情報処理学会論文誌 Vol.56 No.4 1292–1304 (Apr. 2015)

推薦論文

大規模軌跡データからの群パターン発見のための実用的アルゴリズム

耿 暁亮[1,a)] 宇野 毅明[2,b)] 有村 博紀[1,c)]

受付日 2014年4月6日, 採録日 2015年1月7日

概要：群パターンは，Gudmundsson らによって 2006 年に提案された時空間パターンであり，一群の物体が指定した時間以上の間，指定された距離以内で，一緒に移動する様子を表す．本稿では，群パターン発見のための深さ優先探索型アルゴリズム FPM および，そのための 2 つの高速化手法を提案する．さらに，その有用性を評価するために，提案手法と既存の幅優先探索型アルゴリズム BFE を実装し，人工データと実データ上で比較実験を行う．結果として，2 つの高速化手法を組み合わせることによって，著しい高速化を達成できることが示された．

キーワード：軌跡データマイニング，時空間パターン，移動物体，群パターン，閉パターンマイニング

Practical Algorithms for Mining Flock Patterns from Trajectories

Xiaoliang Geng[1,a)] Takeaki Uno[2,b)] Hiroki Arimura[1,c)]

Received: April 6, 2014, Accepted: January 7, 2015

Abstract: Flock patterns are a class of spatio-temporal patterns that represent groups of objects moving close each other in a given time segment, proposed by Gudmundsson et al. in 2006. In this paper, we propose depth-first algorithms for mining flock patterns based on depth-first frequent itemset mining approach. Especially, propose two speed-up techniques for flock pattern mining, where one uses a class of closed patterns, called rightward length-maximal flock patterns, and the other uses geometric indexes such as quad trees. To evalute these extensions, we make empirical comparison of our algorithms to BFE algorithm (Vieira et al., 2009) on synthesis datasets and real datasets. The experiments demonstrated that the modified algorithms with the above speed-up techniques are order of magnitude faster than the basic algorithm and BFE algorithm in most parameter settings.

Keywords: trajectory data mining, spatio-temporal pattern, moving object, flock pattern, closed pattern

1. はじめに

1.1 背景

移動体デバイスと位置センサの爆発的な普及により，大規模軌跡データの高度利用が注目されている．**群パターン**（flock pattern，むれ/ぐんパターン）は，Gudmundsson ら [9] によって提案された時空間パターンであり，n 個の移動体の時間長 T にわたる軌跡からなる入力データベースにおいて，m 個以上の一群の物体が少なくとも時間 k の間，指定された距離 r 以内で，一緒に移動する様子を表すパターンである．以下では，これらのパラメータ m と，k，r を，それぞれ，パターンの最小支持度と，最小長さ，最大幅と呼ぶ．

本稿では，二次元平面上の与えられた軌跡データの集合から，最小時間と最大幅を満たすすべての群パターンをも

1 北海道大学大学院情報科学研究科
Hokkaido University, Sapporo, Hokkaido 060–0814, Japan
2 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
a) gengxiaoliang@ist.hokudai.ac.jp
b) uno@nii.jp
c) arim@ist.hokudai.ac.jp

本稿の内容は 2013 年 9 月の FIT2013 第 12 回情報科学技術フォーラムにて報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

© 2015 Information Processing Society of Japan

れなく発見する問題を考察する．以下では，このような群パターンを (r,k)–群パターンと呼ぶ．(r,k)–群パターンの発見は，野生動物や歩行者の移動に関する GPS データの解析や，移動車両等のプローブカーデータからの知識発見に有用である．

群パターン発見問題に対して，様々なアルゴリズムが提案されている [4], [9], [11], [15]．たとえば，文献 [4] は，群パターン発見の高速なアルゴリズムを与えているが，これはパターンの中心となる軌跡を選択し，この中心軌跡から距離が r 以内にある他の軌跡を集めることで，パターンの幅が r から $2r$ の範囲にある群パターンを見つける．しかし，このアルゴリズムは近似アルゴリズムであり，中心軌跡を持たないような，いくつかの幅 r 以内の群パターンを見落としてしまうという問題点を持つ [4]．我々の知る限りでは，現在までに，すべての群パターンを見つける多項式遅延領域計算量を持つアルゴリズムは提案されていない．

そこで本稿では，効率良いパターン発見手法として知られているパターン成長アプローチ [14] に基づいて，群パターン発見のための効率良い深さ優先マイニングアルゴリズムを与える．

1.2 主結果

本稿では，軌跡データベースから群パターンを発見するための新しい深さ優先探索型アルゴリズム FPM（Flock Pattern Miner）を開発・実装し，その実験的評価を行う．さらに，2 つの高速化手法として，右極大な群パターンを用いた改良版プログラム FPM-R と，右極大群パターンと空間索引の両方を用いた手法による改良版プログラム FPM-R-G を与える．

基本アルゴリズムである FPM は，頻出集合発見で広く用いられているパターンの深さ優先探索手法 [16], [18] を，群パターン発見に拡張したものである．このアルゴリズムは，ただ 1 つの軌跡からなる最小のパターンから出発して，パターンの幅が変わらない限り，新しい軌跡を 1 つずつ付け加えてパターンを拡張しながら，すべての群パターンを系統的に列挙する．最初に基本的な結果として，我々は FPM が入力サイズの多項式遅延・領域で，すべての (r,k)–群パターンを重複なく見つけることを示す．この結果は，文献 [16] のアイテム集合発見手法と文献 [2] の系列パターン発見手法の双方を合わせて，軌跡データからの群パターン発見に拡張したものと考えることができる．

提案の 2 つの高速化手法に関して，第 1 の高速化手法の FPM-R は，発見対象のパターンクラスである右極大パターン（rightward-maximal patterns）を用いた高速化手法である．右極大パターンとは，パターンが含む移動物体の集合を保存したまま，時区間を可能な限り右側に延長して得られる一種の閉パターン（closed pattern）[16] であり，これにより発見するパターンの情報を失わずに，パターン数を削減することで高速化を行う．実際に，FPM-R はすべての右極大 (r,k)–群パターンに対する多項式遅延・領域列挙アルゴリズムになっている．

第 2 の高速化手法の FPM-R-G は，幾何的な空間索引（geometric spatial index）を用いる高速化手法である．これは，探索対象となる移動物体が満たす幾何的な制約を導入し，これを 4 分木や領域木等の空間索引 [7] を用いて高速に絞り込むことで，発見の高速化を行う．結果として，FPM-R-G は計算機実験で良好な高速化性能を示している．

人工データ上の実験では，右極大群パターンと空間索引の 2 つの高速化手法を組み合わせた改良版プログラムの FPM-R-G は，比較対象の BFE アルゴリズム [17] と比較して，網羅的にパターンを見つけるとともに，数千倍以上高速に全パターンを発見した．さらに，改良型の FPM-R-G と FPM-R は，広いパラメータ値に対して効率良く動作し，FPM より大幅な高速化を示した．特に，FPM-R-G は，サイズ 10 万点以上の軌跡データに対して約 1 秒以下で動作し，広いパラメータ値に対して BFE より大変高速だった，FPM に対して約 930 倍高速であった．

最後に，実際の台風軌跡データを用いた実験では，提案アルゴリズムが膨大な候補の中から現実的な時間でデータの共通の特徴を表した群パターンを発見し，実データにおける有用性を示した．

1.3 本稿の貢献

本稿の貢献は次のようにまとめられる．

- 群パターン発見問題を，完全な解集合を出力する列挙アルゴリズムの立場から研究した．
- 群パターンに対する多項式遅延・領域計算量を持つ効率良い全解パターン発見アルゴリズムを初めて与えた．
- 連続空間における閉パターンの一種として，右極大群パターンのクラスを導入し，幾何的な制約に基づく閉パターンの効率的な処理技法を考案した．
- 提案したアルゴリズムを実装し，人工データと実データ上で，計算機実験による実証的評価を行った．その結果，既存アルゴリズムに対して，高速化手法の有効性や，性能の安定性を示すことができた．

以上のように，本稿は，群パターンとその拡張に対する完全かつ重複のないパターン発見問題を考察し，理論的性能保証を持ち，実際のデータ上でも効率良く動作する，新しい方式を提案している．

1.4 本稿の構成

本稿の構成は次のとおりである．2 章では，既存の群パターン発見手法を概観する．3 章では群パターンを定義する．4 章では提案手法を述べる．5 章では実験の結果を述べる．最後に，6 章で本稿をまとめる．

© 2015 Information Processing Society of Japan

2. 先行研究

2.1 頻出集合発見と群パターン発見

頻出集合や系列パターンのパターン発見アルゴリズムの歴史は古く，データマイニング研究の最初期の Apriori アルゴリズム [1] 等が知られている．特に，1990 年代終わりから，PrefixSpan [14] や LCM [16] 等のパターン空間の深さ優先探索型アルゴリズム[*1]が提案された．メモリ効率の良さと実装の容易さから，現在，広く用いられている．

