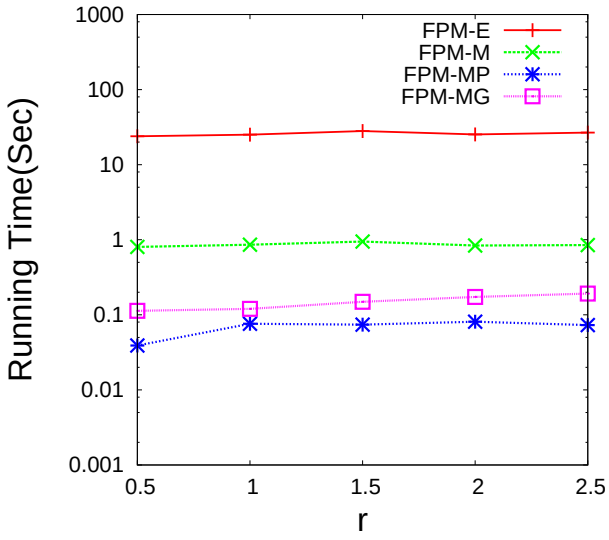

Figure 4.4: Exp 3: Running time varying the true and mining pattern widths r_* and r with $r = r_*$.

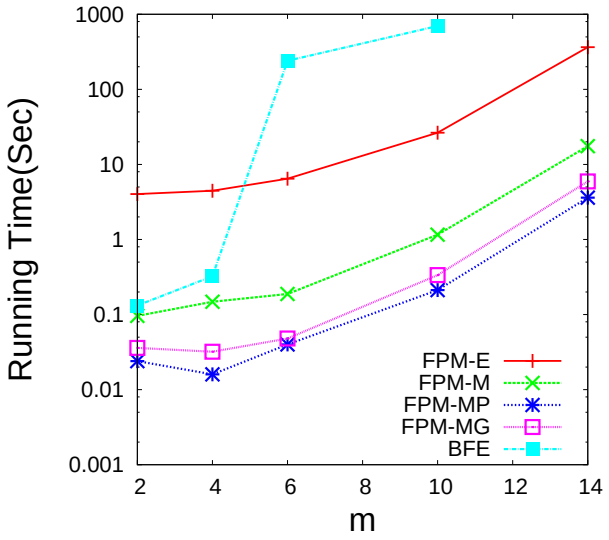

Figure 4.5: Exp 4: Running time varying the true and mining minimum subset size $m_* = m$. Only this experiment was run on a PC server (CPU Intel Xeon 3.6GHz, 32GB) since larger memory was required for BFE.

Chapter5

Table-Based Depth-First Mining Algorithm for Size-Maximal Flock Patterns

In this chapter, we present an algorithm **DFM** (Depth-First Miner) for mining a subclass of size-maximal envelope patterns, called size-maximal and rightward length-maximal flock patterns, under given constraints from an input trajectory database. In reality, the algorithm works on equivalent representation of size-maximal and rightward length-maximal flock patterns, namely, rightward length-maximal envelope patterns (REVPs). Using hash-based checking, it could find all REVPs without any duplication. Then we run experiments on a trajectory dataset to examine its basic behavior.

5.1 An algorithm DFM for RSFPs using tabulation

5.1.1 The algorithm

In Alg.11 and Alg.13, we present the main algorithm **DFM** and its recursive subprocedure **RecDFM**, respectively. The main algorithm **DFM** in Alg.11 first receives an input trajectory database S and constraint parameters $r > 0$, $k > 0$, and $m \geq 1$, and invokes the recursive subprocedure **RecDFM** in Alg.13 with each singleton ID set

Algorithm 11 The main algorithm for finding all REVPs with length $\geq k$, width $\leq r$, and frequency $\geq m$ in a database $S = \{s_1, \ldots, s_n\}$ of n trajectories of length T.

1: **procedure DFM**(S, r, k, m)
2: $\Theta \leftarrow (r, k, m)$;
3: $HM \leftarrow \emptyset$; //a hash table for envelope patterns
4: **for** $i \leftarrow 1, \ldots, n$ **do**
5: **for** $b \leftarrow 1, \ldots, T$ **do**
6: **RecDFM**$(\{i\}, b, i, n, S, \Theta, HM)$;
7: **end for**
8: **end for**
9: **end procedure**

$X = \{i\}$, where $i \in \mathbb{O}$, and other arguments.

Starting from a singleton ID set $X = \{i\}$, the subprocedure **RecDFM** in Alg.13 recursively generates all REVPs P satisfying given constraints by recursively expanding the current core set $X \subseteq \mathbb{O}$ using depth-first search approach such as Eclat algorithm. The algorithm **RecDFM** consists of the following steps.

Generation of core sets

At lines 1, 3, 4, and 13, the algorithm generates a child core set $Y \subseteq \mathbb{O}$ from the parent core set X using the depth-first search [96]. In the DFS, the algorithm starts from a singleton set $X = \{i\}$ for each $i \in \mathbb{O}$. In an iteration, the algorithm expands X by adding a new object $i \in \mathbb{O}$ such that $i > \ell = \max(X)$. This ensures enumeration of core sets in a unique way. By incrementally maintaining $\ell = \max(X)$, this step can be implemented in $O(1)$ time.

Algorithm 12 An algorithm that computes the one-sided closed envelope pattern in $\mathrm{EVP}(X, b, r)$ generated by an ID set X with start time s and maximum width r.

