密度に基づく意味的な軌跡パターンの発見

姜　仁河† 趙　菁† 董テイテイ† 佐々木勇和†† 石川　佳治†,†††

† 名古屋大学大学院情報科学研究科
†† 名古屋大学未来社会創造機構
††† 国立情報学研究所
E-mail: {jiang,zhao,dongtt,yuya}@db.ss.is.nagoya-u.ac.jp, ishikawa@is.nagoya-u.ac.jp

あらまし　近年，軌跡処理に関連した研究分野において，研究の重点は単純な移動軌跡データから意味的な移動軌跡データへ変化しつつある．特に，意味的な情報と空間的な情報を両方兼ね備える意味的な軌跡パターンの発見という新しい研究が最近出現してきた．系列パターンの発見の技術と軌跡のクラスタリングの技術の組合せが鍵となる．本研究では，先行研究を参考にした上で，密度に基づく意味的な軌跡パターンを提案する．そして，提案した意味的な軌跡パターンが持つ性質について述べる．最後に，発見した性質を利用することで効率よく意味的な軌跡データベースからすべての密度に基づく意味的な軌跡パターンを発見可能な手法を提案する．

キーワード　軌跡, 意味的な軌跡, 系列パターン, クラスタリング, DBSCAN

1. はじめに

GPS デバイス，センサ，ソーシャルネットワークサービスなどの，いろいろな情報源から膨大な軌跡データが発生している．移動軌跡データには，人間活動の情報がたくさん含まれている．そこから検出された人間活動の習慣やパターンを利用して，我々の日常生活への支援を提供できるとされる．そのため，移動軌跡データを対象にした問合せ処理やマイニング処理に関する研究が積極的に行われている．それに関する研究は主に以下の三つに分けられる．

(1) 軌跡のクラスタリング：TRACLUS (Trajectory Clustering) [6] は入力された軌跡データを短い軌跡セグメントに分割して，それらの短い軌跡セグメントに対してクラスタリングを行うという軌跡のクラスタリング手法である．TRACLUS の研究において，動物と台風の移動軌跡データが研究対象とされた．

(2) 軌跡からのパターンの発見: T-Pattern (Trajectory Pattern) [7] は代表的な軌跡パターンの発見に関する研究である．T-Pattern の研究では $RoI_0 \xrightarrow{t_1} RoI_1 \xrightarrow{t_2} ... \xrightarrow{t_n} RoI_n$ のような一連のシーケンスで軌跡パターンを表す．RoI_i(Region of Interest) は発見された多くの軌跡により経由された領域であり，t_i は二つの RoI 間の遷移時間である．T-Pattern の研究において，トラックの移動軌跡データが用いられた．T-Pattern の他，flock [11], swarm [12], convoy [10], gathering [13] の四つの軌跡パターンも提案された．

(3) 軌跡からの最適経路の発見: MPR (Most Popular Route) [8] は起点と終点を入力として，軌跡データをもとに，二点間の一番人気がある経路を発見するという人気経路に関する問合せ処理の研究である．MPR の研究では，T-Pattern の研究と同じく，トラックの移動軌跡データが利用されていた．TPMFP (Time Period-based Most Frequent Path) [9] は起点と終点に加えて，時間帯も入力として，その時間帯の二点間の一番頻繁な経路を発見するという頻繁経路に関する問合せ処理の研究である．TPMFP の研究で，タクシーの移動軌跡データを利用して実験が行われた．

以上の研究で取り扱っていた単純な GPS 移動軌跡データは以下のような時刻 (t) と位置 (l) のペアで構成された一連のシーケンス $\{Oid, (t_1, l_1), (t_2, l_2),..., (t_n, l_n)\}$ で表現可能である．

車などの GPS デバイスから収集してきた単純な軌跡データに対して，Facebook や Twitter，Foursquare などのソーシャルネットワークサービスにおいても，大量の意味的な移動軌跡データが得られる．例えば，Foursquare において，あるユーザはオフィス，レストラン，家で前後して 3 回チェックインしたとき,「オフィス → レストラン → 家」という一つの意味的な移動軌跡が生成されたと考えることができる．意味的な移動軌跡データには，時刻情報と位置情報に加えて，オフィスやレストランや家などのカテゴリ情報も含まれる．従来の単純な移動軌跡データの拡張として，意味的な移動軌跡データは時刻 (t)，位置 (l)，カテゴリ (c) のトリプルで構成された一連のシーケンス $\{Oid, (t_1, l_1, c_1), (t_2, l_2, c_2),..., (t_n, l_n, c_n)\}$ で表現可能である．

そのような意味的な軌跡データから意味的な軌跡パターンを発見する代表的な研究は Splitter [1] である．研究の本質は軌跡のマイニング処理で，軌跡のクラスタリングの研究と軌跡からのパターンの発見の研究の組合せだと考えられる．Splitter により発見されたパターンは意味的な情報と空間的な情報を両方兼ね備えるのが最大の特徴になる．また，Splitter の研究において，Foursquare におけるチェックインデータを実データとして実験が行われた．