これに対して，軌跡データからの群パターン発見アルゴリズムの研究の歴史は比較的新しく，2000 年代に入ってから，いくつかの手法が提案されている [4], [9], [11], [15]．萌芽的な研究である Laube らの研究 [11] は，長さが 1 の群パターンを導入し，それらの変種として，出会い（meet）と，発散（divergence），リーダ（leading）等の位置と速度の両方を考慮した群パターンの発見を考察している．

2.2 近似型の群パターン発見アルゴリズム

続く研究として，Benkert ら [4] は，本稿で扱っている一般の最小長さ k と，最大幅（半径）r，最小頻度 m を持つ群パターンのクラス $\mathcal{FP}(r,k,m)$ を導入し，最大幅 r に関する近似アルゴリズムを与えている．

彼らのアルゴリズムでは，中心となる移動体を考えて，その軌跡から，すべての時点で距離が r 以内にあるような移動体すべてからなる群パターンに制限してパターン発見を行う．真の半径 r_* に対して最大半径を $r = 2r_*$ にとることで，彼らのアルゴリズムは，半径が r から $2r$ の範囲にあるパターンのいくつかを，$O(k2^k n \log n + k^2/\epsilon^{2k-1})$ 時間で近似的に見つける．ここに ϵ は定数である．

パターン発見の立場からは，Benkert ら [4] のアルゴリズムは近似アルゴリズムであり，すべての $\mathcal{FP}(r,k,m)$ 群パターンを見つける保証を持たない．また k に関して指数時間を要するという問題を持つ．

最近の研究として，Buchin ら [5] は位相空間における Reeb グラフ [13] のアイディアに基づいて，多数の軌跡データ中に含まれる群パターンの離合集散に関する大局的構造を，グラフ構造として抽出する効率良いアルゴリズムを与えている．この構造はある意味で，データ中の群パターンの代表元の集合を表現しているが，これを用いてすべての (r,k)–群パターンを網羅的に見つけられるかは分かっていない．Fort ら [8] は，極大群パターンを発見するアルゴリズムの GPU 上での並列実装アルゴリズムを与えている．

2.3 全列挙型の群パターン発見アルゴリズム

一方で，頻出パターン発見のように，制約を満たす群パターンすべての完全な集合を発見する研究は，著者らの知る限りでは，現在の時点で次の 2 つだけである．どちらのアルゴリズムも，深さ優先探索方式を採用しておらず，大きな作業用メモリを要し，低メモリアルゴリズムではない．

BFE（Basic Flock Evaluation）アルゴリズム [17] は，可能な r–円盤（半径 r の点集合のクラスタ）の組合せを幅優先探索して，$\mathcal{FP}(r,k,m)$ の群パターンすべてを発見する．このアルゴリズムは，パターン出力 1 回あたり n と r，k，m の多項式時間で動くが，実際には同一のパターンを複数回出力してしまうため，多項式遅延と多項式領域の列挙アルゴリズムではない．また，幅優先探索における中間解の保持のためにメモリ効率が良くない．一方，オンライン発見ができるのが長所である．

Romero による前処理型の群パターン発見アルゴリズム [15] は，与えられた軌跡データベース S をあらかじめトランザクションデータベース S' に変換し，S' から頻出集合発見アルゴリズム LCM [16] を用いてすべての頻出集合を発見し，これから群パターンすべてを発見する．このアルゴリズムは，指数的な数の中間解の生成が必要なため，多項式遅延と多項式領域の列挙アルゴリズムではない．

3. 準備

本章では，軌跡データベースからの群パターン発見問題のための基本的な定義と記法を導入する．ここで説明されない用語については，アルゴリズム分野 [6] や計算幾何学分野 [7] の教科書を参照されたい．

3.1 基礎的な定義

$\mathbb{R}$ と $\mathbb{N}$ のそれぞれを実数と非負整数の集合とする．整数 a，b（$a \le b$）に対して，$[a,b] = \{a, a+1, \ldots, b\}$ を a から b までの離散区間とする．もし a，b（$a \le b$）が実数ならば，$[a,b]$ で $\mathbb{R}$ の連続的な区間 $\{\, x \in \mathbb{R} \,|\, a \le x \le b \,\}$ を表す．集合 A に対して，$|A|$ は A の中の要素数を表す．A^* で，長さ零以上の A の要素列の全体を表す．

3.2 軌跡データベース

非負整数 n と $T \ge 0$ のそれぞれで，移動体の数と離散時間印の最大値を表す．$\mathbb{R}^2$ で二次元平面全体を表す．

ここで，**空間領域**（spatial domain）を $\mathbb{D} = \mathbb{R}^2$ とおき，**時間領域**（temporal domain）を $\mathbb{T} = [1,T] = \{1, \ldots, T\}$ とおく．空間の**点**（point）または**位置**（location）は，任意の実数の組 $p = (p.x, p.y) \in \mathbb{R}^2$ であり，**時刻**（time）は，任意の正整数 $t \in \mathbb{T}$ である．$(\mathbb{D}, \mathbb{T})$ 上の**軌跡**（trajectroy）は，平面上の長さ T の点列

$$s = s[1] \cdots s[T] \in (\mathbb{D})^T$$

である．ここに，各 $t = 1, \cdots, T$ に対して，$s[t] \in \mathbb{D}$ は軌跡 s の第 t 番目の時刻の点である．

移動体の個数を n とおく．n 個の**移動体** $o_1, \ldots, o_n$ の**識**

[*1] パターン成長型（pattern growth type）とも呼ばれる．

© 2015 Information Processing Society of Japan

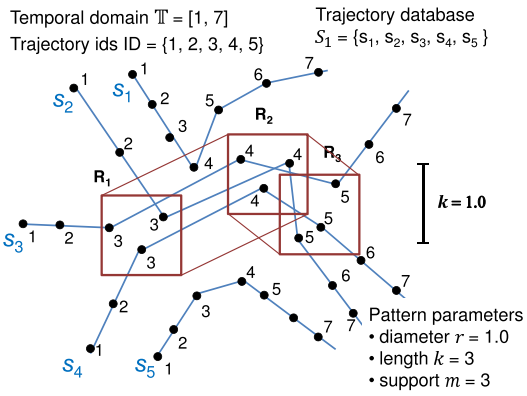

図 1 移動体集合 $ID = \{1, \ldots, 5\}$ と，空間領域 $\mathbb{D} = \mathbb{R}^2$，時間領域 $\mathbb{T} = [1, 7]$ 上の軌跡データベース $S_1 = \{S_1, \ldots, S_5\}$ と群パターン $P_1 = (X_1 = \{2, 3, 4\}, I_1 = [3, 5])$．ここに，$S_1$ の軌跡上の数字と黒点は時刻と移動体の位置を表す

Fig. 1 Examples of a trajectory database S_1 on $ID = \{1, \ldots, 5\}$ and $\mathbb{T} = [1, 7]$ and a flock pattern $P_1 = (X_1 = \{2, 3, 4\}, I_1 = [3, 5])$. Here, each line indicates a trajectory and the numbers attached to points are time stamps.

別番号集合を $ID = \{1, \ldots, n\}$ とおく．簡便のため，移動体 ID $i \in ID$ と対応する移動体 o_i を同一視することがある．

定義 1 $(\mathbb{D}, \mathbb{T})$ 上の**軌跡データベース**（trajectroy database）は，n 個の移動体 $o_1, \ldots, o_n$ の軌跡データの集合

$$S = \{\, s_i \,|\, i = 1, \ldots, n \,\} \subseteq \mathbb{D}^T \tag{1}$$

である．ここに，各 $s_i = s_i[1] \cdots s_i[T] \in \mathbb{D}^T$ $(i = 1, \ldots, n)$ は第 i 番目の軌跡であり，$s_i[t]$ は，移動体 $i \in ID$ の時刻 $1 \le t \le T$ における位置を表す[*2]．

軌跡データベースの例として，GPS センサを持つ野生動物や歩行者等の GPS の軌跡の集合がある [11]．

例 1 軌跡データベースの例を，**図 1** に示す．この図の軌跡データベース S_1 は，時間区間 $\mathbb{T} = [1, 7]$ を持つ $n = 5$ 個の移動体 $o_1, \ldots, o_5$ に対する軌跡データを含む．

3.3 群パターンのクラス

定義 2（群パターン [4]） $(\mathbb{D}, \mathbb{T})$ に対する**群パターン**（flock pattern，FP）は対 $P = (X, I)$ である．ここに，$X \subseteq ID$ は移動体 ID の有限集合であり，P の **ID 集合**（ID set）と呼ばれ，$I = [b, e] \subseteq \mathbb{T}$ は $[1, T]$ 上の離散区間であり，**時間区間**（time interval）と呼ばれる．時間 b と e $(b \le e \le T)$ は，それぞれ，P の**開始時間**と**終了時間**と呼ばれる．今後，P の ID 集合および，時間区間，開始と終了時間を，それぞれ，$P.set = X$ および，$P.span = I$，$P.start = b$，$P.end = e$ と表す．