1: **procedure GetLongestEVP**(X, b, r)
2: $j \leftarrow 1$; $h \leftarrow b$;
3: **while** true **do**
4: $X[h] = \{\, p_h^i \mid i \in X \,\}$;
5: $\widehat{E}[j] \leftarrow$ the MBR of the set $X[h]$;
6: **if** $width(\widehat{E}[j]) > r$ **then**
7: $k \leftarrow j - 1$; $t \leftarrow h - 1$; **break**;
8: **end if**
9: $j \leftarrow j + 1$; $h \leftarrow h + 1$;
10: **end while**
11: **return** $\widehat{P} = (\langle \widehat{E}[1], \ldots, \widehat{E}[k] \rangle, b, t)$;
12: **end procedure**

Computation of the REVP and its cover sets

In Algorithm 12, we present the algorithm **GetLongestEVP** that computes the longest and most specific envelope pattern $P = \mathbf{CEVP}(X, b, r)$ for a core set X. In what follows, we denote by $\mathbf{CEVP}(X, b, r)$ the flock pattern that the procedure **GetLongestEVP**(X, b, r) returns. At line 5 of Algorithm 13, the algorithm computes $P = \mathbf{CEVP}(X, b, r)$ using the subprocedure **GetLongestEVP**. At line 8, the algorithm computes the cover set $L = \mathbf{Cov}_{\mathcal{S}}(E)$ by simply scanning all of m trajectories one by one in $O(k)$ time per trajectory. These computations take $O(km)$ time.

Checking anti-monotone constraint

At line 6, the algorithm checks if the obtained pattern P satisfies the minimum length constraint k. The next lemma says that the pruning of the current core set X with r and k is sound and does not lose any successful branches.

Lemma 21. *Let $s \in [1..n]$. Let $X_i \subseteq \mathbb{O}$ and $P_i = \boldsymbol{CEVP}(X_i, b, r)$ for every $i = 1, 2$. Then, we have:*

(a) $X_1 \subseteq X_2$ implies $width(P_1) \leq width(P_2)$.

(b) $X_1 \subseteq X_2$ implies $len(P_1) \geq len(P_2)$.

Proof. The lemma follows from the property that $X_1 \subseteq X_2$ implies $EVP_{\mathcal{S}}(X_1) \sqsubseteq EVP_{\mathcal{S}}(X_2)$. □

Checking monotone constraint and duplicate detection

At line 9, the algorithm checks the frequency constraint m of P and the duplicate detection for P. Since two distinct core sets X_1, X_2 can generate the same EVP P, explicit test for duplicates is required by using a hash table HM that stores all discovered EVPs P with the unique key $(start(P), \mathbf{Cov}_{\mathcal{S}}(P))$ for P. Note that the above tests cannot prune the descendants since these constraints are not anti-monotone unlike those for r and k.

5.1.2 Conversion of REVPs to RSFPs

According to lemma 4, we could compute size-maximal flock patterns from EVPs. Algorithm 14 shows the process of the computation. The second line and 3th line show the initialization for ID set I and trajectory set X'. From the 4th line to the 6th line,

Algorithm 13 A DFS algorithm that recursively searches for REVPs with start time b in $\mathcal{CEVP}(b)$

```
1: procedure RecDFM(X, b, ℓ, m, S, Θ, HM)
2:     (r, k, m, Δ) ← Θ;
3:     for i ← ℓ + 1, ..., m do
4:         Y ← X ∪ {i}; //Expanding a core set.
5:         P ← GetLongestEVP(Y, b, r);
6:         if P.length < k then
7:             continue; //Prune descendants by k
8:         L ← Cov_S(P);
9:         if |L| ≥ m and HM[(s, L)] = ⊥ then
10:            Output P as an answer;
11:            HM[(s, L)] ← P;
12:        end if
13:        RecDFM(Y, s, i, m, S, Θ, HM);
14:    end for
15: end procedure
```

the algorithm computes range query from R_b to R_e to get the trajectory ID sets. Then it computes the intersection between the last result ID set I and the range query result ID set computed this time. From the 7th line to the 9th line, the algorithm gets all trajectories from database S by using the ID set I. Finally, it outputs the result $(X', [b, e])$.

5.2 Analysis

Combining the above arguments, we give the correctness and the complexity of our main algorithm. Let M be the *output size*, i.e., the number of REVPs in S as solutions and N be the number of all core sets X that **RecDFM** examined. Clearly, $M \leq N \leq T2^n$.

Theorem 6 (main theorem)**.** *The algorithm **DFM** of Alg. 11 correctly finds all s* $P \in \mathcal{CEVP}$ *in* S *such that* $width(P) \leq r$*,* $len(P) \geq k$*, and* $freq(P) \geq m$ *without duplicates in* $O(kmN)$ *time and* $O(kM)$ *space.*

Unfortunately, **DFM** is not a depth-first search algorithm in strict sense as in [96] since we cannot make complete pruning by length constraint and duplicate detection. To cope with this, the algorithm uses a hash table *HM* to avoid duplicates, but it still generates N candidate core sets. Due to this fact, the algorithm does not have output-polynomial time complexity. Furthermore, since the hash table stores M entries, it may require exponential space in the worst case.