本研究は先行研究 Splitter をベースにして，Splitter で考慮されていない問題点に対して，密度に基づく概念 [5] で意味的な軌跡パターンを再定義して，さらにその定義と合わせた効率的なマイニング手法を提案する．最後に評価実験により提案手法の有効性と効率性を示す．先行研究 Splitter の具体的な内容及び問題点は次の章で紹介する．

SID	系列 (Sequence)
1	**A B C** D
2	**A** E **B C**
3	**A B C**
4	**A C** D E

(a) 系列データベース

系列パターン	頻度
A	4
B	3
C	4
A B	3
A C	4
B C	3
A B C	3

(b) 系列パターン

図 1　系列データベースと系列パターン

2. 問題の概要

2.1 既存研究

2.1.1 系列パターンの発見に関する研究

系列パターンの発見（sequential pattern mining）[2] は，系列データベースから，ユーザが指定した閾値を満たす出現順序を維持した部分列を系列パターンとして発見するデータマイニング分野で非常に有名な研究である．

系列データベース (sequence database) は複数の ID 付きの系列データ (sequence data) により構成され，系列データはアイテム (item) により構成される．系列データベースの例を図 1(a) に示す．全部合わせて四つの系列データ，五つのアイテム (A B C D E) がある．ユーザにより指定された閾値が 3 の場合，図 1(a) の系列データベースから，図 1(b) が示すように，頻度が 3 以上の 7 個の系列パターンを発見可能である．その閾値は最小支持度 (minimum support) という．部分列がデータベースで出現した最低頻度を制限する．すなわち，部分列の頻度が最小支持度より小さくなければ，系列は頻出系列になり，すなわち，系列パターンになる．

系列パターンの発見の研究 [2] で提案された AprioriAll 手法より，Pei らは，効率的な PrefixSpan [3] 手法を提案した．PrefixSpan は系列パターンの発見問題への有効な手法として，よく使われる．

また，系列パターンを効率よく発見することを可能にした重要なポイントは以下の Apriori 性質である．

- Apriori 性質：頻出する系列の部分系列も頻出する．

例えば，表 1(b) に示すように，長さ 3 の頻出系列 A B C の部分系列はすべて頻出である．Apriori 性質に基づいて，長さ k の系列パターンを検出するとき，長さ $k-1$ の系列パターンが活用可能になる．本研究も意味的な軌跡パターンの発見へ Apriori 性質の応用を基礎にする．

2.1.2 意味的な軌跡パターンの発見に関する研究

この節で本研究の基礎となる意味的な軌跡パターンの発見の研究 Splitter [1] を紹介する．

先に紹介した通り，Splitter の研究で取り扱っていた軌跡データは従来の単純の軌跡データに加えてカテゴリ情報も付いた意味的な軌跡データになる．具体的に，図 2 に示すような場所 p_i のシーケンスで表示された軌跡のデータベースがあるとする．各場所 p_i は図 3 に示すように，オフィス，ジム，レストランのいずれのカテゴリに属する．

オブジェクト	意味的な軌跡
o_1	$< p_3, \boldsymbol{p_1}, \boldsymbol{p_7}, \boldsymbol{p_9} >$
o_2	$< p_5, p_7, \boldsymbol{p_2}, \boldsymbol{p_7}, \boldsymbol{p_{10}} >$
o_3	$< p_3, p_6 >$
o_4	$< p_2, \boldsymbol{p_1}, p_6, \boldsymbol{p_8}, \boldsymbol{p_{11}} >$
o_5	$< p_{12}, p_8, p_{11}, p_4 >$

図 2　軌跡データベース

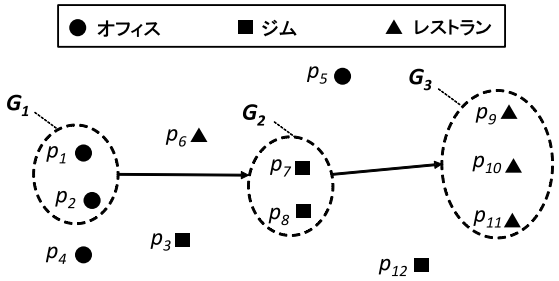

図 3　意味的な軌跡パターンの発見

このような軌跡データベースを系列データベースとして処理して，各場所 p_i をアイテムとして取り扱う．すると，最小支持度が 3 の場合，場所 p_i で構成された系列パターンは一つも発見できない．しかし，距離が近く同じカテゴリに属する場所を一つのグループにして，このようなグループをアイテムとして取り扱うと，最小支持度が 3 の場合，図 3 に示すように，系列パターン $G_1 \rightarrow G_2 \rightarrow G_3$ を発見可能になる．

以上のような空間的な情報（位置）と意味的な情報（カテゴリ）の両方を兼ね備える系列パターンを発見するために，Splitter 手法は二つのステップでマイニング処理を行う．

- まず，意味的な軌跡データベースをカテゴリで構成された系列データベースへ変換して，系列パターンの発見手法 PrefixSpan [3] により粗粒度パターン（Coarse Pattern）を発見する．同時に，Snippet と呼ばれる粗粒度パターンに対応する部分軌跡の集合も求められる．例えば，図 3 に示す「オフィス → ジム → レストラン」の粗粒度パターンに対応する Snippet は $< p_1, p_7, p_9 >$（o_1 の部分軌跡），$< p_2, p_7, p_{10} >$（o_2 の部分軌跡），o_4 の $< p_1, p_8, p_{11} >$（o_4 の部分軌跡）の三つがある．