Algorithm 1 データベース $S = \{\, s_i \,|\, i = 1, \ldots, n \,\}$ において，群パターン $P = (X, [b, e])$ の幅 $width_S(P)$ を計算するアルゴリズム．

```
1: width ← 0;
2: for all t ∈ [b, e] do
3:     S_t ← { s_i[t] | i ∈ X };                ▷ the t-th slice
4:     width ← width と diam_S(S[x][t]) の最大値;
5: return width;
```

以下のように群パターンの支持度と，長さ，幅を定義する．P の**支持度**（support）を $supp(\mathrm{P})$ と表し，X 中の軌跡データの個数を $supp(P) \triangleq |P.set|$ と定義する．P の**長さ**を，$len(P) \triangleq |P.span| = P.end - P.start + 1$ と定義する．明らかに，$0 \le supp(P) \le n$ と $0 \le len(P) \le T$ が成立する．

例 2 図 1 に，データベース S_1 上の群パターンの例 P_1 を示す．このパターンの ID 集合は $P_1.set = X_1 = \{2, 3, 4\}$ であり，時間区間は $P_1.span = I_1 = [3, 5]$ である．

群パターンの幅を定義するために，以下の定義を準備する．二次元平面 $\mathbb{D}$ において，点 p と p' の L_∞ **距離**（または単に**距離**）を $d_\infty(p, p') \triangleq \max\{|p.x - p'.x|, |p.y - p'.y|\}$ と定義する[*3]．

点集合 $A = \{p_1, \ldots, p_n\}$ の**直径**（diameter）を A に含まれるすべての 2 点間の最大 L_∞ 距離 $diam_S(A) \triangleq \max_{p,p' \in A} L_\infty(p, p')$ と定義する．直径 $diam_S(A)$ は非負であり，A の要素数の線形時間で計算できる．

入力データベースを $S \subseteq \mathbb{D}^T$ とし，ID 集合を $X \subseteq ID$，時刻を $t \in \mathbb{T}$ $(1 \le t \le T)$ とする．X に含まれるすべての軌跡データの時刻 t における位置すべてのなす集合を，X の軌跡の時刻 t における**スナップショット**（snapshot），または**断面**（cross section）と呼び，$S[X][t] = \{\, s_i[t] \in \mathbb{D} \,|\, i \in X \,\}$ と定義する．以上をもとに，群パターン P の**幅**（width）を，P の軌跡の時刻 $t \in P.span$ における断面 $S[X][t]$ の直径の最大値 $width_S(P) \triangleq \max_{t=1,\ldots,T} diam_S(S[X][t])$ と定義する．

補題 1 群パターン P に対して，アルゴリズム 1 は P の幅 $width_S(P)$ を $O(m\ell)$ 時間で計算する．ここに，m は P の支持度であり，ℓ は長さである．

正数を $r > 0$ とし，非負整数を $k, m \ge 0$ とする．パラメータ r と，k，m を，**最大幅**（max-width）と，**最小長さ**（min-len），**最小支持度**（min-sup）パラメータという．S 上の任意の群パターン P が **r–群パターン**であるとは，$width_S(P) \le r$ を満たすことをいう．さらに，もし P が

[*2] 本稿では，すべての軌跡は同じ長さを持つものとする．短い軌跡を扱う場合には，先頭または末尾に互いに無限の距離離れた仮想的な点列をパディングすることで，結果を変えずに扱える．詳細は略する．

[*3] 本稿の結果は二次元連続空間の L_∞ 距離を仮定しているが，定数次元の連続空間に容易に一般化できる．さらに，線形時間の最小包含円計算 [7], [12] を用いることで，固定次元の L_2 距離に拡張できる．詳細は別の機会に譲る．

© 2015 Information Processing Society of Japan

$len(P) \geq k$ を満たすならば，P は (r,k)–群パターンであるといい，もしさらに，$supp(P) \geq m$ を満たすならば，P は (r,k,m)–**群パターン**であるという．(r,k)–群パターン全体のクラスを $\mathcal{FP}(r,k)$ と定義し，同様に，$\mathcal{FP}(r)$，$\mathcal{FP}(r,k,m)$ も定義する．

例 3 図 1 のパターン P_1 の直径は $width_S(P_1) \leq 1.0$ であり，長さは $len(P_1) = 3$，支持度は $supp(P_1) = 3$ である．ここに，$S = S_1$ である．よって，図の群パターン $P_1 = (X_1, I_1)$ は，$r = 1.0$ と，$k = 2$，$m = 3$ に対する (r,k,m)–群パターンである．

主に本稿では，軌跡データベース S に含まれる (r,k)–群パターンを網羅的に見つける方法を考察する．

3.4 右極大群パターン

新しく右極大群パターンのクラスを導入する．与えられた最大幅パラメータ $r > 0$ に対して，パターンが右極大であるとは，S 上で $width_S(P)$ を大きくすることなく，時間区間 $P.span$ を時間軸に沿って右方にできるだけ長く延長したものをいう．

定義 3（右極大群パターン） (r,k)–群パターン P が，S 上で**右極大群パターン**（rightward-maximal flock pattern, RFP）であるとは，次の条件 (1) と (2) を満たすような他の (r,k)–群パターン Q が存在しないことという：

(1) $P.set = Q.set$.

(2) $P.start = Q.start$ かつ $P.end < Q.end$.

以後，群パターンと右極大群パターンを，それぞれ，**FP** と **RFP** と呼ぶ．このような右極大な群パターンの発見は，解の総数と計算時間を減らすのに役に立つ．**右極大な (r,k)–群パターン全体のクラス**を $\mathcal{RFP}(r,k)$ で表し，同様に，$\mathcal{RFP}(r)$，$\mathcal{RFP}(r,k,m)$ も定義する．$\mathcal{RFP}(r,k) \subseteq \mathcal{FP}(r,k)$ であるが，この逆は一般には成立しない．

例 4 図 1 の軌跡データベース S_1 において，長さ 3 の群パターン $P_1 = (X_1 = \{2,3,4\}, I_1 = [3,5])$ は右極大群パターンであるが，同じ ID 集合を持ち，区間が真の接頭辞であるパターン $P_2 = (X_1, I_2 = [3,4])$ と $P_3 = (X_1, [3])$ は右極大ではない．一方，同じ ID 集合を持ち，区間が真の接尾辞であるパターン $P_4 = (X_1, [4,5])$ と $P_5 = (X_1, [5])$ は右極大である．

3.5 データマイニング問題

以下では，最大幅 r と最小長さ k の制約を持つ群パターンのクラス $\mathcal{FP}(r,k)$ とクラス $\mathcal{RFP}(r,k)$ を考える．初めに，クラス $\mathcal{FP}(r,k)$ に対する群パターン発見問題を与える．

定義 4（群パターン発見問題） 軌跡データベース S と，パラメータ $r > 0$ と $k \geq 0$ を入力として受け取り，S における (r,k)–群パターン $P \in \mathcal{FP}(r,k)$ のすべてを，重複なしに出力せよ．

支持度 m を持つ (r,k)–群パターン全体の総数は kn^mT 個以下である．これは k および n，T に関しては多項式時間だが，m に関して指数的である．

次に，クラス $\mathcal{RFP}(r,k)$ に対する右極大パターン発見問題を与える．

定義 5（右極大群パターン発見問題） 軌跡データベース S と，パラメータ r，$k \geq 0$ を入力として受け取り，S における右極大 (r,k)–群パターン $P \in \mathcal{RFP}(r,k)$ のすべてを，重複なしに出力せよ．

上記の 2 つの問題は，それぞれ，クラス $\mathcal{FP}(r,k)$ と $\mathcal{RFP}(r,k)$ に対するパターンを，解として網羅的に見つける列挙問題にほかならない．一般に列挙問題は，入力サイズ N の指数個の解を持つことが多い．そこで，列挙アルゴリズムの性能は，出力依存計算量で測ることが多い [3]．入力サイズ N と解数 M に対して，列挙アルゴリズム $\mathcal{A}$ が**出力多項式時間**であるとは，その総計算時間が N と M の多項式 $poly(N,M)$ でおさえられることをいう．また，$\mathcal{A}$ が**多項式遅延**であるとは，その**遅延**（delay），すなわち，連続した 2 つの解の間にかかった時間の最大値が，N の多項式 $poly(N)$ でおさえられることをいう．

（非極大な）群パターン列挙問題は，すべての可能なパターンを数えあげる素朴な方法を用いて $O(kn^mT)$ 時間で解ける．しかし，この素朴な方法は見つけられる群パターンの頻度 m に対して，指数遅延を持つ．さらに，極大版の問題に対して，中間解を記憶して極大性のチェックが必要なため，指数領域を必要とする．