5.3 Experiments

Finally, we report preliminary computational experiments to examine basic properties of the proposed algorithm **DFM** on real trajectory dataset.

Algorithm 14 The algorithm that computes size-maximal flock pattern $P' = (X', [b, e])$ from EVP $P = (E, b, e)$ in database S, where $E = \langle R_b, \ldots, R_e \rangle$ and $freq_{\mathcal{S}}(P) > 0$.

1: **procedure** EVPTOFP(P, S)
2: $I \leftarrow ID(S)$ ▷ I is a trajectory ID set
3: $X' \leftarrow \phi$ ▷ X' is a trajectory set
4: **for** $i \leftarrow b, \ldots, e$ **do**
5: $I \leftarrow U.Range(R_i) \bigcap I$
6: **end for**
7: **for all** $i \in I$ **do**
8: $X' \leftarrow X' \bigcup \{s | s_i \in S\}$
9: **end for**
10: **Output:** $(X', [b, e])$
11: **end procedure**

5.3.1 Data

We used GPS trajectory dataset collected in Microsoft Research Asia, *GeoLife* project, consisting of 12,034 GPS-trajectories obtained from 165 pedestrians in 30 cities including Beijing from April 2007 to August 2009, containing over 18 million points in 795MB. As test data, we selected a small subset consisting of 80 trajectories with average length from 3 to 4 points, whose total size are 316 points in 14KB.

5.3.2 Methods

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

5.3.3 Results

In Fig. 5.1, we show the result on the running time (left) and memory usage (right) by varying the input size. We observed that the running time increases as the input size increases, and we did not observe any clear dependency of memory usage on the input size. For example, the algorithm took 130 sec for finding six patterns in 80 trajectories. Hence, the current implementation is quite slow, and will take much time for large data sets.

In Fig. 5.2, Fig. 5.3, and Fig. 5.4, we show the results on the running time (left) and the number of solutions (right) by varying the minimum frequency m, the maximum width r, and the minimum length k, respectively. In these figures, we first saw that the algorithm discovered from zero to six patterns in the data set depending on the values of parameters m, r, and k as expected. In Fig. 5.3 and Fig. 5.4 with varying r, and k, we observed that the running time is almost proportional to the number of solutions. In

Fig. 5.2 with varying m, we did not observed such dependency.

5.4 Discussion

The above behavior of our algorithm was similar to other transaction data mining algorithms [66, 96]. By inspection, a large potion of running time was consumed in computing cover sets and duplicate detection, which is partly a reason of the large computation time of the current implementation of our algorithm **DFM**.

