RA-004

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

Practical Algorithms for Mining Flock Patterns from Trajectories

耿　暁亮 †　　宇野　毅明 ‡　　有村　博紀 †
Xiaoliang Geng　　Takeaki Uno　　Hiroki Arimura

† 北海道大学大学院情報科学研究科
Hokkaido University
{gengxiaoliang,arim}@ist.hokudai.ac.jp

‡ 国立情報学研究所
National Institute of Informatics
uno@nii.jp

概要

群パタンは，Gudmundsson と van Kreveld ら (Proc. ACM GIS 2006) によって提案された時空間パタンであり，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，群パタン発見のための新しい深さ優先探索型アルゴリズムを実装し，その実験的評価を行う．人工データ上の実験では，二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した．

1　はじめに

1.1　背景

移動体デバイスと位置センサー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．**群パタン** (flock pattern) は，Gudmundsson と van Kreveld ら (Proc. ACM GIS 2006) によって提案された時空間パタンであり，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 L_∞-距離 r 以内で，一緒に移動する様子を表す．

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全ての群パタンをもれなく発見する問題を考察する．このような群パタンの発見は、野生動物や歩行者の移動に関する GPS データの解析や、移動車両などのプローブカーデータからの知識発見に有用である．

そこで本稿では，効率よいパタン発見手法として知られているパタン成長アプローチ (Pei et al., 2004) に基づいて、群パタン発見のための効率よい深さ優先マイニングアルゴリズムを与える．頻出集合や系列のマイニング分野では，PrefixSpan (Pei et al., 2004) や LCM (Uno et al., 2004) 等のパタン成長に基づく高速なアルゴリズムが提案されている．一方で，群パタンマイニングにおいては，未研究である．

1.2　先行研究

現在までに，いくつかの群パタン検索アルゴリズムが提案されているが (Benkert et al., 2008; Gudmundsson and van Kreveld, 2006; Laube et al., 2005)、データマイニング分野で盛んに研究されてきたパタン列挙に基づく研究は、少数である．(Vieira et al., 2009) は，点集合のクラスタを組合せる方式を提案しているが，発見したパタンを保存し，レンジクエリを用いて重複除去を行っている．(Romero, 2011) は，群パタン発見を前処理を用いて頻出集合発見に帰着する．どちらの研究も，一時メモリを用いるため大きなメモリを要し，低メモリの深さ優先探索方式ではない．

1.3　主結果

本稿では，この群パタン発見のための新しい深さ優先探索型アルゴリズム FPM を実装し，その実験的評価を行う．二つの高速化手法として，右極大な群パタンを用いた改良版プログラム RFPM と，空間索引を用いた手法による改良版プログラム G-RFPM を与える．

前者は、それが含む移動物体の集合を保存したまま，パタンを可能な限り右側に延長して得られる一種のクローズドパタンを発見する手法であり，パタン数を軌跡長程度の因数で削減することで高速化を行う．後者は，2 次ユークリッド平面上の距離制約を利用し，4 分木や領域木等の空間索引を用いて、群パタンに追加する物体の候補をあらかじめ絞り込む手法である．

人工データ上の実験では，この二つの手法を組み合わせることによって，元の FPM に対して、改良版プログラムの G-RFPM は合わせて約 2000 倍の高速化を達成した．

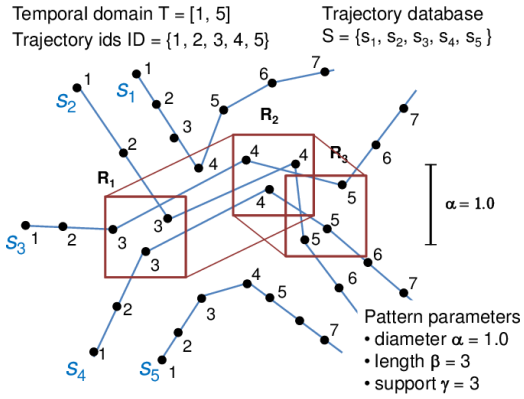

図 1　例：軌跡データベース S_1 である．$ID = \{1,\ldots,5\}$ と $\mathbb{T} = [1,7]$ と $(1.0,2,2)$-フログパタン $P_1 = (X_1, I_1) = (\{2,3,4\},[3,5])$ になる．幅 $||P_1||_\infty^{S_1} \leq 1.0$，長さ $len(P_1) = 3$ と支持度 $supp(P_1) = 3$．　ここで線は軌跡データに代表している．点の隣の数字は時間印である．

1.4　本稿の構成

本稿の構成は次の通りである．2 節ではフログパタンの定義する．3 節では提案手法を述べる．4 節では実験の結果をのべる．最後に，5 節で本稿をまとめる．

2　準備

2.1　基礎定義

$\mathbb{R}$ と $\mathbb{N}$ それぞれは実数と非負整数の集合である．整数 a, b $(a \leq b)$ に関し，$[a,b] = \{a, a+1, \ldots, b\}$ は a から b までの離散区間である．まだ，$a \leq b$ は実数の場合，$[a,b]$ は $\mathbb{R}$ に連続的な区間である．集合 A に対して，$|A|$ は A の中の元素数を表す．A^* は空集合も含まれる全集合と集合 A の有限数列を表す．