4. 提案手法

本章では，$\mathcal{FP}(r,k)$ と $\mathcal{RFP}(r,k)$ を網羅的に見つける効率良いマイニングアルゴリズムを与える．さらに，地理インデックスを使用して無駄なパターンの探索を除去する技術を導入する．

4.1 FP のための基本 DFS アルゴリズム

Algorithm 2 に，$\mathcal{FP}(r,k)$ のための深さ優先探索型の群パターンマイニングアルゴリズム FPM（Flock Pattern Miner）を導入する．このアルゴリズムは，主手続き FPM と副手続き RecFPM から成るバックトラック型アルゴリズムであり，単純だが，以降の節でのアルゴリズムの基本となる．

トップレベルの主手続 FPM は，2 行目から 4 行目の 3 重の for ループにおいて，可能なすべての長さ $\ell \in [k,T]$ と，すべての初期 ID $1 \leq i \leq n$，すべての開始位置 $b \in [1,T]$ の組合せに対して，単一 ID 集合 $X = \{i\}$ と長さ ℓ の時間区間 $I_b = [b,e]$) を引数として，副手続き RecFPM を呼び出す．ここに，$e = b + \ell - 1$ である．これらの異なる呼び出しの総数はたかだか $O(nT^2)$ 個である．

このとき，副手続き RecFPM の呼び出しは，頻出アイ

© 2015 Information Processing Society of Japan

Algorithm 2 最大幅 $r > 0$ と最小長さ $k \geq 1$ に対して，軌跡データベース S に含まれるすべての (r,k)–群パターンを見つけるアルゴリズム．

```
1: procedure FPM(ID, S, r, k)
2:     for ℓ ← k, ..., T do                          ▷ Each length ℓ ≥ k
3:         for b ← 1, ..., T do                      ▷ Each start time
4:             for i ← 1, ..., n do                  ▷ Each ID
5:                 P を生成する; P.set ← {i}; P.span ← [b, e];
6:                 ID ← ID − {i}, where e = b + ℓ − 1;
7:                 RecFPM(P, ID, S, r)
8: procedure RecFPM(P, ID, S, r)
9:     if width_S(P) > r then return;                ▷ backtrack
10:    output P;
11:    ID_1 ← ID;
12:    while ID_1 ≠ ∅ do
13:        i = deletemin(ID_1);
14:        Q を生成する; Q.set ← P.set∪{i}; Q.span ← P.span;
15:        RecFPM(Q, ID_1, S, r);                    ▷ recursive call
16:    end while
```

テム集合マイニングのための深さ優先探索アルゴリズム（Zaki [18] や LCM [16]）のように深さ優先探索を用いて，初期 ID を含む部分集合 $X \subseteq ID$ をすべて列挙する．これには，次の**末尾拡張**（tail expansion）[16], [18] と呼ばれる方法を用いる．ID 集合 $X, Y \subseteq ID$ に対して，Y が X の末尾拡張であるとは，ある $i > \max(X)$ を満たす要素 $i \in ID$ に対して，$Y = X \cup \{i\}$ が成立することをいう．この条件を満たすために，アルゴリズムの 6 行目と 13 行目で候補 ID 集合から追加した ID を除去している．単一集合 $\{i\}$ からスタートして，末尾拡張によって親 X から子 Y を生成することを繰り返すことで，初期 IDi を含むすべての部分集合を重複なく列挙できる[*4]．

今，現在の引数としてパターン P を受け取ったとすると，副手続き RecFPM は S 中の軌跡データにアクセスして，群パターン P の幅 $width_S(P)$ を計算し，これが最大幅の制約を満たすかどうか検査する．もし検査に失敗すると，P とその全部の子孫の探索をやめる．この枝刈規則の正当性は以下の補題で保障される．

補題 2（群パターンの反単調性） P と Q を，$P.span \subseteq Q.span$ を満たす 2 つの群パターンとする．もし $P.set \subseteq Q.set$ ならば，$width_S(P) \leq width_S(Q)$ が成立する．

上の補題より，候補パターン P が最大幅の検査に失敗すると，P に新しい軌跡データ i を加えてできた候補パターン $Q = P \cup \{i\}$ も幅の検査を通らない．したがって，子 Q のすべての子孫の探索空間を取り除くことができる．これによって，$\mathcal{FP}(r,k)$ の解が誤って除去されることはない．

アルゴリズム FPM の時間と領域の計算量について，以下の命題が得られる．

命題 1 長さ T の軌跡 n 個からなるデータベース S において，Algorithm 2 のアルゴリズム FPM は，S 中に出現するすべての (r,k)–群パターン $P \in \mathcal{FP}(r,k)$ を，$O(knT^2)$ 遅延と $O(m^2)$ 領域で重複なしにすべて見つける．ここに，$m = supp(P)$ は列挙されるパターン P の支持度である．

[*4] FPM だけのためには $\emptyset$ からスタートしてすべての部分集合を列挙すれば十分だが，4.3 節の FPM-R-G では初期 ID i が必要なので，拡張性のため i を与えている．

Algorithm 3 入力群パターン $P = (X, [t_0, e_0])$ と同じ ID 集合と開始位置を持つ右極大群パターンを求めるアルゴリズム．

```
1: procedure RClosure(P = (X, [t_0, e_0]), S, r)
2:     t ← t_0;
3:     while diam_S(S[X][t]) ≤ r do
4:         t ← t + 1;
5:     b ← t_0; e ← t − 1;
6:     return P_max(X, [b, e]);
7: 補足：任意の t ∉ [1, T] に対して，diam_S(S[X][t]) ≜ ∞.
```

4.2 RFP のための改良マイニングアルゴリズム

本節では，右極大群パターンのクラス $\mathcal{RFP}(r,k)$ に対する改良アルゴリズム FPM-R（Rightward-maximal Flock Pattern Miner）を与える．Algorithm 4 に，アルゴリズム FPM-R と副手続き RecFPM-R を示す．

4.2.1 右向き閉包

頻出パターンマイニングの視点から見ると，右極大群パターンは一種の**閉パターン**（closed pattern）であると考えられる．頻出アイテム集合マイニング（FIM）[16], [19] の分野において，各種の閉パターンを列挙する多数のアルゴリズムが提案されている．これらのアルゴリズムの多くは，**閉包計算**（closure computation）と呼ばれるあるクラスの手続きを使用している．ここで，群パターンに対する閉包計算を導入しよう．

定義 6（右向き閉包） S を軌跡データベースとし，$r > 0$ を任意の正数とする．任意のパターン P の**幅 r の右向き閉包**（rightward closure）は，次の条件を満たす唯一の群パターン $\mathsf{RClosure}(P, S, r) = Q$ である：

(1) $P.set = Q.set$.

(2) $P.start = Q.start$.

(3) $Q.end = \max\{\, P.end \leq e' \leq T \,|\, (P.end \leq \forall t \leq e')\ diam_S(S[P.set][t]) \leq r \,\}$.

右向き閉包操作は，幅を r 以下に保ち，同時に，ID 集合 $P.set$ と開始時間 $P.start$ はまったく変えない．右向き閉包は，Algorithm 3 の手続き RClosure を用いて，$O(k\ell)$ 時間で計算可能である．ここに，入力パターン $P = (X, I)$ と出力パターン $P_{max} = (X, I')$ に対して，$k = supp(P) = |X| = O(n)$ かつ $\ell = len(P_{max}) = O(T)$ である．

次の補題は右向き閉包の特徴づけを与える．ここに，$t < 1$ または $t > T$ のように区間を外れた時刻 t に対して，$diam_S(S[X][t]) = \infty$ と定義しておく．

© 2015 Information Processing Society of Japan

Algorithm 4 最大幅 $r > 0$ と最小長さ $k \geq 1$ に対して，軌跡データベース S に含まれるすべての右極大 (r,k)–群パターンを見つけるアルゴリズム．

```
1: procedure FPM-R(ID = {1,...,n}, S, r, k)
2:     for t0 ← 1,...,T do                          ▷ Each start time in 𝕋
3:         for i0 ← 1,...,n do                      ▷ Each id in ID
4:             P0 ← ({i0}, [t0, ∗]); ID ← ID − {i0};
5:             RecFPM-R(P0, ID, S, r, k);
6: procedure RecFPM-R(P, ID, S, r, k)
7:     P∗ ← RClosure(P, S, r);
8:     if len(P∗) < k then return ;                 ▷ backtrack
9:     output P∗;
10:    ID1 ← ID;
11:    while ID1 ≠ ∅ do
12:        i = deletemin(ID1);
13:        Q を生成する; Q.set ← P∗.set ∪ {i};
14:        Q.start ← P.start; Q.end ← P∗.end;
15:        RecFPM-R(Q, ID1, S, r, k);               ▷ recursive call
16:    end while
```