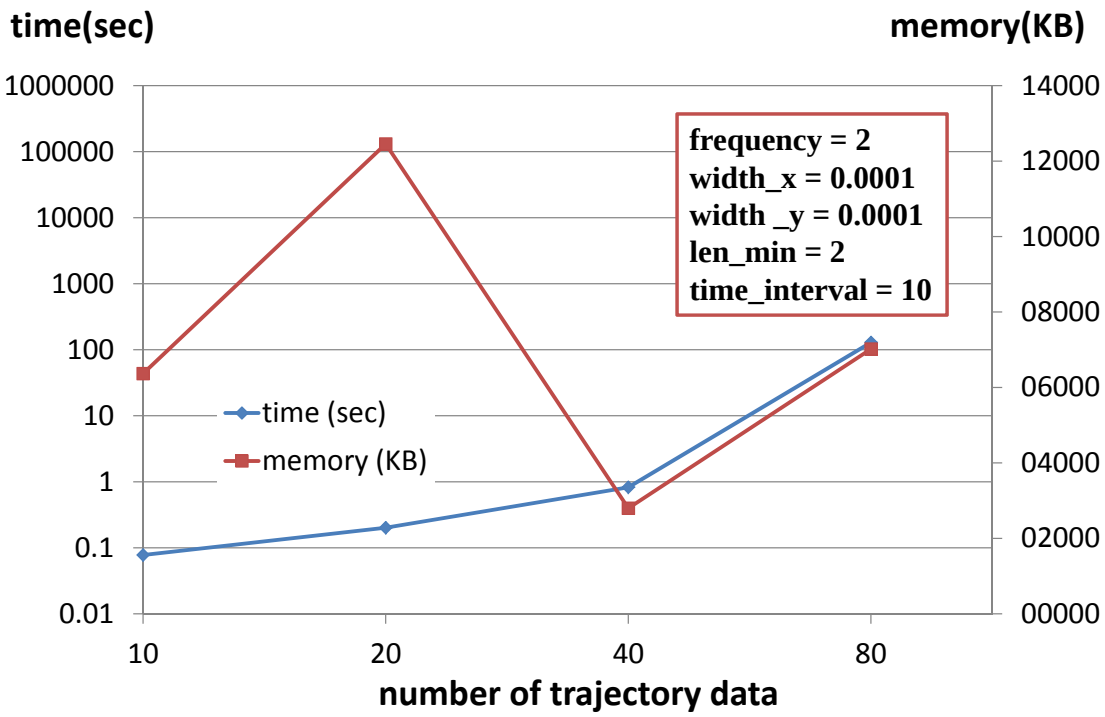

Figure 5.1: Running time and memory against input size

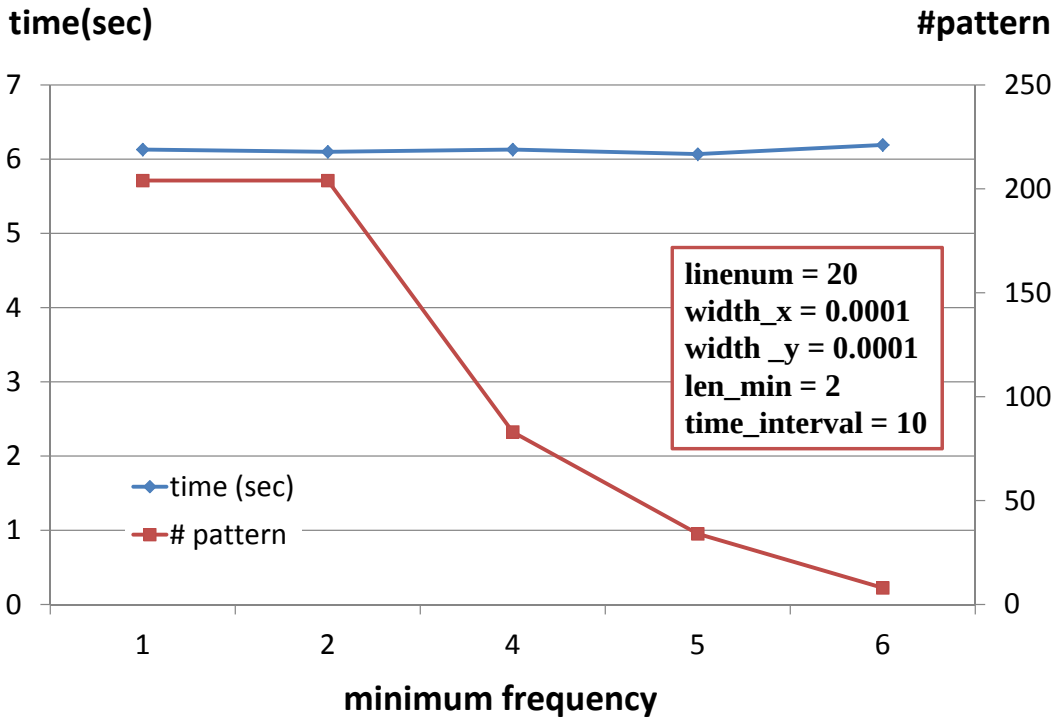

Figure 5.2: Running time against minimum frequency

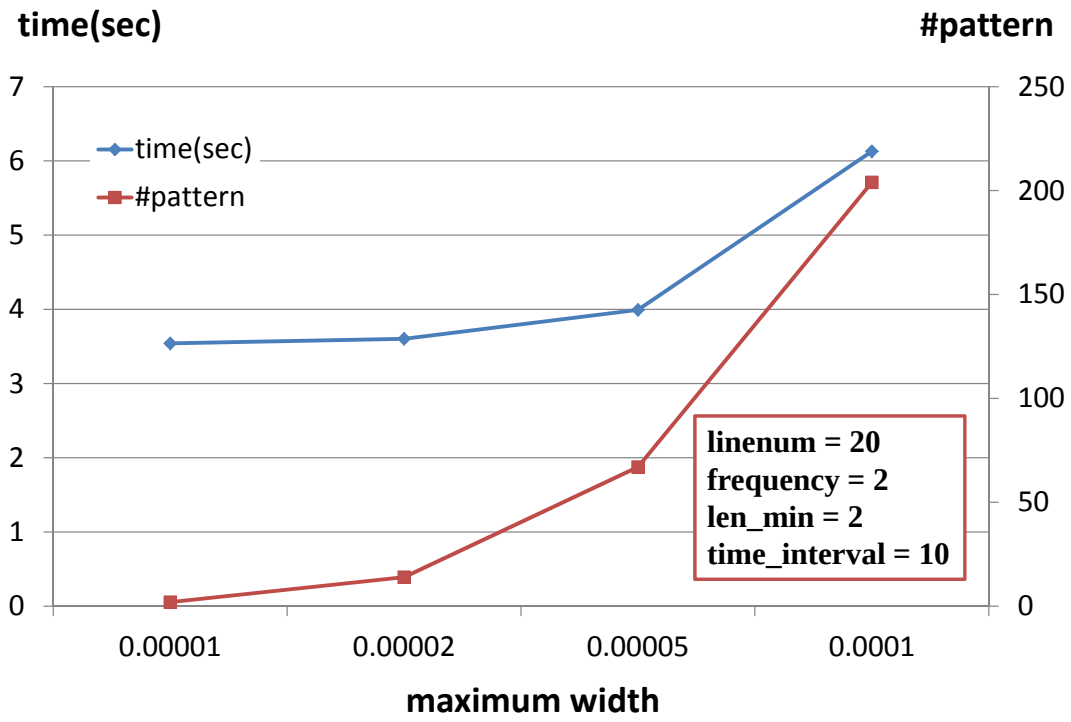

Figure 5.3: Running time against maximum width

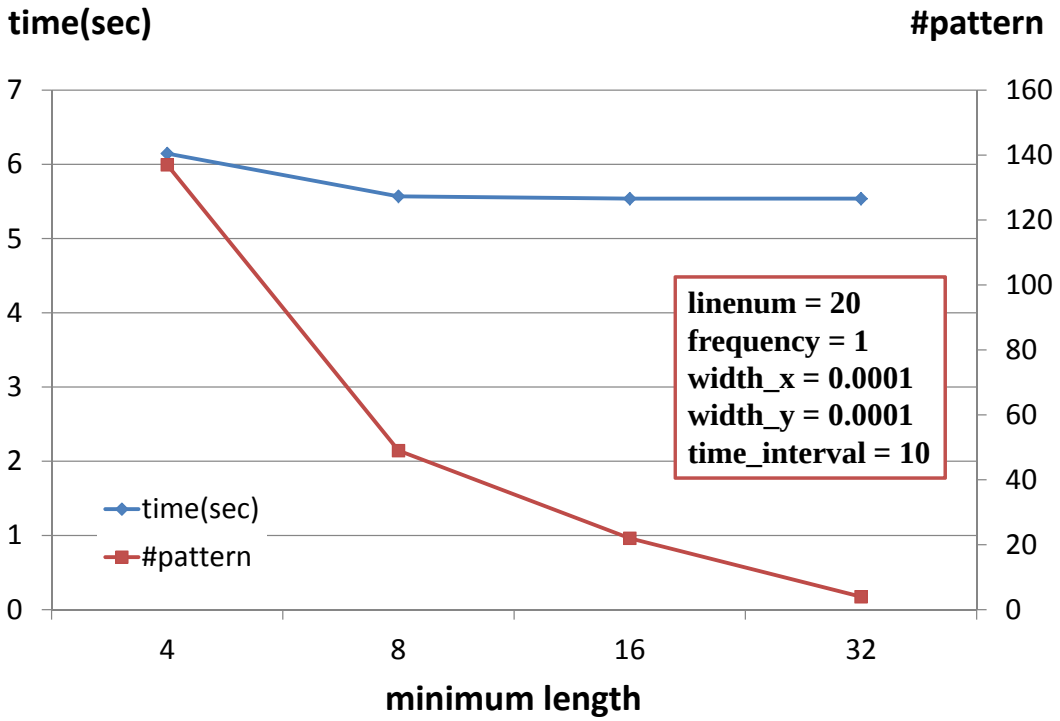

Figure 5.4: Running time against minimum length

Chapter6

Conclusion

6.1 Summary of the results

In this thesis, we presented the definitions of basic notions about trajectory data. And we defined problems of mining trajectory data. We would like to find useful information for traffic, sightseeing and any field which is related with geography information by mining trajectory data. Then we gave algorithms that solve these problems. These algorithms could find the patterns efficiently without redundant and duplicate patterns. The real data experiments showed that our trajectory data study is useful and interesting.

In Chapter 3, we focused on pattern-growth approach for the problem of finding all (r, k)-flock patterns ((r, k)-FPs), where our purpose is to make complete mining of all patterns that satisfy a given constraint in an input database. For the purpose, we developed our algorithm FPM for completing mining of FPs [10], the first pure pattern-growth style algorithm. Particularly, this chapter focuses on two extensions of FPM, called RFPM and FPM-MG. The former RFPM finds the class of, called *rightward length-maximal flock patterns* (RFP), while the latter FPM-MG uses *speed-up technique using a geometric index* [10]. We implemented the basic and the improved algorithms above based on pattern-growth mining approach. To evaluate these extensions, we then ran

experiments on implanted synthesis datasets. Experiments demonstrate that both of extensions significantly improve on the efficiency of the original algorithm FPM in a wide range of parameter settings. We also ran the real data experiment on typhoon trajectory data. According to the completeness of the algorithm FPM-MG and the result that there are 10 RFPs found from all combinations of all trajectory data,we could affirm that there are only 10 RFPs which the width is no more than 200km and the length is longer than 90 hours. Since FPM-MG could find all RFPs from 320 typhoon trajectory data, it is useful for real trajectory data.