2.2　軌跡 (Trajectory) データベース

非負整数 n と $T \geq 0$ は移動物体数と離散時間印の最大値を表す．$\mathbb{R}^2$ は二次元平面を表す．

時間範囲 $\mathbb{T} = \{1,\ldots,T\}$ 上の軌跡データベース (*trajectroy database*) は n 個移動物体 $o_1,\ldots,o_n$ の軌跡データの有限集合

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

である．ここに，インデックス $i = 1,\ldots,n$ に対し，i は軌跡データの ID である．n 個の ID の集合は $ID = \{1,\ldots,n\}$ である．第 i 番目の軌跡 s_i は二次元 $\mathbb{R}^2$ に T 点集合の列

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

である．第 i 番点は $s_i[t] = (x_{it}, y_{it}) \in \mathbb{R}^2$.

例 1　図 1 には，軌跡データベース S の例を示している．長さが 7 の 5 本の軌跡データがある．

GPS センサーを持っている野生動物や通行者などの GPS の軌跡データベースを構成できる．

2.3　群パタンのクラス

定義 1（FP）　$\mathbb{T}$ に対する群パタンはペア $P = (X,[b,e])$ である．ここでは，$X \subseteq ID$ は ID の有限集合である．P の *ID set* と呼ばれる．$I = [b,e]$ は $[0,T]$ の離散区間である．そして $b \leq e \leq T$，b と e は P の *start* と *end time* と呼ばれる．

以下のように群パタンの支持度 (support)，長さ (length) と幅 (width) を定義する．

P の支持度 (*support*) は $supp(\mathrm{P})$ に表し，X 中の軌跡データ (ID) の個数である．$supp(P) = |X|$．P の長さは $len(P)$ で表して，I の幅である．$len(P) = e - b + 1$.

$0 \leq supp(P) \leq n$ と $0 \leq len(P) \leq T$ を限定しているということを明らかにわかる．

例 2　図 1 には，群パタンの例 $P_1 = (X_1, I_1)$．ID 集合は $X_1 = \{2,3,4\}$，時間区間は $I_1 = [3,5]$ である．

群パタンの幅を定義するために，様々な定義が必要である．二次元 $\mathbb{R}^2$ 平面では，点 $p = (x,y)$ の x と y の座標は $p.x = x$ と $p.y = y$ である．点 p と p' の距離 L_∞-*distance* は $L_\infty(p,p') = \max\{|p.x - p'.x|, |p.y - p'.y|\}$ を定義する．この定義によって，$L_\infty(p,p')$ は非負である．$p = p'$ にすると，零になる．

点集合 $A = \{p_1,\ldots,p_n\}$ の直径は $||\,A\,||_\infty$ で表させ，A の中の毎二つ点の最大距離と定義される．

$$||\,A\,||_\infty = \max_{p,p' \in A} L_\infty(p,p'), \qquad (2)$$

直径 $||\,A\,||_\infty$ は非負である．二次元 $\mathbb{R}^2$ 平面では，$||\,A\,||_\infty$ が線形時間で計算できる．$\mathbb{R}^d$ 区間で任意な $d \geq 2$ に $||\,A\,||_\infty$ が $O(dn)$ で計算できる．d が定数の時，計算時間も線形になる．

入力データベース S に，第 t 番目の点の集合は $S[X][t]$ で表して，X 中の軌跡データの第 t 番目の点の集合を定義する．群パタン $P = (X,I) = (X,[b,e])$ の幅 *width* $||P||_\infty^S$ は $S[X][t]$ の直径の最大値を定義する．t は $t \in [b,e]$ になる．

実際に，以下のようの定理がある．

定理 1　P の支持度は $m = supp(X)$，長さは $\ell = len(P)$ であり，P の幅 (width) はアルゴリズム 1 が $O(m\ell)$ の時間に計算できる．

正数 $r > 0$ とし，$k, m \geq 0$ は非負数とする．r, k, m は *maximum width* (max-width)，a *minimum length* (min-len) と a *minimum support* (min-sup) 関数にな

Algorithm 1 データベース $S = \{ s_i \mid i = 1, \ldots, n \}$ に幅 $||P||_\infty{}^S$ の群パタン $P = (X, [b, e])$ の探索アルゴリズムである．

```
1: width ← 0;
2: for t ← b, b + 1, ..., e do
3:     S_t ← { s_i[t] | i ∈ X };          ▷ the t-th slice
4:     width ← max{width, ||S_t||_∞};
5: return width;
```

る．以下は定義である．$||P||_\infty \leq r$ にすると，*r-flock pattern* は任意な r-群パタン P になる．$||P||_\infty \leq r$ が軌跡データベース S にあると考える．$len(P) \geq k$ にすると，群パタン (r, k) は任意な r-群パタン P.