補題 3 任意の (r,k)–群パターン P に対して，P が幅 r の右極大群パターンである $\iff$ $P = \mathsf{RClosure}(P,S,r)$ が成立する．

証明：任意の r–群パターン P に対して，P が右極大 r-群パターンであることと，以下の (1) と (2) が成立することは同値である：(1) すべての $t \in [b,e]$ に対して，$diam_S(S[X][t]) \leq r$；(2) $diam_S(S[X][e+1]) > r$．このことと，Algorithm 3 から，ただちに結果が示される． □

4.2.2 アルゴリズム全体

以上の準備の下に，Algorithm 4 にアルゴリズム FPM-R を示す．FPM-R の全体の構造は，$\mathcal{FP}(r,k)$ のための基本的なアルゴリズム FPM と類似しているが，Avis ら [3] の逆探索技法に基づいて，右向き閉包を用いて右極大パターンを見つけるように拡張されている．

主手続き FPM-R は，FPM と同様に，軌跡データベース S を入力として受け取り，2–3 行目の二重の for 文で，$P_0 = (\{i_0\}, t_0)$ を初期パターンとして，再帰的副手続きの RecFPM を呼び出す．ただし，初期軌跡 ID i と開始位置 b は引数に渡すが，長さ ℓ は渡さずに未定義値 $*$ を渡す．これより，RecFPM の呼び出しはたかだか $O(nT)$ 個になる．4 行目では重複回避のため，ID から i_0 を除去する．

RecFPM($P = (X, [b,*])$) の計算は以下のステップで行う．

- 再帰的手続き RecFPM は，親パターン P を受け取り，その幅 r の右向き閉包 $P_* = \mathsf{RClosure}(P,S,r)$ を計算する（7 行目）．
- 得られたパターン P_* は，補題 3 より，最大幅 r を満たす右極大パターンである．8 行目と 9 行目では，P_* が最小長さ k の制約を満たしていれば P_* を出力し，逆に満たしていない場合は，P とその子孫を探索から除去する（枝刈り）．
- 最後に，12 行目から 15 行目では，パターン P_* の ID 集合に，新しい軌跡 ID i を加えて拡張し，子パターン Q を得て，再帰的に RecFPM-R を呼び出す．これを $max(P_*.set) < i$ なるすべての i に対して行い，可能な子をすべて生成する．このとき，重複パターンを出さないように i は ID から消す．

Algorithm 5 最大幅 $r > 0$ と最小長さ $k \geq 1$ に対して，空間索引を用いて，軌跡データベース S に含まれるすべての右極大 (r,k)–群パターンを見つけるアルゴリズム．

```
1: procedure FPM-R-G(ID, S, r, k)
2:     Let S = { s_i | i = 1,...,n };
3:     for t0 ← 1,...,T do                          ▷ Each start time
4:         Build a grid index for point set U ← S[t0];
5:                                  ▷ The time slice at time t0
6:         for i0 ← 1,...,n do                      ▷ Each ID
7:             c = (x, y) ← s_i0[t0];               ▷ initial point c
8:             R_c,r ← [x−r, x+r]×[y−r, y+r];
9:                          ▷ 2r×2r-query rectangle at center p
10:            ID0 ← { i ∈ U.Range(R_c,r) | i > i0 };
11:            P0 ← (X = {i0}, [t0, ∗]);
12:            RecFPM-R(P0, ID0, S, r, k);
```

以上から，FPM-R の時間と領域の計算量を示す．

定理 2（アルゴリズム FPM-R の正当性と計算量） 長さ T の軌跡 n 個からなるデータベース S において，Algorithm 4 のアルゴリズム FPM-R は，S 中に出現するすべての右極大 (r,k)–群パターン $P \in \mathcal{RFP}(r,k)$ を，$O(knT)$ 遅延と $O(m^2)$ 領域で重複なしにすべて見つける．ここに，$m = supp(P)$ は列挙されるパターン P の支持度である．

4.3 地理インデックスを用いた高速化

本節では，前節で与えた FPM-R に対して，2 次元平面上の空間索引を利用した高速化技術を導入する．Algorithm 5 に，改良したアルゴリズム FPM-R-G（Grid-based FPM-R）と副手続 RecFPM を示す．これは，四分木または領域木等 [7] の空間索引を用いて，FPM-R を改良したものである．

空間索引（spatial index）とは，N 個の点集合 $U = \{p_1, \ldots, p_N\} \subseteq \mathbb{D}$（$N \geq 1$）を格納するためのデータ構造であり，任意の与えられた長方形 R に対して，解集合

$$U.Range(R) = U \cap R \subseteq U$$

を返すレンジクエリ（range query）を提供する．以下では，空間索引として，点の挿入を $O(\log N)$ 時間で，K 点を返すレンジクエリを $O(\log(N) + K)$ 時間で提供する領域木を用いる [7]．

入力を受け取ると，アルゴリズム FPM-R-G は，3 行目の各時刻 $t_0 \in [1,T]$ に対する外側 for ループを実行し，前処理として，初期時刻 t_0 におけるすべての移動体の位置のなす集合

$$S[ID][t_0] = \{\, s_i[t_0] \mid i \in ID \,\} \subseteq \mathbb{D}$$

を空間索引 U へ格納する（4 行目）．

© 2015 Information Processing Society of Japan

次に，アルゴリズム FPM-R-G は，移動体 $i_0 \in ID$ に対する 6 行目の内側 for ループで，r–群パターンの深さ優先探索を実行する．初期パターンを $P_0 \triangleq (X = \{i_0\}, [t_0, *])$ とおく．今，r–群パターン Q は，$i_0 \in Q.set$ かつ $t_0 = Q.start$ を満たすとき，P_0 **を先祖に持つ**という．初期移動体 i_0 の時刻 t_0 での位置を $c = s_{i_0}[t_0] \in \mathbb{D}$ とおくと，次が成り立つ．

補題 4（幅 r の群パターンの十分条件） P_0 を先祖に持つ任意の群パターンを Q とおく．もし $width_S(Q) \leq r$ ならば，Q の移動体の集合 $Q.set$ は，その開始位置の集合 $H(Q) = S[Q.set][Q.start] \subseteq \mathbb{D}$ を，$c = (x, y)$ を中心点とする 1 辺 $2r$ の長方形

$$R_{c,r} = [x-r, x+r] \times [y-r, y+r] \subseteq \mathbb{D}, \tag{2}$$

の中に持つ，すなわち，$H(Q) \subseteq U.Range(R_{c,r})$ が成立する．

証明：Q の任意の移動体 i の初期位置を q とおく．Q は P_0 を先祖に持つので c を含む．一方，群パターン Q 中にともに含まれる移動体どうしは，同時刻には互いに距離 r 以内にある．よって，$L_\infty(c, q) \leq r$ であることから，点 c と q の両方が長方形 $R_{c,r}$ に含まれる． □

初期パターン P_0 からスタートして，完全かつ重複なしに，(r, k)–群パターンの探索を行うために，次の 2 つの条件

- (a1) $i \in ID_0 \triangleq U.Range(R_{c,r})$（補題 4 から）．
- (a2) $i > i_0$ を満たす移動体 ID $i \in ID$（重複回避）

を満たすような i だけを初期パターン P_0 に追加すればよい．10 行目でレンジクエリとリストの走査を用いて，

$$ID_0 \leftarrow \{\, i \in U.Range(R_{c,r}) \,|\, i > i_0 \,\} \tag{3}$$

を計算し，副手続き RecFPM-R 引数に与えて探索を開始する．以上の議論より，Algorithm 5 の正当性が示される．

FPM-R-G の計算量は，U の点の総数が $N = nT$ のときに，前処理において $O(N \log N)$ 時間で空間インデックス構築を行い，再帰的な副手続き RecFPM-R のトップレベルの呼び出しごとに，レンジクエリの時間 $O(\log(N) + |ID_0|)$ がかかる．これより，FPM-R-G の最悪時計算時間は，FPM-R とほぼ同じで，理論的には改善されていない．しかし，次の 5 章の実験で示されるように人工データでは計算時間削減にきわめて有効であった．

5. 実験

提案アルゴリズムを評価するために，人工データと実データ上の計算機実験を行った．

5.1 データ

5.1.1 人工データ

実験 A と実験 B は次の人工データを用いた．人工データの生成に用いたパラメータとそのデフォルト値を**表 1** の上部に示す．

表 1 人工データ上の実験のパラメータ値とそのデフォルト値
Table 1 The mining parameters and default parameters of experiments on synthesis datasets.