In Chapter 4, we presented a variety of depth-first mining algorithms for the maximal-duration r-flock pattern enumeration problem. The maximal-duration r-flock pattern is two side extension longest pattern. Thus, shorter patterns could not be included in the longest patterns. As a result, the number of patterns is fewer. We introduced the algorithm FPM-MP. It finds all maximal-duration r-flock patterns (r-MFPs) in $O(dkn)$ time per pattern P using $O(dm^2)$ additional space. We also showed that a modified algorithm FPM-MG with practical speed-up technique using spatial indexes. 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 could extend the above results to L_2-metric for any fixed dimension $d = O(1)$. 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 pro-

posed algorithms and the previous flock pattern mining algorithm BFE (Vieira et al. [91]) 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.

In Chapter 5, we introduced the rightward size-maximal flock patterns from trajectory data and studied a mining problem for the class rightward size-maximal and length-maximal flock patterns (RSFPs). We also introduced its equivalent representation, called rightward length-maximal envelope patterns (REVP).

An REVP is composed of a list of minimum bounding rectangles and a time interval. We could compute RSFPs from REVPs. And REVP is convenient to find size-maximal flock patterns. With the definition of REVP, there could be potentially infinitely many similar patterns for the continuous nature of space domain. To overcome this problem, we presented a procedure that recovers a rightward longest, and most-specific EVP from the current candidate of its cover set. As a main result, we presented an efficient algorithm DFM that, given maximum width θ, minimum length ℓ, and minimum frequency σ, finds all length-maximal REVPs with width $\leq \theta$, length $\geq \ell$, and frequency $\geq \sigma$ in a trajectory database. In addition, we presented an algorithm that converts a discovered REVP into the equivalent RSFP. Consequently, the algorithm combines depth-first search, efficient closure computation, and hash-based duplication check to achieve complete mining of all RSFPs.