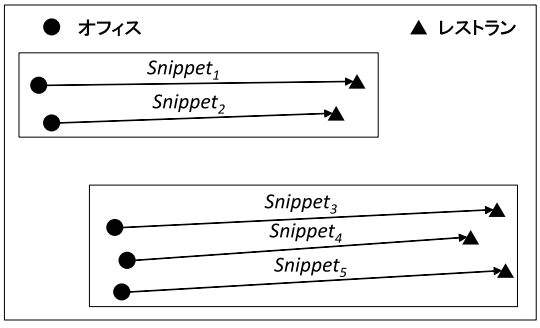

図 4　細粒度系列パターンの例

- 次に，粗粒度パターンに対応する部分軌跡をクラスタリングすることで，距離が近い軌跡により構成されたクラスタを得

る．そのようなクラスタを最終の結果となる細粒度系列パターン（Fine-Grained Sequential Pattern）と呼ぶ．粗粒度パターンに対応する部分軌跡を Snippet と呼ぶ．図 4 を例に説明すると，粗粒度パターン「オフィス → レストラン」が発見され，それに対応する Snippet が五つがあり，それから二つの細粒度パターン $\{Snippet_1, Snippet_2\}$ と $\{Snippet_3, Snippet_4, Snippet_5\}$ が発見可能になる．

2.2 既存研究の問題点と本研究のポイント

本研究の基礎である [1] の問題点には主に以下の二つがある．

- (1) Snippet の距離関数が定義されていない．細粒度系列パターンの本質は，距離が近い Snippet により構成されたクラスタである．Splitter は空間的な偏差（spatial variance），すなわち，座標値の偏差に基づいた細粒度系列パターンの定義を提案し，距離の制約を実現した．具体的には，細粒度パターン $G_1 \to G_2 \to \ldots \to G_k$ に対し，$\frac{1}{k}\sum_{i=1}^{k} Var(G_i) \leqq \rho$ という制約条件がある．$Var(G_i)$ は G_i にある場所 p_i の座標値の偏差で，ρ は閾値である．Splitter 手法は二つ目のステップで Snippet を処理対象としてクラスタリングを行うが，直観的で計算しやすい Snippet の距離関数が提案されていない．そのため，Splitter 手法のクラスタリングのステップでの処理はとても複雑になる．
- (2) すべての粗粒度パターンから細粒度系列パターンを効率よく発見する手法は提案されていない．Chao らは [1] で一つだけの粗粒度パターンに対して細粒度系列パターンを発見することに着目して，有効性と効率性がある Splitter 手法を提案したが，一つ目の処理ステップで PrefixSpan 手法により発見されたすべての粗粒度パターンに対して，どのように効率よく細粒度系列パターンを発見するべきかについて議論しなかった．

以上の問題点に対して，本研究のポイントは以下の三つがある．(a) 直観的で実用性が高い Snippet 間の距離関数を提案して，DBSCAN [5] で効率よく Snippet をクラスタリングすることを実現する．(b) Snippet 間の距離関数の性質に基づいて，空間索引構造 M-tree [4] でクラスタリングを効率化する．(c) すべてのパターンの発見に対する Apriori 性質をベースにした効率よいマイニング手法を提案する．

3. 問題の定義

本稿で説明に用いる記号を表 1 に示す．

3.1 基本概念

- 意味的な軌跡：意味的な軌跡は時刻，位置，カテゴリのトリプルで構成されたシーケンスである．簡単のため、これから本稿で意味的な軌跡を軌跡と呼ぶ．$T = Oid, (t_1, l_1, c_1), \ldots, (t_n, l_n, c_n)$ で表す．
- 意味的なパターン：[1] における粗粒度パターン，すなわち，カテゴリ（c_i）により構成された系列パターンを本稿で意味的なパターンと呼ぶ．$C_1 \to C_2 \to \ldots \to C_n$ で表す．
- Snippet：[1] における Snippet の概念をそのまま使う．Snippet は意味的なパターンに対応する部分軌跡であり，軌跡と同様に表現できる．これから本稿では軌跡の代わりに，Snippet をキーワードにして述べる．

表 1　本稿で用いる記号

記号	意味
T	軌跡データ
Oid	移動オブジェクトの ID
(t_i, l_i)	i 番目の時刻と位置のペア
(t_i, l_i, c_i)	i 番目の時刻，位置，カテゴリのトリプル
C_i	意味的なパターンの i 番目のアイテム
S	Snippet
SS	Snippet の集合
p	Point または場所
$S.p_i$	Snippet のポジション i での Point
$MinSup$	パターン発見のための最小支持度
η	DBSCAN 手法における距離の閾値
$MinObjs$	DBSCAN 手法における密度の閾値