例 3 図 1 のパタン P_1 の直径は $||P_1||_\infty{}^{S_1} \leq 1.0$，長さ $len(P_1) = 3$ である．支持度 $supp(P_1) = 3$．そして，群パタン $(1.0, 2, 3)$ は $r = 1.0$，$k = 2$ と $m = 3$ になる．

本稿では，全部の (r, k)-群パタンは入力される軌跡データベースに含まれることを考える．

2.4 右拡張最大長さパタン

パラメータ $r > 0$，できるだけ右に拡張することは仮定した r に (r, k)-**フロックパタン** $P = (X, [b, e])$ を探索するために，よく役に立つ (Gudmundsson and van Kreveld, 2006).

$P = (X, [b, e])$ は幅を変わらない場合に時間帯に右に拡張できなければ，右拡張最大長さパタンと言える．

正式に以下のように定義する．

定義 2（RFP） S に群パタン $P = (X, [b, e])$ は右拡張最大長さパタン (*rightward length-maximal* 群パタン, (RFP)) である．もし，他のパタン $P' = (X, [b, e'])$ が S に存在しなく，(i)P' が P と同じな ID 集合 X があり，(ii)P' の右辺末尾は P より大きである．

この定義によって，S 中に任意な RFP は FP である．一般的には，逆になりたたない．そして，$\mathcal{RFP}(r, k) \subseteq \mathcal{FP}(r, k)$ の包含が成り立つ．

例 4 図 1 には，長さ 3 の群パタン $P_1 = (X_1, [3, 5])$ は S_1 に RFP である．$P_2 = (X_1, [3, 4])$ と $P_3 = (X_1, [3])$ は非右拡張最大長さパタンの FP である．$X_1 = \{2, 3, 4\}$ である．P_1 が RFPs $P_4 = (X_1, [4, 5])$ と $P_5 = (X_1, [5])$ を含む．

2.5 データマイニング問題

任意な $\mathcal{C} \in \{\mathcal{FP}, \mathcal{RFP}, \ldots\}$ とパラメータ $r, k \geq 0$ を与え，全部のクラス $\mathcal{C}$ 中に (r, k)-群パタンのクラスを $\mathcal{C}(r, k)$ に定義する．同様にクラス $\mathcal{C}(r)$ と $\mathcal{C}(r, k, m)$ も定義する．これから，クラス $\mathcal{FP}(r, k)$ とクラス $\mathcal{RFP}(r, k)$ を考える．

マイニング問題については以下のように述べる．

定義 3 (FLOCK PATTERN MINING PROBLEM FOR PATTERN CLASS $\mathcal{C}$) **$\mathcal{C}$ を群パタンのクラスさせる．入力は軌跡データベース S の組み (S, r, k) とパラメータの r と $k \geq 0$ である．任務は S の中にクラス $\mathcal{C}$ の全部の群パタン P を重複なしに最大幅 r までと最短長さ k から探索することである**.

同様にパラメータ (r, k, m) とマイニング問題を考える．

群パタンマイニングアルゴリズム $\mathcal{A}$ 性能を列挙アルゴリズム (Avis and Fukuda, 1993) によって評価する．N と M は入力サイズと解のパタン数になる．もし延期は二つ連続出力の間の最大計算複雑度にすると，パタンマイングアルゴリズム $\mathcal{A}$ の延期は多項式時間になる．N に多項式 $p(N)$ に限定している．

3 提案手法

本節では，FPs と RFPs を探すための提案手法を導入する．更に地理インデックスを使用して過剰的なパタンを除去する技術を導入する．

3.1 FPs のための基本 DFS アルゴリズム

先に FPs ための基本的なマイニングアルゴリズム FPM (basic 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．コードは省略する．

DFS アプローチに再帰的マイニング手続きは小さいパタンから大きいパタンまでにパタンの全部的な子孫を引き返し深さ優先に探索する．DFS アプローチの有利な点は DFS マイナーが主記憶で性能が速くなることを証明できる．そして簡単的な再帰的手続きを実現するコストが安いである．

FPM をトップレベルにそれぞれの可能な k より長い長さ $\ell \in [k, T]$ に，初期 ID 単集合 $X_0 = \{i_0\}$ と時間帯 $[b_0, e_0]$ から成り立った $P_0 = (X_0, [b_0, e_0])$ を変数として再帰的副手続き RecFPM に代入する．終わる時間 e_0 は b_0 と ℓ により計算する．

再帰的副手続き RecFPM は長さがちょうど ℓ の全部的な r-群パタンの探索空間を以下のように探索する．

単集合 ID$X_0 = \{i_0\}$ から構成した初期のなパタン $P_0 = (X_0, [b_0, e_0])$ から初めて，手続きは頻出アイ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 LCM (Uno et al., 2004) のように全部の ID 部分集合 X を列挙する．それぞれの部分集合 X を生成するために，この手続きは候補 (r, k)-群パタン $P = (X, [b, e])$