6.2 Discussion and future researches

We proposed flock pattern and mining algorithms on trajectory datasets. However, there are many more applications in traditional data mining. Therefore it would be interesting to present another applications for trajectory data mining.

Until now, our algorithms require that the length and the time interval of trajectory data in database should be same. We plan to improve our algorithms to reduce the preprocess on trajectory data. Then we could run our algorithms on real trajectory data of GPS sensor or others. We also plan to extend our algorithms to process three dimension or high dimension data.

To get more useful information for sightseeing, shopping and traffic management, we would study new pattern on trajectory data. For example, we are also interested in the trajectory data which perform the similar form. Since many trajectory data comes from GPS sensors or weather radars, we would design on-line algorithms in order to find patterns from real-time data. Especially, we could add the temporal property to new pattern.

In order to develop more applications, we plan to run our algorithms on real trajectory data sets. For big size data sets, we would like to design parallel algorithms to speed up. Or we could use cloud server such as Hadoop, to speed up our algorithms. In addition, we plan to run our algorithms through spatial database. Spatial database always supports spatial index and range query. This would simplify our algorithms.

We also plan to extend our flock pattern research. We have studied spatial feature of flock pattern. As trajectory data is temporal-spatial data, we would focus on the temporal feature of flock pattern. For example, we could care about the time when flock patterns form.

Bibliography

[1] Ramesh C. Agarwal, Charu C. Aggarwal, and V.V.V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of parallel and Distributed Computing*, 61(3):350–371, 2001.

[2] Charu C. Aggarwal and Philip S. Yu. A new framework for itemset generation. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 18–24. ACM, 1998.

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[4] Rakesh Agrawal, Manish Mehta, John C. Shafer, Ramakrishnan Srikant, Andreas Arning, and Toni Bollinger. The quest data mining system. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, volume 96, pages 244–249, 1996.

[5] Rakesh Agrawal and John C Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, 1996.

[6] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference Very Large Data Bases (VLDB'94)*, volume 1215, pages 487–499, 1994.

[7] Khalil M. Ahmed, Nagwa M. El-Makky, and Yousry Taha. A note on beyond market baskets: Generalizing association rules to correlations. *ACM SIGKDD Explorations Newsletter*, 1(2):46–48, 2000.

[8] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12:497–528, 2007.

[9] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leadership patterns among trajectories. In *Proceedings of the 2007 ACM symposium on Applied computing (SAC'07)*, pages 3–7. ACM, 2007.

[10] Hiroki Arimura, Xiaoliang Geng, and Takeaki Uno. Efficient mining of length-maximal flock patterns from large trajectory data. Manuscript, DCS, IST, Hokkaido University, March 2013. `http://www-ikn.ist.hokudai.ac.jp/~arim/papers/flockpattern201303.pdf`.

[11] Hiroki Arimura and Takeaki Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In *Proceedings of the SIAM International Conference on Data Mining (SDM'09)*, pages 1087–1098, 2009.

[12] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1993.

[13] Petko Bakalov, Marios Hadjieleftheriou, Eamonn Keogh, and Vassilis J. Tsotras. Efficient trajectory joins using symbolic representations. In *Proceedings of the 6th International Conference on Mobile Data Management (MDM'05)*, pages 86–93. ACM, 2005.

[14] Petko Bakalov, Marios Hadjieleftheriou, and Vassilis J. Tsotras. Time relaxed spatiotem-

poral trajectory joins. In *Proceedings of the 2005 ACM International Symposium on Advances in Geographic Information Systems (GIS'05)*, pages 182–191. ACM, 2005.

[15] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *ACM SIGMOD Record*, volume 27, pages 85–93. ACM, 1998.

[16] Marc Benkert, Joachim Gudmundsson, Florian Hubner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(11):111–125, 2008.

[17] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. *ACM SIGMOD Record*, 26(2):265–276, 1997.

[18] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD Record*, volume 26, pages 255–264. ACM, 1997.

[19] Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. 8037:219–230, 2013.

[20] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings 17th International Conference on Data Engineering (ICDE'01)*, pages 443–452. IEEE, 2001.

[21] Huiping Cao, Nikos Mamoulis, and D.W. Cheung. Discovery of periodic patterns in spatiotemporal sequences. *Knowledge and Data Engineering*, 19:453–467, 2007.

[22] V. Prasad Chakka, Adam C. Everspaugh, and Jignesh M. Patel. Indexing large trajectory data sets with seti. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, 2003.

[23] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a

database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[24] Yun Chen and Jignesh M. patel. Design and evaluation of trajectory join algorithms. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'09)*, pages 266–275. ACM, 2009.

[25] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *IEEE 27th International Conference on Data Engineering (ICDE'11)*, pages 900–911. IEEE, 2011.

[26] David W. Cheung, Jiawei Han, Vincent T. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering (ICDE'96)*, pages 106–114. IEEE, 1996.

[27] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[28] Xiaoliang Geng, Hiroki Arimura, and Takeaki Uno. Pattern mining from trajectory gps data. In *Proceedings of the 3rd IIAI International Conference on e-Services and Knowledge Management (IIAI ESKM'12)*, pages 60–65. CPS, 2012.