- Snippet 集合：本研究でデフォルトとして，Snippet 集合はすべて同じ長さの Snippet により構成される．SS で表す．
- Point 及びポジション： Snippet の i 番目の位置 l_i を Snippet のポジション (Position)i での Point と呼ぶ．$S.p_i$ で表す．図 3 にある各場所 p_i と同じ意味となるので，本稿で区別しない．そして，これから本稿において p のシーケンス $\{p_1, p_2, ..., p_n\}$ で Snippet を表す．
- 軌跡パターン：[1] における細粒度系列パターン，すなわち，距離が近い Snippet により構成された Snippet のクラスタを本稿で軌跡パターンと呼ぶ．
- 極大の軌跡パターン： 二つの長さ k の Snippet クラスタ A と B があり，もし $A \cap B \neq \emptyset$ であれば，A と B は重なるという．そして，他の Snippet クラスタと重なりがなく，サイズが極大の Snippet クラスタを極大の軌跡パターンと呼ぶ．本研究の目標はこのような極大の軌跡パターンを発見することなので，これから本稿では極大の軌跡パターンを軌跡パターンとして述べる．

3.2 Snippet の距離関数

意味的なパターンから意味的な軌跡パターンを発見するとき，処理対象である Snippet はすべて意味的なパターンと同じ長さを持つ．この前提条件のもとで，Snippet の距離関数を以下のように定義する．

$$d(S_1, S_2) = \max_{i=1}^{n} |EuclideanDistance(S_1.p_i, S_2.p_i)|$$

二つの Snippet 間の距離は各対応する点の間のユークリッド距離の最大値になる．

3.3 Snippet の密度に関する概念

DBSCAN [5] は任意形状のクラスタの抽出を目的とした密度（density）に基づくクラスタリング手法である．要求されるパラメータは二つがあり，一つ目は距離閾値 η で，二つ目は密度閾値 $MinObjs$ である．DBSCAN [5] で Neighbor Object, Core Object, Direct Density-Reachable, Density-Reachable, Density-Connected という五つの概念が提案された．改めて Snippet に拡張したこの五つの概念を説明する．

- Neighbor Snippet（近傍 Snippet）：距離が η の Snippet は近傍 Snippet になる．図 5 において，S_2 の近傍 Snippet は

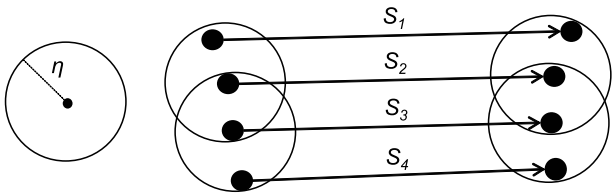

図 5　Snippet の Density-Connected の例，$MinObjs = 2$

S_1, S_3 となる．

- Core Snippet：近傍 Snippet の数が $MinObjs$ 以上の Snippet は Core Snippet となる．図 5 に S_2, S_3 は Core Snippet となる．
- Snippet の Directly Density-Reachable：Snippet P,Q があり，Q は P の近傍 Snippet で，P は Core Snippet であるなら，Q は P から Directly Density-Reachable となる．図 5 において，S_1 は S_2 から Directly Density-Reachable である．
- Snippet の Density-Reachable：一連の Snippet $S_1S_2...S_i...S_n$ があり，$P = S_1, Q = S_n$，S_{i+1} は S_i から Directly Density-Reachable であるならば，Q は P から Density-Reachable となる．図 5 において，S_1 は S_3 から Density-Reachable である．
- Snippet の Density-Connected：Snippet P，Q，O があり，P は O から Density-Reachable であり，Q も O から Density-Reachable であれば，P と Q は Density-Connected となる．図 5 において，S_1 と S_4 は Density-Connected である．

3.4　密度に基づく軌跡パターン

同じ長さの Snippet により構成された Snippet 集合 SS があり，以下の二つの条件を満たせば，SS を密度に基づく軌跡パターンと呼ぶ．

(1) 集合 SS のサイズ $\geqq MinSup$.

(2) 集合 SS 中の Snippet は η と $MinObjs$ 条件のもとで Density-Connected である．

3.5　密度に基づく意味的な軌跡パターン

意味的なパターン $C_1 \rightarrow C_2 \rightarrow ... \rightarrow C_k$ があり，それに対応する長さ k の Snippet 集合 SS があるとする．SS が密度に基づく軌跡パターンになれば，密度に基づく意味的な軌跡パターンにもなる．

3.6　密度に基づく意味的な軌跡パターンの発見

本研究の最終の問題定義である密度に基づく意味的な軌跡パターンの発見の定義は以下になる．

与えられた $MinSup, \eta, MinObjs$ の条件のもとで，意味的な軌跡データベースからすべての密度に基づく意味的な軌跡パターンを発見する．

4.　提 案 手 法

4.1　密度に基づく軌跡パターンの発見

4.1.1　SDBSCAN

密度に基づく軌跡パターンの発見は実際には Snippet のクラスタリングとなる．前章で提案した Snippet の密度に基づく概念に基づいて，DBSCAN で Snippet をクラスタリングすることができる．具体的には，Snippet を元々の DBSCAN の処理対象である Point として取り扱い，前章で提案した Snippet 距離関数により Snippet 間の距離を計算して，これ以外は元々の DBSCAN と同じようなプロセスで処理する．元々の DBSCAN と区別するために，Snippet に拡張した DBSCAN を SDBSCAN と呼ぶ．