と時間帯 $[b,e]$ を生成する．そして，アルゴリズムは S 中の軌跡テータをアクセスすることでパタン P の幅 $||\,P\,||_\infty$ を計算する．後から P が幅の制限 $||\,P\,||_\infty \leq r$ を満足するかどうかチェックする．もし条件に反すると，P とその全部の子孫の探索をやめる．

この幅によって取り除く規則の正当性は以下の定理にされる．

定理 2（反単調性） $P_1 = (X_1, I_1)$ と $P_2 = (X_2, I_2)$ を二つ群パタンとする．P_2 が S 中の (r,k)-群パタンであり，$X_1 \subseteq X_2$ と $I_1 \subseteq I_2$ が成り立つならば，P_1 も S 中の (r,k)-群パタンである．

この定理より，候補パタン $P = (X, I)$ は幅と長さの条件に反すると，P に新軌跡データを加えてできた候補パタンも幅と長さの条件に反する．したがって P の (r,k)-群パタンための部分の探索空間が取り除ける．

アルゴリズム FPM の時間と空間の計算量は以下の命題が本稿 (Arimura et al., 2013) で簡単に得る．

命題 1 (Arimura et al., 2013) S を長さが T の軌跡テータ n 個とする．アルゴリズム FPM が S 中の全ての (r,k)- 群パタンを漏れ無く，重複なく出力する．このアルゴリズムの一つパタンに対する探索時間計算量が $O(knT^2)$ であり，空間計算量が $O(k^2)$ である．$k = supp(X) = |X|$ は列挙されるパタン X の支持度である．

次に RFPMs のマイニングに注目して最終的な計算量を示す．

3.2 RFPs を探索するための改良したアルゴリズム

次に RFPs(右拡張最大長さ群パタン) 探索するために改良したアルゴリズム RFPM(右に向き群パタンマイナー) と右拡張最大長さ群パタンクラスを導入する．アルゴリズム 3 に，アルゴリズム RFPM と (r,k)-RFPs ための RFPM の副手続き RecRFPM を示す．

3.2.1 右向き水平的閉包

頻繁パタンマイニングの見方によって，軌跡データベースに RFPs は**閉パタン**の一種である．**閉パタン**は頻出アイテム集合マイニング (FIM) (Uno et al., 2004; Zaki and Hsiao, 2005) の分野と formal concept analysis (FCA) の分野に広く研究された．多数の高効率閉パタンマイニングアルゴリズムは閉包作業という作業のクラスを使用している．

定義 4（右向き水平的閉包） 軌跡データベース S 中に，$P = (X, I = [b,e])$ は任意なパタンとする．P の**右向き水平的閉包**は RH_Closure$(P; S, r)$ と定義する．それは唯一のパタン $P_{max} = (X, I = [b, e_{max}])$ である．

Algorithm 2 唯一の右向き最大長さ群パタンをマイニングする．$t \notin [1,T]$ おいて $||\,S[X][t]\,||_\infty$ は ∞ と定義される．

```
1: procedure RH_Closure((X, [b0, e0]); S, r)
2:     t ← b0;
3:     while || S[X][t] ||∞ ≤ r do
4:         t ← t + 1;
5:     b ← b0; e ← t − 1;
6:     return (X, [b, e]);
```

$e_{max} \in [0,T]$ は尾の位置 e' の最大値である．以下の公式を満足している．

$$||\,P' = (X, [b, e'])\,||_\infty = ||\,P\,||_\infty. \tag{3}$$

右向き水平的閉包操作は尾の位置だけを変えている．最初の P の ID 集合もしくは開始時間 b を完全に変えない．アルゴリズム 2 に RH_Closure の手続きを示す．その手続きは non-RFP P の右向き水平的閉包は $O(k\ell)$ 時間で計算することである．$k = supp(P) = |X| = O(n)$ と $\ell = len(P_{max}) = O(T)$ になっている．

次の定理が右に向き水平的な閉包の正当性を示す．先に RFPs の定義と性質から正当性の要点を示す．

定理 3（characterization） S に $P = (X, [b,e])$ が (r,k)-群パタンに仮定する．そすれば，P が右に向け最長になることが成り立つのは以下のように

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

が成り立つ時，またはその時に限る．ここで，$t < 1$ または $t > T$ が成り立つ時，第 t 番目の $||\,S[X][t]\,||_\infty$ は ∞ まで拡張されている．

上の定理より，正当性は下に示す．

定理 4 (Arimura et al., 2013)