種類	パラメータ	記号	デフォルト値
軌跡データ	領域サイズ	a	100.0
	軌跡数	n	100
	軌跡長	T	100
	総点数	$N = nT$	10,000
正解	個数	h	6
パターン	長さ	k_*	50
	幅	r_*	1.0
	頻度	m_*	5
マイニング	最小長さ	k	40
パターン	最大幅	r	1.0
	最小頻度	m	5

人工データは，C++ で実装したデータ発生機を用いて，植込みパターン法に基づき，以下のように生成した．最初に，大きさ $a \times a$ の領域 $\mathbb{A}$ 上の一様分布に従う $N = nT$ 個のランダム点からなる長さ T の軌跡データ n 本を生成した．次に，$\mathbb{A}$ の乱歩として h 本の長さ k_* の正解パターンを生成し，それぞれから m_* 個の軌跡のコピーを作り，先に生成した軌跡にランダムな先頭位置と r_* の幅の摂動で埋め込んだ．

各実験では，1 つのパラメータを指定した範囲で変化させて，その他のパラメータは，表 1 に示したデフォルト値を用いて．次のように設定した：(a) 入力点数 n を変化させる実験では，$n = 10K, 20K, 40K, 80K, 160K$ の範囲で変化させた．(b) 最小長さ k を変化させる実験では，正解パターンの最小長さを $k_* = 40$ に固定し，$k = 10, 20, 30, 40, 50$ の範囲で変化させた．(c) 最大幅 r を変化させる実験では，正解パターンの最大幅を $r_* = r$ に保ちながら，$r = 0.5, 1.0, 1.5, 2.0, 2.5$ の範囲で変化させた．(d) 最小支持度を変化させる実験では，正解パターンの最小支持度を $m_* = m$ に保ちながら，$m = 3, 4, 5, 6, 7, 8$ での範囲で変化させた．

5.1.2 実データ

「デジタル台風」[10][*5]のサイトから，1951 年から 2014 年 10 月までのすべての台風軌跡データ 1,668 本をダウンロードし，50 点（約 300 時間）以上の長さを持つすべての軌跡データ 322 本を選択し，先頭からの長さ 50 点までを切り出して実データとした．各台風軌跡データは，発生から消滅までの台風の中心位置の緯度経度をサンプリング間隔 6 時間で記録したものであり，切り出す前の長さは 3 点から約 80 点であった．軌跡の開始時点（年月日時間）は，各軌跡ごとに異なるので，台風発生からの相対経過時間を時刻として使っている．

[*5] http://agora.ex.nii.ac.jp/digital-typhoon/

© 2015 Information Processing Society of Japan

表 2 実データ上の実験のパラメータ値
Table 2 The mining parameters of experiments on real datasets.

種類	パラメータ	記号	値
軌跡データ	軌跡数	n	322
	軌跡長	T	50
	総点数	$N = nT$	16,100

5.2 実験手法

実験では，4 章で提案した FPM 族のアルゴリズムと，比較対象として既存の BFE（Basic Flock Evaluation）アルゴリズム [17] の両方を C++ で実装し，GNU の g++ 4.6.3 版でコンパイルして，実験に用いた．具体的には，次のプログラムを用いた：

- FP パターンに対する円盤の組合せに基づく幅優先探索型のアルゴリズム BFE [17]
- FP パターンに対する提案アルゴリズム FPM（4.1 節）
- RFP パターンに対する提案アルゴリズム FPM-R（4.2 節）
- RFP パターンに対する空間索引を用いた提案アルゴリズム FPM-R-G（4.3 節）

なお，上記のうち，プログラム FPM-R-G と BFE は，C++ で実装した空間索引を内部で用いている．また，FPM 族の 3 つのアルゴリズムは対象クラスのすべてのパターンを重複なく出力するが，BFE は同じパターンを何回も出力する可能性があることに注意されたい．

上のプログラムに対して，実験 A と実験 B では，5.1.1 項の人工データ上で，表 1 の下部に示したパラメータ値を用いて，それぞれ，群パターンの総数と総計算時間を計測した．実験 C では，5.1.2 項の実データ上で，**表 2** に示したパラメータ値を用いて実験を行った．なお，総計算時間は入力データの読み込み時間を含む．なお，BFE は大きな使用メモリと総計算時間を要するため，実行時間の上限 10,800 sec（3 時間）を超えた場合は実行を打ち切った．

実験環境は，Intel(R) Xeon(R) CPU E5-1620, 3.60 GHz, 32 GB メモリを持つ PC を用いて，Ubuntu Linux, version 12.04 上で行った．予備実験では，すべての実装が正解パターン数以上のパターンを正しく発見したことを確認した．

5.3 実験 A の結果：パターン数の比較

実験 A では，5.2 節で示したプログラムのうち，FPM-R-G を除く 3 つに対して，入力点数 N と，最小長さ k，最大幅 r，最小支持度 m を変化させて出力パターン数を計測した．**図 2**，**図 3**，**図 4**，**図 5** に結果を示す．すべての実験において，FPM と FPM-R は完全かつ非冗長にパターンを見つけているが，その一方で BFE は多数の重複したパターンを生成していることが確認できた．

具体的には，図 2 で $N = 80$ K かつ長さ 40 の場合，データは 396 個の真の群パターンと 66 個の真の右極大群パターンを含む[*6]．これに対して，FPM と FPM-R は，それぞれ，真の数に一致する 396 個と 66 個の群パターンと右極大群パターンを見つけたが，一方で，BFE は重複も数えて，真の数の 10^5 倍以上である 26,595,008 個の群パターンを出力した．そのほとんどは重複パターンであった．

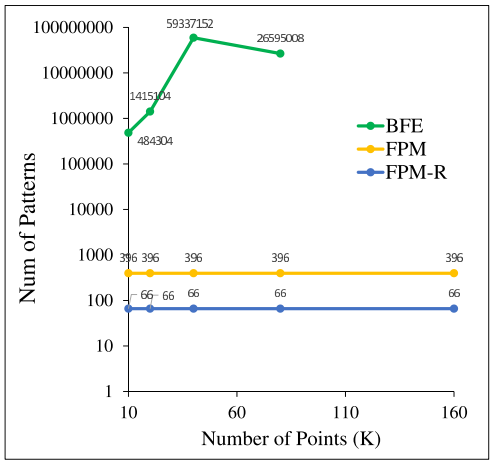

図 2 実験 A1：入力点数 N を変化させたときのパターン個数
Fig. 2 Exp A1: The pattern number by varying number of input points N.

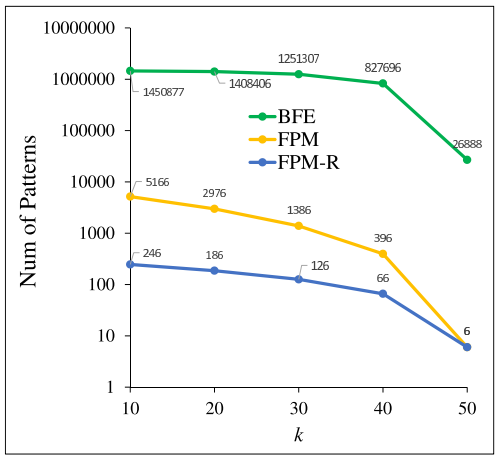

図 3 実験 A2：最小長さ k を変化させたときのパターンの個数
Fig. 3 Exp A2: The pattern number by varying minimum length k.

5.4 実験 B の結果：アルゴリズムによる計算時間の比較

5.2 節で示した 4 つのプログラムに対してアルゴリズムの総計算時間を計測した．**図 6**，**図 7**，**図 8**，**図 9** に，それぞれ，入力点数 N と，最小長さ k，最大幅 r，最小支持度 m を変化させた場合に，アルゴリズムの総計算時間を示した．

アルゴリズムの規模耐性については，図 6 のグラフにおいて，FPM と FPM-R の入力点数の増加に対する計算時間

[*6] この実験の設定では，長さ k_{max} の埋め込みパターン 1 個に対して，$g = k_{max} - k_{min} + 1$ とおくと，長さが $k = k_{min} \sim k_{max}$ の範囲の群パターンは $M_{FP} = \binom{g}{2} = g(g+1)/2$ 個存在し，右極大群パターンは $M_{RFP} = g$ 個存在することが容易に示せる．

© 2015 Information Processing Society of Japan

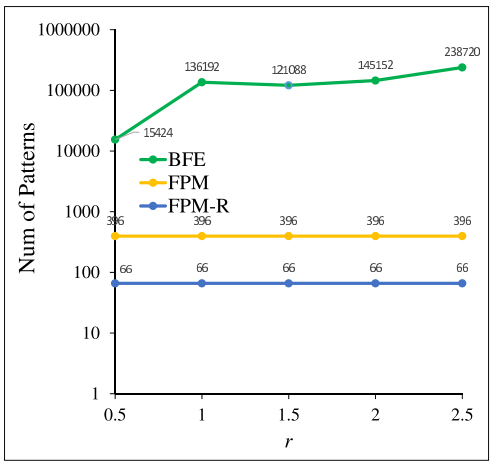

図 4 実験 A3：最大幅 $r_* = r$ を変化させたときのパターン個数

Fig. 4 Exp A3: The pattern number by varying maximum width $r_* = r$.