4.1.2　SDBSCAN の効率化

元々の DBSCAN 手法においては，空間索引構造 R-tree の利用で，時間がかかる範囲問合せを効率よく処理可能になる．しかし，SDBSCAN において，ある Snippet の η 以内の近傍 Snippet 集合を求めるという Snippet の範囲問合せ処理が必要である．前章で提案した Snippet の距離関数はユークリッド距離でないので，直接 R-tree を利用できない．したがって，ボトルネックである Snippet の範囲問合せ処理に対して，R-tree の代わりに，もう一つのよく使われる空間索引構造である M-tree を利用した改善手法を提案する．

M-tree は距離に基づく検索に対する有効な空間索引である．以下の四つの性質を満たすと，関数 $d(O_i, O_j)$ は距離関数である．

- 同一律 (reflexivity): $d(O_i, O_i) = 0$
- 非負性 (non-negativity): $d(O_i, O_j) \geqq 0$
- 対称性 (symmetry): $d(O_i, O_j) = d(O_j, O_i)$
- 三角不等式 (triangular inequality): $d(O_i, O_k) + d(O_k, O_j) \geqq d(O_i, O_j)$

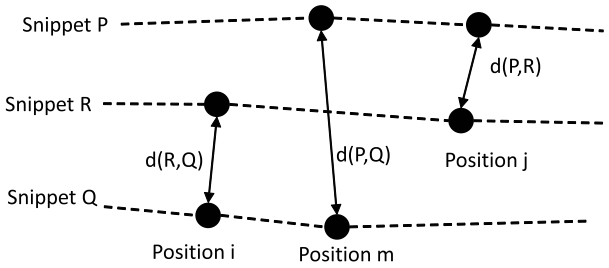

図 6　Snippet 間の距離が三角不等式を満たすことの例

定理 1. *Snippet* 間の距離は距離の公理を満たす.

証明.　P，R，Q の三つの Snippet があり，i，j，m 三つのポジションがあり，P，Q 間の距離 $d(P,Q) = d(P.p_m, Q.p_m)$，R，Q 間の距離 $d(R,Q) = d(R.p_i, Q.p_i)$，P，R 間の距離 $d(P,R) = d(P.p_j, R.p_j)$ となるとする．Snippet 間の距離関数により，$d(R.p_i, Q.p_i) \geqq d(R.p_m, Q.p_m)$，$d(P.p_j, R.p_j) \geqq (P.p_m, R.p_m)$ がある．

$\because\ d(R.p_i, Q.p_i) \geqq d(R.p_m, Q.p_m),\quad d(P.p_j, R.p_j) \geqq (P.p_m, R.p_m)$

$\therefore\ d(R.p_i, Q.p_i) + d(P.p_j, R.p_j) \geqq d(R.p_m, Q.p_m) + d(P.p_m, R.p_m)$

$\because d(R.p_m, Q.p_m) + d(P.p_m, R.p_m) \geqq d(P.p_m, Q.p_m)$

$\therefore d(R,Q) + d(P,R) \geqq d(P,Q)$

以上で三角不等式の性質を満たすことを証明した．また同一律，非負性，対称性を満たすことは簡単に証明可能で，本稿で

は省略する．□

定理 **1** により，入力された Snippet 集合に対して M-tree を作って，M-tree で Snippet の範囲問合せを処理することが可能になる．これで SDBSCAN を効率化する．

4.2 密度に基づくすべての意味的な軌跡パターンの発見

4.2.1 単純な発見手法

単純な発見手法の処理の流れをアルゴリズム 1 にまとめる．前述の通り，二つのステップでマイニング処理を行う．(1) すべての意味的なパターンの発見: 軌跡データベースをカテゴリにより構成された系列パターンのデータベースに変換する．PrefixSpan 手法ですべての意味的なパターンと対応する Snippet 集合を得る．データ構造でパターンと Snippet 集合をペアとして格納する．(2) 意味的なパターンに対する密度に基づくすべての軌跡パターンの発見: Snippet 集合を一つずつ SDBSCAN でクラスタリングする．η と $MinObjs$ の条件のもとで Density-Connected な Snippet のクラスタ集合を候補集合に追加する．最後に候補集合からサイズが $MinSup$ 以上の Snippet クラスタを最後の結果集合に追加する．

4.2.2 成長型の発見手法

単純な発見手法は各 Snippet 集合を独立で一つずつ処理することで，すべての密度に基づく意味的な軌跡パターンを発見する．M-tree を利用した SDBSCAN でクラスタリング処理を効率化できるが，すべてのパターンを発見するには，まだ不十分だと考えられる．長い軌跡パターンを発見する時，処理済の短い軌跡パターンを活用できれば効率をさらに改善可能であると考える．以下で，これに関連して導出した性質を列挙する．

前提条件．*Snippet* 集合 *SS* があり，*SS* にある *N* 個の *Snippet* が同じポジションで同じ *Point* を持つ場合，*N* 個のユークリッド距離がゼロの *Point* と見なす．以下の補助定理や性質はすべてこの前提条件に基づく．