非右向き最長 r-群パタン P の右向き水平的な閉包は唯一の最長 r-RFP である．ここで ID 集合と開始時間は P の同一である．

P_{max} に $len(P_{max}) \geq len(P)$ よく成り立つから，もし P が (r,k) 条件に満たすと，得った RFP P_{max} は満足している．したがって，P_{max} は P の唯一の最長 (r,k)-RFP 番である．ID 集合と開始時間は P と同じである．

3.2.2 合併

アルゴリズム RFPM の計算を導入する．RFPM の全体の構造は基本アルゴリズム FPM と同じである．軌跡

Algorithm 3 軌跡データベース S の ID 中に，全部の最大長さ (r,k)-群パタンを探索する．ペラミータは最大幅 r と最小長さ k である．

```
1: procedure RFPM(ID, S, r, k)
2:     for b_0 ← 1, ..., T do   ▷ Each start time in 𝕋
3:         for i_0 ← 1, ..., n do        ▷ Each id in ID
4:             P_0 = ({i_0}, [b_0, *]);
5:             RecRFPFM(P_0, ID, S, r, k);
6: procedure RecRFPM((X, [b, *]), ID, S, r, k)
7:     P = (X, [b, e]) ← RH_Closure((X, [b, *]); S, r);
8:     if len(P) < k then
9:         return ; ▷ P is not an (r, k)-flock pattern
10:    output P;
11:    ID_1 ← ID;
12:    while ID_1 ≠ ∅ do
13:        i = deletemin(ID_1);
14:        P_1 = (X ∪ {i}, [b, *]);
15:        RecRFPM(P_1, ID_1, S, r, k);
16:    end while
```

データベース S を入力して，主アルゴリズム RFPM は初期パタン P_0 を初期パンタとして，再帰的副手続きの RecFPM とを呼び出す．その計算は以下のステップである．

- 変数として RFP$P_* = (X, b, *)$ を代入して，再帰的手続き RecFPM が P_* から右に向け水平的な閉包 $P = (X, [b, e])$ を最大幅 r と RH_Closure で計算する．
- 得った RFPP は (r, k) 条件に満足していると，出力する．逆に満たしていない場合は，P とその子孫は探索から除去する．
- 最終的に RecFPM は再帰的に拡張されたパタン $P_1 = (X \cup \{i\}, [b, *])$ を RecFPM に代入する．重複パタンを出さないように i は ID から消す．

関連研究 (Uno et al., 2004) は (Avis and Fukuda, 1993) の逆探索技術に基づいて繰り広げる．そこから，以下のように RFPM の時間と空間の計算量を示す．

定理 5 (Arimura et al., 2013)

S を長さが T の n 個の軌跡データのデータベースとする．アルゴリズム 3 の RFPM が S に (r,k)-群パタンの $\mathcal{RFPM}(r,k)$ クラスに対するマイニング問題を解決する．一つパタンに対する時間は $O(knT)$ であり，空間は $O(k^2)$ 文字である．ここで列挙されたパタン X の支持度は $k = supp(X) = |X|$ である．

Algorithm 4 データベース S の ID 中にアルゴリズム G-RFPM が最大長さ (r,k)-群パタンを探索する．ペラミータは最大幅 r と最小長さ k である．

```
1: procedure G-RFPM(X, b, k, ID, S, r, k)
2:     Let S = { s_i | i = 1, ..., n };
3:     for b_0 ← 1, ..., T do   ▷ Each start time in 𝕋
4:         Build a grid index for point set U ← S[b_0];
5:                         ▷ The time slice at time b_0
6:         for i_0 ← 1, ..., n do        ▷ Each id in ID
7:             p ← s_{i_0}[b_0];   δ ← r;    ▷ initial point p
8:             R ← [p.x−δ, p.x+δ]×[p.y−δ, p.y+δ];
9:                 ▷ 2r×2r-query rectangle at center p
10:            ID_0 ← U.Range(R);
11:            P_0 ← (X = {i_0}, [b_0, *]);
12:            RecRFPM(P_0, ID_0, S, r, k);
```

$d \geq 1$ と $O(d)$ の時間と空間に対する RH_Closure を $\mathbb{R}^d$ に修正して，d-次元空間 $\mathbb{R}^d$ に RFPM を適用できる

3.3 地理インデックスを使用する高速化

本節では，$\mathbb{R}^2$ の地理インデックスを使用した高速化技術を導入する．アルゴリズム 4 に，改良したアルゴリズム G-RFPM (grid-based flock pattern miner) と副手続 RecFPM を示す．これは (r,k)-群パタンのための RFPM に基づいて改良したものである．アルゴリズム G-RFPM は，軌跡データベース S と，最大幅 $r > 0$，最小長さ k を入力変数として受け取り，各開始時間 $\mathbb{T} = [1, T]$ に b_0 を選び，計算を開始する．