[29] Xiaoliang Geng, Hiroki Arimura, and Takeaki Uno. Trajectory pattern mining in practice: Algorithms for mining flock patterns from trajectories. In *Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval (IC3K KDIR'13)*, pages 143–151, 2013.

[30] Xiaoliang Geng, Hiroki Arimura, and Takeaki Uno. Enumeration of complete set of flock patterns in trajectories. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS'14)*, pages 53–61. ACM, 2014.

[31] Xiaoliang Geng, Takeaki Uno, and Hiroki Arimura. Practical algorithms for mining flock patterns from trajectories. *IPSJ Journal*, 56(4):1292–1304, 2015. In Japanese.

[32] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 330–339. ACM, 2007.

[33] Bart Goethals and Mohammed J Zaki. Fimi '03: Workshop on frequent itemset mining implementations. In *Third IEEE International Conference on Data Mining Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, pages 1–13, 2003.

[34] Gösta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Proceedings of Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, volume 3, pages 123–132, 2003.

[35] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM International Symposium on Advances in Geographic Information Systems (GIS'06)*, pages 35–42, 2006.

[36] Joachim Gudmundsson, Marc. van Kreveld, and Bettina Speckmann. Efficient detection of motion patterns in spatio-temporal sets. *GeoInformatica*, 11:195–215, 2007.

[37] Marios Hadjieleftheriou, George Kollios, Petko Bakalov, and Vassilis J. Tsotras. Complex spatio-temporal pattern queries. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05)*, pages 877–888. ACM, 2005.

[38] Jiawei Han, Zhenhui Li, and Lu An Tang. Mining moving object, trajectory and traffic data. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA'10)*, volume 5982 of *LNCS*, pages 485–486. Springer-Verlag, 2010.

[39] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[40] Tomasz Imieliński, Leonid Khachiyan, and Amin Abdulghani. Cubegrades: Generalizing association rules. *Data Mining and Knowledge Discovery*, 6(3):219–257, 2002.

[41] Seung-Hyun Jeong, Norman W. Paton, Alvaro A.A. Fernandes, and Tony Griffiths. An experimental performance evaluation of spatio-temporal join strategies. *Transactions in GIS*, 9:129–156, 2005.

[42] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'08)*, pages 70–79. IEEE, 2008.

[43] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. Convoy queries in spatio-temporal databases. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'08)*, pages 1457–1459. IEEE, 2008.

[44] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19:585–602, 2010.

[45] Asanobu Kitamoto. Digital typhoon, NII. 2014. http://agora.ex.nii.ac.jp/digital-typhoon/.

[46] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups of trackable moving point objects. *Geographic Information Science (GIS'02)*, 2478:132–144, 2002.

[47] Patrick Laube, Stephan Imfeld, and Robert Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19:639–669, 2005.

[48] Patrick Laube, Stephan Imfeld, and Robert Weibel. Discovering relative motion patterns in groups of moving point objects. *Geographical Information Science (GIS'05)*, 19:639–668, 2005.

[49] Patrick Laube and Ross S. Purves. An approach to evaluating motion pattern detection techniques in spatio-temporal data. *Computers, Environment and Urban Systems*, 30:347–374, 2006.

[50] Patrick Laube, Marc van Kreveld, and Stephan Imfeld. Finding REMO — detecting relative motion patterns in geospatial lifelines. In *Proceedings of the 11th International Symposium on Spatial Data Handling (SDH'05)*, pages 201–215. Springer-Verlag, 2005.

[51] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[52] Young-Koo Lee, Won-Young Kim, Y. Dora Cai, and Jiawei Han. Comine: Efficient mining of correlated patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 581–584, 2003.

[53] Zhenhui Li, Bolin Ding, and Jiawei Han. Swarm: mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3:723–734, 2010.

[54] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 1099–1108. ACM, 2010.

[55] Guimei Liu, Hongjun Lu, Wenwu Lou, and Jeffrey Xu Yu. On computing, storing and querying frequent patterns. In *Proceedings of the ninth ACM SIGKDD International*

Conference on Knowledge Discovery and Data mining (KDD'03), pages 607–612. ACM, 2003.

[56] Junqiang Liu, Yunhe Pan, Ke Wang, and Jiawei Han. Mining frequent item sets by opportunistic projection. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'02)*, pages 229–238. ACM, 2002.

[57] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 236–245. ACM, 2004.

[58] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Effcient algorithms for discovering association rules. In *KDD-94: AAAI Workshop on Knowledge Discovery in Databases (KDD'94)*, pages 181–192, 1994.

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

[60] Mario A. Nascimento and Jefferson RO Silva. Towards historical r-trees. In *Proceedings of the 1998 ACM Symposium on Applied Computing (SAC'98)*, pages 235–240, 1998.

[61] E. Omniecinski. Alternative interest measures for mining associations. *IEEE Transactions on Knowledge and Data Engineering*, 15:57–69, 2003.

[62] Feng Pan, Gao Cong, Anthony KH Tung, Jiong Yang, and Mohammed J. Zaki. Carpenter: Finding closed patterns in long biological datasets. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'03)*, pages 637–642. ACM, 2003.

[63] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. *An effective hash-based algorithm for mining association rules*, volume 24. ACM, 1995.