Algorithm 1 NAIVE MINING

```
1: 入力：TrajectorySet, MinSup, η, MinObjs
2:         ▷ 軌跡集合，最小支持度の閾値，距離の閾値，密度の閾値
3: 出力： Snippet Cluster Set
4:         ▷ 条件を満たした密度に基づく意味的な軌跡パターンの集合
5: アルゴリズム：
6: R ← ∅;                                  ▷ Result Snippet Cluster Set
7: C ← ∅;                                  ▷ Candiadate Snippet Cluster Set
8: SequenceSet ← Transform(TrajectorySet);
9: PatternSet ← PrefixSpan(SequenceSet, MinSup);
10: foreach pattern ∈ PatternSet do
11:     SnippetSet ← GetSnippetSet(pattern);
12:     C ← SDBSCAN(SnippetSet, η, MinObjs);
13:     foreach cluster ∈ C do
14:         if cluster.size ≧ MinSup then
15:             R ← R ∪ cluster;
16:         end if
17:     end for
18: end for
19: output R;
```

補助定理 1. *Snippet* 集合 *SS* があり，*SS* にある *Snippet S* が与えられた η と $MinObjs$ について *Core Snippet* であるとき，*SS* のポジション *i* での *Point* 集合において，*S* のポジション *i* での *Point* $S.p_i$ は同じ条件のもとで *Core Point* である．

補助定理 2. *Snippet* 集合 *SS* があり，*SS* にある *Snippet X* と *Y* が与えられた η と $MinObjs$ について *Directly Density-Reachable* であるとき，*SS* のポジション *i* での *Point* 集合において，$X.p_i$ と $Y.p_i$ は同じ条件のもとで *Directly Density-Reachable* である．

補助定理 3. *Snippet* 集合 *SS* があり，*SS* にある *Snippet X* と *Y* が与えられた η と $MinObjs$ について *Density-Reachable* であるとき，*SS* のポジション *i* での *Point* 集合において，$X.p_i$ と $Y.p_i$ は同じ条件のもとで *Density-Reachable* である．

定理 2. *Snippet* が *Density-Connected* であるならば，対応する *Point* は *Density-Connected* である． すなわち，*Snippet* 集合 *SS* が与えられた η と $MinObjs$ について *Density-Connected* な *Snippet* のクラスタであれば，*SS* のポジション *i* での *Point* 集合は同じ条件のもとで *Density-Connected* な *Point* のクラスタである．

証明． 先に示した補助定理により定理 **2** が成立する．□

性質 1. *Snippet* が *Density-Connected* であるならば，対応する部分 *Snippet* は *Density-Connected* である． すなわち，*Snippet* 集合は与えられた η と $MinObjs$ について *Density-Connected* な *Snippet* のクラスタであれば，同じポジションでの部分 *Snippet* により構成された *Snippet* 集合は同じ条件のもとで *Density-Connected* な *Snippet* のクラスタである．

例を用いて説明する．図 2 にある Snippet $S_1\{p_1, p_7, p_9\}$, $S_2\{p_2, p_7, p_{10}\}$, $S_3\{p_1, p_8, p_{11}\}$ は同じポジション 1, 3 での部分 Snippet をそれぞれ $S'_1\{p_1, p_9\}$, $S'_2\{p_2, p_{10}\}$, $S'_3\{p_1, p_{11}\}$ となる．S_1, S_2, S_3 が Density-Connected であるならば，S'_1, S'_2, S'_3 も Density-Connected である．

性質 **1** は本質的に Apriori 性質と同じである．すなわち，Snippet の Density-Connected は Apriori 性質を満たす．

性質 2. 長さ *k* の *Snippet* 集合 *SS* があり，その中の *Snippet* の長さ $k-1$ の接頭辞（注 *1*）により構成された長さ $k-1$ の *Snippet* 集合 SS' があるとする．*SS* が与えられた η と $MinObjs$ について *Density-Connected* な *Snippet* クラスタであれば，SS' は同じ条件のもとで *Density-Connected* な *Snippet* クラスタである．

注 1. 長さ k の $Snippet\{p_1, p_2, ..., p_k\}$ の長さ $k-1$ の接頭辞は $Snippet\{p_1, p_2, ..., p_{k-1}\}$ となる．たとえば，長さ *4* の $Snippet\{p_1, p_2, p_3, p_4\}$ の長さ *3* の接頭辞は $Snippet\{p_1, p_2, p_3\}$ となる．簡単のため，長さ k の $Snippet\{p_1, p_2, ..., p_k\}$ の長さ $k-1$ の接頭辞を直接 *Snippet* の接頭辞と呼ぶ．