まず前処理として，軌跡データベース S 中の全ての点を探して，時刻 $t = b_0$ をもつ点を集めた集合 $U = \{\, p = s_i[t] \,|\, s_i \in S, i \in ID, t = b_0 \,\}$. を求める．次に，各軌跡 id $i_9 = 1, \ldots, n$ に対して，単一集合 $X = \{i_0\}$ を初期値として，マイニングを始める．これに関連する軌跡 s_{i_0} の開始点は $c = s_{i_0}[b_0] \in \mathbb{R}^2$ である．このとき，$P = (X, [b, e])$ を，s_{i_0} を含む任意な目標パタン (r,k)-群パタンとする．そのような目標パタンが含む軌跡の開始点はすべて，点 c を中心とする平面上の $2r \times 2r$ の長方形

$$R = [x-\delta, x+\delta] \times [y-\delta, y+\delta] \subseteq \mathbb{R}^2, \qquad (4)$$

中にある．ここで，$x = c.x$, $y = c.y$ と $\delta = r$ である．したがって，X ための候補 IDs の元領域は以下の小部分領域になる．$ID(R) = \{\, i \in ID \,|\, p_i = s_i[b_0] \in U \cap R \,\}$. 適切な地理インデックスを例えば，四分木もしくはレンジ木などを使用し，レンジクエリで $ID_0 = ID(R)$ を計算できる．$ID(R) = U.Range(R)$ 計算時間は四分木で $q = O(\log^2 \sigma)$ になり，レンジ木で $O(\log^2 n)$ になる．ここに，$\sigma = ||\, U \,||_\infty$ は点集合 U の直径である．

表 1　実験のパラメータとそのデフォールト値

名前	記号	サイズ（default）
領域サイズ	a	40.0
軌跡数	n	100
軌跡長	T	200
埋込みパタン数	K	6
埋込みパタン長	L_*	20–200
埋込みパタン幅	r_*	1.0
埋込みパタン頻度	C	5
発見パタン長 (最小)	k	20–200
発見パタン幅 (最大)	r	1.0
発見パタン頻度 (最小)	m	5

四分木のとき，G-RFPM 全体の計算量は，前処理が $O(N \log^2 n)$ 時間で，パターン 1 個あたり $O(knT)$ 時間である．理論的には RFPM の時間と同じだが，節 4 の実験では，改良した G-RFPM は時間計算量を非常に減らしていることを示す．

4　実験

人工データで実験し，アルゴリズムの効率を測る．

4.1　データ

以下のように植え込み人工軌跡データを C++ のデータ発生機で作る．実験のパラメータとそのデフォールト値を表 1 に示す．表中でパラメータは，上から順に，データベースと，埋め込みパタン，マイニングするパタンのものである．初めに，大きさ $a \times a$ の領域 $\mathbb{A}$ を確定し，同一分布で n 個の長さ T の軌跡テータの集合を作成する．次に，各 C 個の軌跡テータをもつ K 個のランダムパタンを前に作ったデータベースに植えこむ．パタンの長さは L_* にし，軌跡の位置はランダムに幅 r_* 以内にする．その他のパラメータは実験より違っている．

4.2　実験手法

C++ で FPM (BFPM), RFPM (BFPM R) と G-RFPM (GFPM R) of Sec. 3 を実装し，地理インデックス (grid-based geometric index) も実装した．ここで平面は $b \times b$ 個の房 (cells) に割っており，房を定数時間のランダムアクセスで探索する．ここで $b = 5$ にする．以上のプログラムは GNU の g++ 4.6.3 版にコンパイルした．我々の実験は，Intel(R) Xeon(R) CPU E5-1620， 3.60GHz，32GB のメモリ，Ubuntu Linux, version 12.04 の PC で行った．

実験では，一部のパラメータを変化させて，縦軸に総計算時間とパターン数を示した．埋め込みパタンは，パタンの個数 K と，幅 r, 頻度 C は固定になり，パタン長 L は変化させた．

表 2　実験 1a と 2a で RFPM が探し出したパタン数の見積り f^{est} と実測値 f^{exp}，および再現率

K	k	L	f^{est}(個)	f^{exp}(個)	再現率 (%)
6	20	200	1086	1074	98.9
6	20	180	966	954	98.8
6	20	160	846	834	98.6
6	20	140	726	714	98.3
6	20	120	606	594	98.0
6	20	100	486	474	97.5

4.3　実験 A : 右拡張最大長群パタンの高速化

本節には，RFPs(右拡張最大長フログパタン) と FPs(orginary flock patterns) を比べてマイニング効率を実験した．この目的のために結果の数と RFPM (BFPM R) と FPM (BFPM) の計算時間を測った．

実験 1a: 図 2 に RFPM と FPM の計算時間と探したパタン個数を示す．パラメタ $T = L* = 200$ を固定し，入力点数は 12K から 20K までである（$n = 60 \sim 100$）．アルゴリズム RFPM は $K = 6$ 個のパタンを探し出し，FPM は n によって大量のパタンを探し出した．