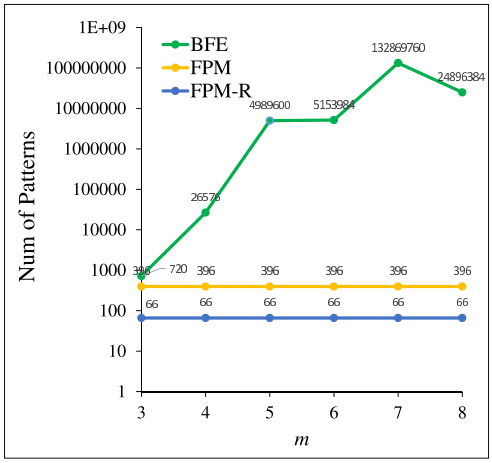

図 5 実験 A4：最小支持度 $m_* = m$ を変化させたときのパターンの個数

Fig. 5 Exp A4: The pattern number by varying minimum support $m_* = m$.

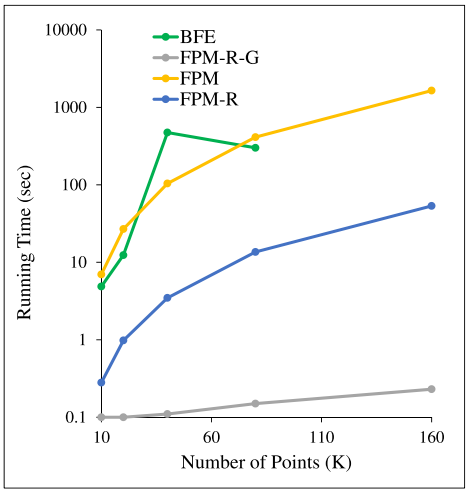

図 6 実験 B1：入力点数 N を変化させたときの計算時間

Fig. 6 Exp B1: The running time by varying number of input points N.

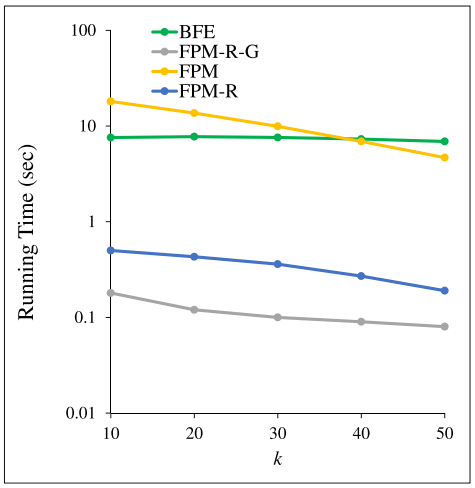

図 7 実験 B2：最小長さ k を変化させたときの計算時間

Fig. 7 Exp B2: The running time by varying minimum length k.

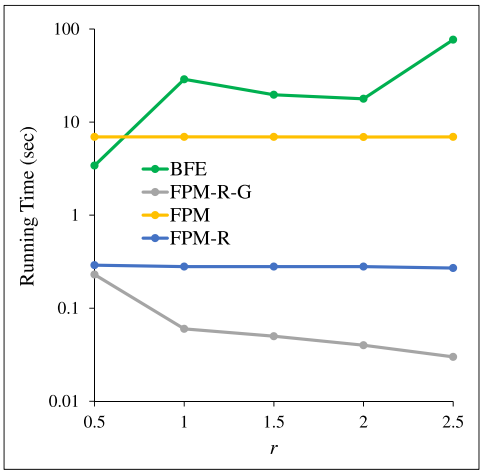

図 8 実験 B3：最大幅 $r_* = r$ を変化させたときの計算時間

Fig. 8 Exp B3: The running time by varying maximum width $r_* = r$.

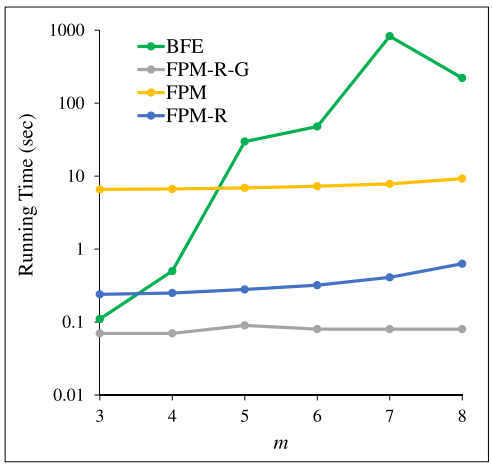

図 9 実験 B4：最小支持度 $m_* = m$ を変化させたときの計算時間

Fig. 9 Exp B4: The running time by varying minimum support $m_* = m$.

© 2015 Information Processing Society of Japan

は命題 1 と定理 2 から予想されるとおりほぼ線形であり，空間索引を用いた FPM-R-G の計算時間はわずかに線形より大きかった．この非線形性は，レンジクエリの領域に含まれる軌跡点数の増加のためと思われる．

アルゴリズムの改良の効果については，初めに，図 6 の入力点数に対する計算時間のグラフから，4 つのアルゴリズムの計算時間を比較すると，入力サイズが $N = 40\,\mathrm{K}$ 点の場合に，BFE の計算時間は約 473.4 秒であり，FPM-R-G は 0.11 秒，FPM-R は 3.47 秒，FPM は 104.1 秒だった．これより，この人工データを用いた実験におけるアルゴリズム間の速度比較として次が結論された．

既存アルゴリズム BFE と基本アルゴリズム FPM の速度を比較すると，提案の基本アルゴリズムである FPM は従来アルゴリズムの BFE より約 4.5 倍高速であった．

基本アルゴリズム FPM に対する改良版のアルゴリズム FPM-R と FPM-R-G の高速化の効果を検証すると，基本アルゴリズム FPM を基準としたとき，右極大群パターンを用いた改良アルゴリズム FPM は約 30 倍高速であり．さらに，空間索引版の FPM-R-G は，約 930 倍高速であった．

まとめると，改良アルゴリズム FPM-R-G は，既存アルゴリズムの BFE と基本版の FPM より，それぞれ，約 4,300 倍と約 1,000 倍高速であった．図 6 のグラフから，空間索引を用いた改良版 FPM-R-G は，16 万点のサイズの軌跡データから 0.23 秒で全右極大群パターンを発見することが観察された．これより，提案アルゴリズムは実応用の意味でも十分に高速といえる．

各種パラメータの変化に対する計算時間の振舞いについては，図 7〜図 9 のグラフから，初めに 4 つのプログラムのパラメータ k，r，m の変化に対して，FPM と BFE の速度は FPM の 2 つの改良版アルゴリズムに比べて低速であり，相互の相対速度はパラメータ値によるが比較不能だった．また k，r，m の値が増大したとき，BFE の計算時間が大きく増大したのに対して，FPM の計算時間はあまり変化しなかった．

次に，図 7〜図 9 のグラフから，右拡張パターンを用いた FPM-R は，パラメータ k，r，m のほとんどの値に対して，安定して FPM より 20 倍〜30 倍程度高速であった．より詳細に見ると，図 7 と図 9 からは，広い範囲の最小長さ k と最小頻度 m の値に対して，FPM-R-G が FPM-R に対して 2.0 倍〜3.5 倍程度高速であり，空間索引が安定した性能を示すことが観察された．さらに特徴的な振舞いとして，図 8 からは，最大幅 r が大きいほど，空間索引による高速化の効果が大きくなることが観察された．

5.5 実験 C：台風軌跡データ上への適用

実データ上での提案アルゴリズムの有効性を評価するために，総サイズ 16,000 点の 320 本の軌跡からなる 5.1.2 項の台風データから，アルゴリズム FPM-R-G で，パラメータとして最小長さ $k = 15$ 点（約 90 時間）と最小軌跡数は $m = 3$ 本で，右極大群パターンを発見する実験を行った．距離の単位は緯度と経度である．結果として，最大幅 $r = 2$ 経緯度（約 200 km）のときに，総計算時間 0.08 秒で 10 個の右極大群パターンを見つけた．参考として，$r = 3.0$ かつ最小長さが $k = 15, 30, 35$ のときには，それぞれ，1442 個と，9 個，4 個の右極大群パターンがどれも 0.2 秒以下で見つかった．

表 3 に，見つけた 10 個の群パターンのうちの 1 つが含む 3 つの台風の規模等の情報を示す．また，**図 10** に，これらの台風の軌跡を示す．図の下方の台風の発生地点において，下から順に軌跡の台風番号は 200603，195504，198712 である[*7]．各円は，6 時間おきのサンプリング点を表しており，群パターンに含まれている地点を濃い色で表す．番号 200603 の台風は，総移動距離は 3,872 km で暴風域の最大直径は 280 km である [10]．

この図から，見つけたパターンは，約 1,000 km の長さと

表 3 発見したパターン P_2 が含む台風の情報．各欄は左から台風番号，発生日，生存日数，移動距離（km），平均風速（km/h）である

Table 3 The information of the pattern P_2. From the left of items row, the items are the ID, birth, lift-time, length of movement (km) and average speed (km/h) of typhoon data.