ここからは性質 **2** に基づいた成長型の発見手法を提案する．

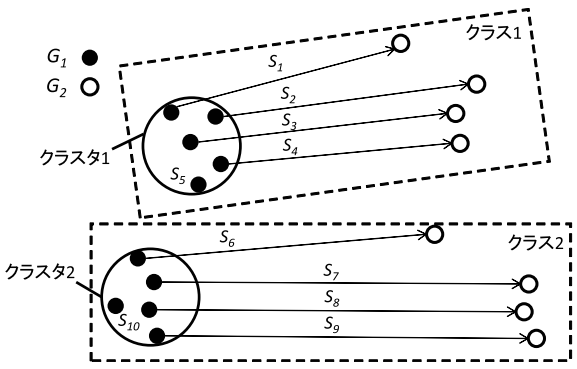

図 7　成長型の発見手法

処理の流れをアルゴリズム 2 にまとめる．手法の概要は以下の四点になる．

(1) 辞書順でソートされたパターンを一つずつ処理

(2) 接頭辞の処理結果により Snippet 集合をクラスに分割

(3) 条件に満足したクラスを一つずつ SDBSCAN で処理

(4) 条件に満足した Snippet のクラスタを結果集合に追加

図 7 を例にして概要を説明すると，パターン $C_1 \rightarrow C_2$ は辞書順でパターン C_1 の後で処理される．$C_1 \rightarrow C_2$ に対応する Snippet 集合 $SS = \{S_1, S_2, S_3, S_4, S_6, S_7, S_8, S_9\}$ がある．SS をクラスタリングする前に，$C_1 \rightarrow C_2$ の接頭辞である C_1 の処理結果，すなわち，クラスタ 1 とクラスタ 2 により SS を二つのクラスにわける．それぞれをクラス 1$\{S_1, S_2, S_3, S_4\}$ とクラス 2$\{S_6, S_7, S_8, S_9\}$ と呼ぶ．この時点でクラスのサイズが $MinSup$ を満たすかを判断し，クラスタとするかを決める．例えば，$MinSup = 5$ の場合，二つのクラスともクラスタとはしない．$MinSup = 3$ の場合，二つのクラスを別々に SDBSCAN でクラスタリングする．

これから成長型の発見手法を具体的に説明する．軌跡データベースの変換と PrefixSpan 手法により意味的なパターンを発見することは前の手法と一致する．本手法のポイントは，長さ k の Snippet 集合を処理するとき，長さ $k-1$ の Snippet 集合の処理結果を参照することである．それを実現するために，まず，パターンと Snippet 集合のペアを格納するデータ構造に加えて，パターンと処理結果のペアを格納するデータ構造も必要になる．また，パターンを辞書順でソートする必要もある．例えば，辞書順でパターンをソートすると，パターン C_1C_2 はパターン $C_1G_2G_3$ の前になる．ソートされたパターンを一つずつ処理する．ここまでは 6 行目から 11 行目までの処理になる．次に，長さ k のパターンの長さ $k-1$ の接頭辞 (prefix) を求める．15 行目でパターンの接頭辞があるかどうかを判断し，パターンの長さが 1 の場合，すなわち，接頭辞がない場合，単純に SDBSCAN 手法でクラスタリングする．パターンの接頭辞があるが，発見された接頭辞と対応する Snippet クラスタ，すなわち，接頭辞の処理結果が存在しない場合，Apriori 性質により，本パターンから Snippet クラスタを一つも発見できないと決められるので，直接候補集合を空にする．これは 19 行目の処理となる．一方，パターンの接頭辞があり，かつ，接頭辞の処理結果が存在する場合，Apriori 性質により，接頭辞の処理結果で Snippet 集合をいくつかの Snippet クラスに分割して，各 Snippet クラスを別々に処理することが可能になる．これは 22 行目までの処理となる．サイズが $MinSup$ より小さい Snippet クラスを直接枝刈りできる．サイズが $MinSup$ より小さくない Snippet クラスを一つずつ SDBSCAN でクラスタリングする．η と $MinObjs$ の条件で Density-Connected な Snippet のクラスタ集合を候補集合に追加する．以上は 23 行目から 27 行目までの処理となる．最後に候補集合からサイズが $MinSup$ より小さくない Snippet クラスタをパターンの結果集合に追加する．これは 30 行目以降の処理となる．

Algorithm 2 Growth-Type Mining

```
    入力：TrajectorySet, MinSup, η, MinObjs
 2:          ▷ 軌跡集合，最小支持度の閾値，距離の閾値，密度の閾値
    出力：Snippet クラスタの集合
 4:       ▷ 条件を満たした密度に基づく意味的な軌跡パターンの集合
    アルゴリズム：
 6: SequenceSet ← Transform(TrajectorySet);
    PatternSet ← PrefixSpan(SequenceSet, MinSup);
 8: SortedPatternSet ← Sort(PatternSet);
    foreach pattern ∈ SortedPatternSet do
10:     Processing(pattern);
    end for
12: function Processing(pattern)
        SnippetSet ← GetSnippetSet(pattern);
14:     C ← ∅                          ▷ Candiadate Snippet Cluster Set
        if pattern.prefix = null then
16:         C ← SDBSCAN(SnippetSet, η, MinObjs);
        else
18:         if pattern.prefix.result=∅ then
                C ← ∅;
20:         else
                pResult ← pattern.prefix.result;
22:             SnippetClassSet ← Classify(pResult, SnippetSet);
                foreach class ∈ SnippetClassSet do
24:                 if class.size ≧ MinSup then
                        C ← C ∪ SDBSCAN(class, η, MinObjs);
26:                 end if
                end for
28:         end if
        end if
30:     foreach cluster ∈ C do
            if cluster.size ≧ MinSup then
32:             pattern.result ← pattern.result ∪ cluster;
            end if
34:     end for
        output pattern.result;
36: end function
```