実験 2a: 図 3 に RFPM と FPM の計算時間と探したパタン個数を見せている．T は 100 から 200 まで，$L_* = 20$ である．

実験 3a: 図 4 に RFPM と FPM の計算時間と探したパタン個数を見せている．$T = 200$ と $L_* = 20 \sim 100$ である．パタンの個数は最小長 L_* により，変わっている．例えば，$K = 20$ の場合に FPM と RFPM の計算時間は約 360 (秒) と 1.44 (秒) である．RFPM は約 250 倍に速くなっている．探したパタンは 594 と 6 になって，RFPM は約 100 倍に減じっている．

実験 4a: 図 5 に RFPM と FPM の計算時間と探したパタン個数を見せている．$T = 200$ と $L_* = 20$ である．パタンの個数は最小支持度 m 6 から 10 までにより，変わっている．この実験は人工データのパタン支持度は 10 にしている．

実験 A のパタンの再現性:

実験 1a と実験 2a において，RFPM と FPM が正しくパターンを探し出しているか検証する．埋込みデータ中に出現する RFP のパタン数は $f_{RFP} = K = 6$ なので，RFPM の実測値 $f^{exp}_{RFP} = 6$ と一致している．

同じく FP のパタン数の見積りは，$f_{FP} = K \cdot (L - k + 1)$ で与えられる．表 2 に，これから計算した予測値 f^{est}_{FP} と実測値 f^{exp}_{FP} を示す．再現率は比 f^{exp}/f^{est} である．これより，FPM はほとんどのパタンを探し出している．[*1]

[*1] この減少分は，埋込みパタン 1 個あたり 2 個分である．軌跡の

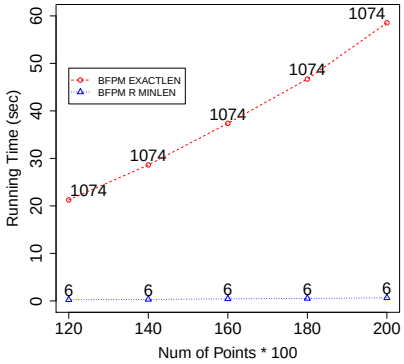

図 2　実験 1a: FPM (BFPM) と RFPM (BFPM R) の計算時間とパタン数である．

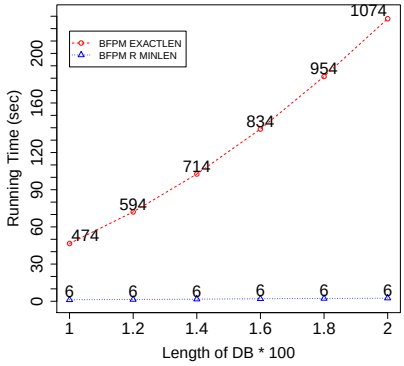

図 3　実験 2a: FPM (BFPM) と RFPM (BFPM R) の計算時間とパタン数である．

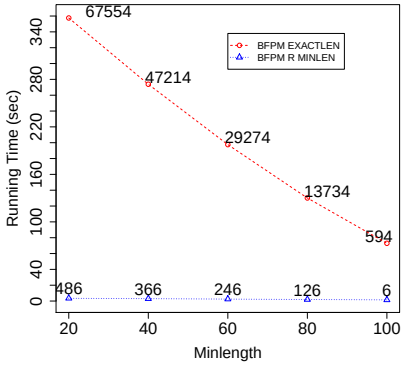

図 4　実験 3a: FPM (BFPM) と RFPM (BFPM R) の計算時間とパタン数である．

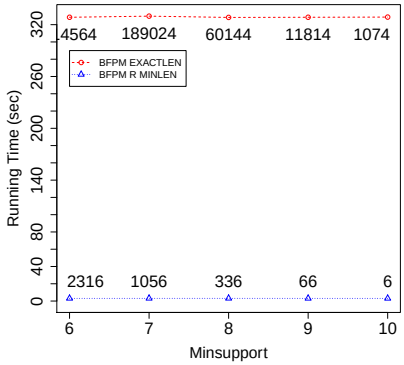

図 5　実験 4a: FPM (BFPM) と RFPM (BFPM R) の計算時間とパタン数である．

実験 A のまとめ: すべての実験を考えて RFPM は非常に FPM より速くなっている．探したパタン RFPs 数は FPs の最多値より 100 倍分の 1 に減らしている．特に RFP は長い軌跡データと小さい最小支持度フログパタンに効率が良い．(Arimura et al., 2013) の道理と同じである．

4.4　実験 B: 地理データベース減少の高速化

本節には，地理データベース減少技術で高速化を実験的に評価する．目的はデータベースに全部の RFPs をマイニングすることである．アルゴリズム RFPM(BFPM R) とアルゴリズム G-RFPM(GFPM R) を比べている．実験に探したパタン数が同じである．