番号	発生日	日数	距離（km）	風速（km/h）
195504	1955/04/17	10.0	2,705	11.0
198712	1987/08/22	9.5	4,966	21.8
200603	2006/06/30	9.8	3,872	16.6

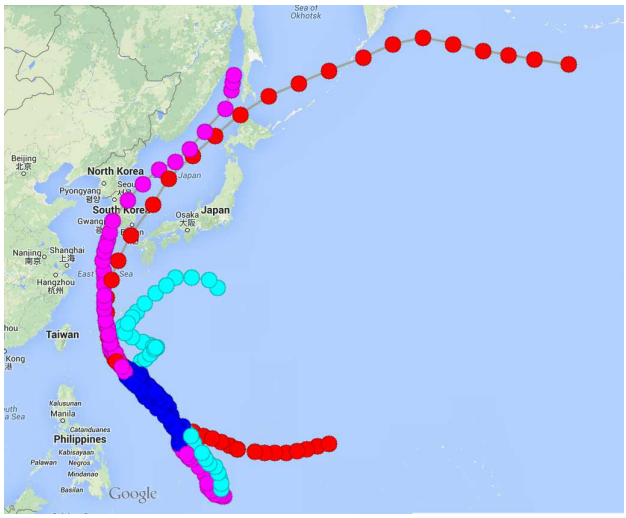

図 10 発見したパターン P_2 を含む 3 つの台風の軌跡．青色の円はパターンの部分で，長さは 15 点で最大幅は 2.0 経緯度である

Fig. 10 There are 3 trajectory data in the pattern P_2 which is tinted by deep blue. Its width is 2.0 latitude and length is 15 points.

[*7] 台風番号 195504 は，それが 1955 年の台風 4 号を表す．

© 2015 Information Processing Society of Japan

約 2.0 経緯度（約 200 km）の幅を持ち，マニラ市から東南東約 1,500 km 東方のセレベス海上から，西向きに移動しながら，同じく東北約 1,000 km 東方海上まで到達するという 3 つの軌跡に共通した特徴を表している．このパターンは右極大なので，同じ台風の組合せと開始時間を持ち，より長い時間継続したパターンは他にはないことが分かる．

またこの実験において，総計 320 本の軌跡が含む 3 つ以上の台風の部分軌跡の組合せは $\binom{320}{3} \times 35^3 = 1.392 \times 10^{12}$ 個という膨大な数になるが，アルゴリズムの網羅性と出力パターン数が 10 であることから，全組合せのうち幅が約 200 km 以内で長さ約 90 時間以上並走し，右極大なものはわずか 10 個しかないと結論できる．

まとめると，提案のアルゴリズム FPM-R-G は，300 本長の複雑な台風経路から，部分軌跡の特徴を表すパターンを網羅的に見つけており，実データに関しても有用であると思われる．

6. まとめ

本稿では，軌跡データからの群パターンマイニング問題を考察し，深さ優先探索型の基本マイニングアルゴリズム FPM と，その改良版である FPM-R と FPM-R-G を与えた．人工データ上での実験結果からは，これらの提案アルゴリズムは既存のアルゴリズム BFE より高速であり，広い範囲のパラメータに対して安定した性能を示すことが分かった．さらに，アルゴリズム FPM-R-G と FPM-R は，FPM より大幅な高速化を示した．台風軌跡の実データ上での実験では，提案アルゴリズムが現実的な時間で，データの共通の特徴を表した群パターンを発見した．

今後の課題として，今回は，アルゴリズムが見つけた群パターンについて，気象や災害対策の観点からみた有用性については，専門家の検証を受けていない．これは今後の課題である．さらにシステム面では，現実の大規模応用のために，Hadoop 等の大規模並列環境での群パターンマイニングの実現は重要な課題である．また，より誤差や欠損を多く含んだ軌跡データを扱うための頑健な群パターンの導入と，マイニング手法の開発も興味深い．

参考文献

[1] Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules in large databases, *Proc. 20th International Conference on Very Large Data Bases*, pp.487–499, Morgan Kaufmann Publishers Inc. (1994).

[2] Arimura, H. and Uno, T.: Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems, *Proc. SIAM Int'l Conf. on Data Mining 2009* (*SDM'09*), pp.1087–1098 (2009).

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

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

[5] Buchin, K., Buchin, M., van Kreveld, M., Speckmann, B. and Staals, F.: Trajectory grouping structure, *Proc. WADS'13*, Vol.8037, pp.219–230, Springer (2013).

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

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

[8] Fort, M., Sellarès, J.A. and Valladares, N.: A parallel gpu-based approach for reporting flock patterns, *International Journal of Geographical Information Science*, Vol.28, pp.1–27 (2014).

[9] Gudmundsson, J. and van Kreveld, M.: Computing longest duration flocks in trajectory data, *Proc. ACM GIS '06*, pp.35–42, ACM (2006).

[10] Kitamoto, A.: Digital typhoon, NII, available from ⟨http://agora.ex.nii.ac.jp/digital-typhoon/⟩ (2014).

[11] Laube, P., van Kreveld, M. and Imfeld, S.: Finding REMO – detecting relative motion patterns in geospatial lifelines, *Spatial Data Handling*, pp.201–215, Springer (2005).

[12] Megiddo, N.: Linear programming in linear time when the dimension is fixed, *J. ACM*, Vol.31, No.1, pp.114–127 (1984).

[13] Parsa, S.: A deterministic $O(m \log m)$ time algorithm for the Reeb graph, *Discrete & Computational Geometry*, Vol.49, No.4, pp.864–878 (2013).

[14] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.-C.: Mining sequential patterns by pattern-growth: The prefixspan approach, *IEEE TKDE*, Vol.16, No.11, pp.1424–1440 (2004).

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

[16] Uno, T., Asai, T., Uchida, Y. and Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases, *Proc. DS'04*, Vol.3245 of *LNCS*, pp.16–31, Springer (2004).

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

[18] Zaki, M.J.: Scalable algorithms for association mining, *IEEE Trans. Knowledge and Data Engineering*, Vol.12, No.3, pp.372–390 (2000).

[19] Zaki, M.J. and Hsiao, C.-J.: Efficient algorithms for mining closed itemsets and their lattice structure, *IEEE TKDE*, Vol.17, No.4, pp.462–478 (2005).

推薦文

本分野の整合性が高く，また新規性・有用性・信頼性についても申し分ない．よって，ここに推薦する次第である．

（FIT2013 第 12 回情報科学技術フォーラム
プログラム委員長　荒川賢一）

© 2015 Information Processing Society of Japan

耿 暁亮

1986 年生．2009 年中国東北大学ソフトウェア学院ソフトウェア工学科卒業．同年北海道大学大学院情報科学研究科研究生を経て，2012 年北海道大学大学院修士課程修了．同年北海道大学大学院情報科学研究科博士後期課程入学．2010 年から地理情報マイニングの研究に従事．

宇野 毅明 （正会員）

1998 年 3 月東京工業大学大学院総合理工学研究科博士課程修了，博士（理学）を取得．1998 年 4 月東京工業大学経営工学専攻助手着任，2001 年 2 月国立情報学研究所助教授着任．2005 年 5 月より 2006 年 8 月までスイス連邦工科大学に滞在．現在，情報学プリンシプル研究系教授．日本オペレーションズリサーチ学会，電子情報通信学会に所属．専門はアルゴリズムの理論と応用，特に離散アルゴリズム，列挙アルゴリズム，計算量理論，組合せ最適化等．データマイニング・データ解析・ゲノム情報学では，クラスタリングや類似性等の基礎計算を大規模データで高速に行う手法を研究．2010 年文部科学大臣表彰科学技術部門若手科学者賞受賞．

有村 博紀 （正会員）

1988 年九州大学理学部物理学科卒業．1990 年九州大学大学院修士課程修了．同年九州工業大学助手，同助教授等を経て，1996 年九州大学大学院助教授，2004 年から北海道大学情報科学研究科教授，現在に至る．博士（理学）．1999～2002 年 JST さきがけ研究 21 研究員．2007～2011 年文科省 GCOE プログラム「知の創出を支える次世代 IT 基盤拠点」拠点リーダ．現在，データマイニングと，情報検索，アルゴリズムの研究に従事．ACM，電子情報通信学会，人工知能学会の各会員．

© 2015 Information Processing Society of Japan