[64] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. Efficient parallel data mining for association rules. In *Proceedings of the fourth International Conference on Information and Knowledge Management (ICKM'95)*, pages 31–36. ACM, 1995.

[65] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, pages 398–416. Springer, 1999.

[66] Jian Pei and Jiawei Han. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 215–224. IEEE, 2001.

[67] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 441–448. IEEE, 2001.

[68] Jian Pei, Jiawei Han, Runying Mao, et al. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues In Data Mining and Knowledge Discovery (SIGMOD'00)*, volume 4, pages 21–30, 2000.

[69] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

[70] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches to the in-

dexing of moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 395–406, 2000.

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

[72] Omar Ernesto Cabrera Rosero and Andres Oswaldo Calderon Romero. Performance analysis of flock pattern algorithms in spatio-temporal databases. In *XL Latin American Computing Conference (CLEI'14)*, pages 1–6. IEEE, 2014.

[73] Arumugam S. and Jermaine C. Closest-point-of-approach join for moving object histories. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'06)*, page 86. IEEE, 2006.

[74] Mahmoud Attia Sakr and Ralf Hartmut Güting. Spatiotemporal pattern queries. *Geoinformatica*, 15:497–540, 2011.

[75] Sunita Sarawagi, Shiby Thomas, and Rakesh Agrawal. *Integrating association rule mining with relational database systems: Alternatives and implications*, volume 27. ACM, 1998.

[76] Ashok Savasere, Edward Robert Omiecinski, and Shamkant B Navathe. An efficient algorithm for mining association rules in large databases. Technical report, 1995.

[77] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4(2-3):163–192, 2000.

[78] Šaltenis Simonas, S. Jensen Christian, T. Leutenegger Scott, and A. Lopez Mario. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pages 331–342. ACM, 2000.

[79] Dodge Somayeh, Weibel Robert, and Lautenschutz Anna-Katharina. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3–4):240–252, 2008.

[80] Zhexuan Song and Nick Roussopoulos. SEB-tree: An approach to index continuously moving objects. In *Mobile Data Management*, pages 340–344. Springer, 2003.

[81] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'02)*, pages 32–41. ACM, 2002.

[82] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04)*, pages 611–622. ACM, 2004.

[83] Yufei Tao and Dimitris Papadias. Efficient historical r-trees. In *Proceedings of Thirteenth International Conference on Scientific and Statistical Database Management (SSDBM'01)*, pages 223–232. IEEE, 2001.

[84] Yufei Tao and Dimitris Papadias. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 431–440, 2001.

[85] Yufei Tao, Dimitris Papdias, and Jimeng Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03)*, pages 790–801. ACM, 2003.

[86] Yufei Tao, Jimeng Sun, and Dimitris Papdias. Analysis of predictive spatio-temporal queries. *ACM Transactions on Database Systems*, 28:295–336, 2003.

[87] Brinkhoff Thomas, Kriegel Hans-Peter, and Seeger Bernhard. Efficient processing of spatial joins using r-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 237–246. ACM, 1993.

[88] Hannu Toivonen et al. Sampling large databases for association rules. In *Proceedings of 22th International Conference on Very Large Data Bases (VLDB'96)*, volume 96, pages 134–145, 1996.

[89] Ulanbek Turdukulov, Andres Oswaldo Calderon Romero, Otto Huisman, and Vasilios Retsios. 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.

[90] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. *Proceedings of the 7th International Conference on Discovery Science (DS'04)*, 3245:16–31, 2004.

[91] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'09)*, pages 286–295. ACM, 2009.

[92] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 236–245. ACM, 2003.

[93] Tianyi Wu, Yuguo Chen, and Jiawei Han. Re-examination of interestingness measures in pattern mining: a unified framework. *Data Mining and Knowledge Discovery*, 21(3):371–397, 2010.

[94] Xiaomei Xu, Jiawei Han, and Wei Lu. Rt-tree: an improved r-tree index structure for spatiotemporal databases. In *Proceedings of the 4th International Symposium on Spatial Data Handling (SDH'90)*, pages 1040–1049, 1990.

[95] Jiong Yang and Meng Hu. Trajpattern: Mining sequential patterns from imprecise trajectories of mobile objects. In *Proceedings of 10th International Conference on Extending Database Technology (EDBT'06)*, volume 3896 of *LNCS*, pages 664–681. Springer-Verlag, 2006.

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

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

[98] Mohammed J Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. Parallel algorithms for discovery of association rules. *Data Mining and Knowledge Discovery*, 1(4):343–373, 1997.

[99] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.

[100] Mohammed Javeed Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining (SDM'02)*, volume 2, pages 457–473. SIAM, 2002.

[101] Kai Zheng, Goce Trajcevski, Xiaofang Zhou, and Peter Scheuermann. Probabilistic range queries for uncertain trajectories on road networks. In *Proceedings of the 14th International Conference on Extending Database Technology (ICDT'11)*, pages 283–294. ACM, 2011.

[102] Yu Zheng and Xiaofang Zhou. *Computing with Spatial Trajectories*. Springer-Verlag, 2011.

[103] Panfeng Zhou, Donghui Zhang, Betty Salzberg, Gene Cooperman, and George Kollios. Close pair queries in moving object databases. In *Proceedings of the 13th annual ACM International Workshop on Geographic Information Systems (GIS'05)*, pages 2–11. ACM, 2005.