実験 1b: 図 6 に RFPM と G-RFPM の計算時間と探したパタン個数を見せている．入力点数は 20K から 200K までで軌跡データ本数は 200 である．例えば入力点数は 200K の場合に RFPM と G-RFPM の計算時間は 61.61(秒) と 0.96(秒) になっている．約 70 倍に速くになっている．

実験 2b: 図 7 に RFPM と G-RFPM の計算時間と探したパタン個数を見せている．入力軌跡データ長さは 0.2K から 1K 点数までになっている．軌跡データ本数 $n = 200$ になっている．

実験 3b: 図 8 に RFPM と G-RFPM の計算時間と探したパタン個数を見せている．最小長 k は 20 から 100 までになっている．

実験 4b: 図 9 に RFPM と G-RFPM の計算時間と探したパタン個数を見せている．最小支持度 m は 6 から 10 までになっている．

実験 B のまとめ: すべての実験 B より，RFPs をマイニングするために，地理データベース減少技術の G-RFPM は 10 から 70 倍まで RFPM より速くなっている．実際には G-FPM 全部の RFPs を探索する時間 1 秒より少なになっていり，データベースの点数は 20 万であり，実際の運用に結構である．

実験 A と B のまとめ: 実験 1a の図 2 と実験 1b の図 6, 点数 200K の場合の計算時間には FPM, RFPM, と G-RFPM 61.61, 0.96, 0.03 (秒) になっている．これらの実験によって，FPM から RFPM までと RFPM から G-RFPM までの高速化は約 64 倍と 32 倍になる．すべての実験によって，最遅の FPM から最速の G-RFPM ま

末端処理の実装ミスと思われる．

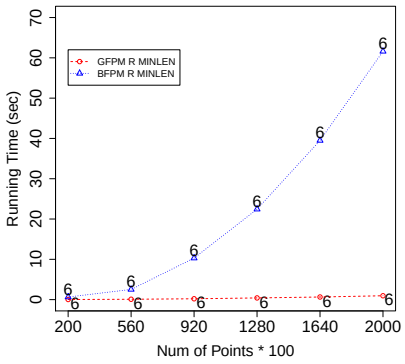

図 6　実験 1b: RFPM (RFPM R) と G-RFPM (GFPM R) の計算時間とパタン数である.

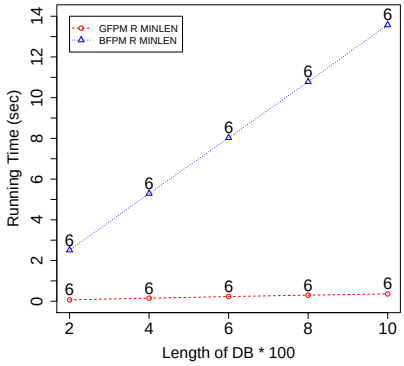

図 7　実験 2b: RFPM (RFPM R) と G-RFPM (GFPM R) の計算時間とパタン数である.

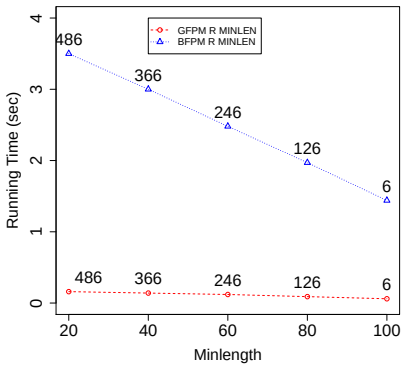

図 8　実験 3b: RFPM (RFPM R) と G-RFPM (GFPM R) の計算時間とパタン数である.

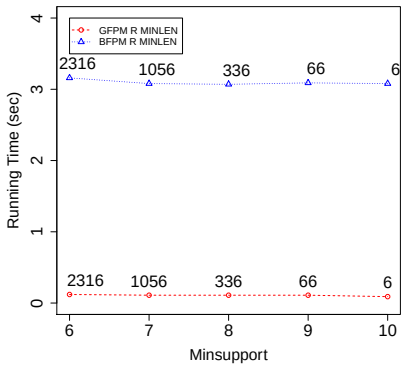

図 9　実験 4b: RFPM (RFPM R) と G-RFPM (GFPM R) の計算時間とパタン数である.

では約 2,000 倍に速くなっている.

5　まとめ

本稿では，軌跡データをマイニング問題を効率よく解決するアルゴリズムを示し，実験の結果により，アルゴリズム RFPM と G-RFPM は FPM より大変加速することを証明した．現実的な応用のために，Hadoop 等の大規模並列環境での群パタンマイニングの高効率実現は今後の課題である.

参考文献

Arimura, H., Geng, X., and Uno, T. (2013). Efficient mining of length-maximal flock patterns from large trajectory data. Manuscript. `http://www-ikn.ist.hokudai.ac.jp/~arim/papers/flockpattern201303.pdf`.

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

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

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

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

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE TKDE*, 16(11):1424–1440.

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

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

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

Zaki, M. J. and Hsiao, C.-J. (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE TKDE*, 17(4):462–478.