本手法における Apriori 性質の応用には二箇所がある．いずれも前で述べた性質 **2** の直接応用になる．(1) 接頭辞の Snippet 集合から Snippet クラスタを発見できない場合，パターンの Snippet 集合から Snippet クラスタも発見できない．(2) 接頭

辞の処理結果によりパターンに対応する Snippet 集合を分割して別々に処理する．

5. 評価実験

5.1 実験概要

5.1.1 実験データ

本研究では，人工データとして，Brinkhoff 生成器 [14] を用いてオブジェクトの軌跡データを生成した．Brinkhoff の生成器は，道路ネットワーク上のオブジェクトの軌跡データを生成する．道路ネットワークとして，サンフランシスコの道路ネットワークを用いた．軌跡データは座標が空間範囲 $(0,0)*(30000,30000)$ 内の点により構成される．長さ 3 の軌跡データセットと，長さ 6 の軌跡データセットを得るための処理プロセスは以下になる．

(1) 特定の長さの軌跡の作成．意味的な軌跡データの特徴として，サンプリングレートが低いことが挙げられる．そのため，生成した軌跡データをさらにサンプリングすることで，軌跡データを作成した．具体的には，生成した軌跡を 1/10 のレートでサンプリングした．本実験では，生成器のタイムスタンプ数を 30 および 60 として生成した軌跡データを，1/10 のレートでサンプリングすることにより，長さ 3 および 6 の軌跡データをそれぞれ作成した．

(2) 意味的な軌跡データの作成．ここで，本手法の有効性を検証するためには，長さ 3 の軌跡パターンの発見ができれば十分であることを例を用いて示す．$C_1 \to C_2 \to C_3$ に対応する意味的な軌跡パターンを発見できれば，$C_1 \to C_2$ に対応する意味的な軌跡パターンも発見できる．これは，すべてのパターンを発見可能であることを示し，手法の有効性を十分に検証できる．(1) の処理により作成した長さ 3 と 6 の軌跡データのそれぞれのポイントに，カテゴリ情報を付与する．具体的には，長さ 3 の軌跡データの三つのポイントそれぞれに A，B，C をカテゴリ情報として付与し，長さ 6 の軌跡データの六つのポイントそれぞれに A，B，C，D，E，F を付与する．これにより，意味的な軌跡データを生成した．

(3) 人工データセットの作成．まず，サイズが 3000 の軌跡データセットを生成器により 20 個作成して，それぞれのデータセットに対して，クラスタリング手法により，軌跡パターンとなる軌跡データを抽出する．20 個のデータセットから軌跡パターンとなる 5000 個の軌跡データを抽出する．その後，無作為の軌跡を 5000 個作成して，抽出した 5000 個の軌跡と合わせて，サイズが 10000 のデータセットを作成した．

5.1.2 実験環境

実験で用いた計算機のスペックを表 2 に示す．実装は Java で行い，コンパイルには Eclipse を使用した．

表 2　実行マシンのスペック

OS	Windows7 Professional 64bit
CPU	Intel(R) Core(TM) i7-4770 CPU @ 3.4GHz
CPU コア数	8
メモリ	8GB

5.2 実験内容

本章で用いるそれぞれの手法の略称を表 3 に示す．MSDBSCAN は，M-tree を用いて効率化した SDBSCAN を表す．実験内容として，提案手法の正確性の検証実験，効率性の検証実験，およびパラメータに関する実験の三つである．

表 3　評価実験に用いる各提案手法の略称

提案手法の略称	提案手法
NaiveS	SDBSCAN を用いた単純な発見手法
NaiveM	MSDBSCAN を用いた単純な発見手法
GrowthS	SDBSCAN を用いた成長型の発見手法
GrowthM	MSDBSCAN を用いた成長型の発見手法

5.2.1 正確性の検証実験

以上の二つのデータセットに対して，NaiveM，GrowthS，GrowthM はすべて NaiveS と同じ処理結果を得た．さらに，他のデータセットに対して実行した場合でも，同じ結果が得られたため，提案手法の正確性が検証された．

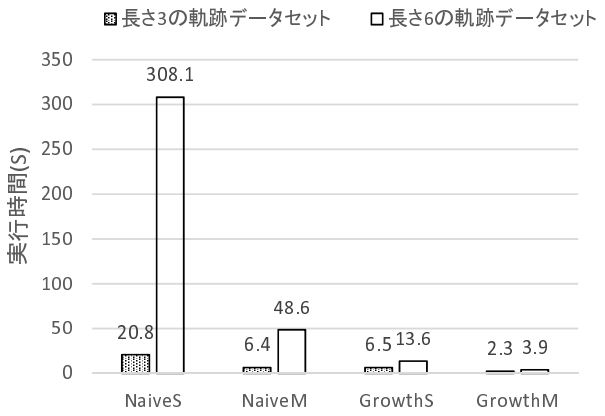

図 8　各手法の実行時間

5.2.2 効率性の検証実験

本実験で各手法の実行時間を比較する．各パラメータの値は $MinSup = 50, \eta = 2.0, MinObjs = 1$ である．

図 8 に示す通り，軌跡データの長さが 3 の場合，NaiveM は NaiveS の約 1/3 程度の実行時間になり，GrowthM も GrowthS の約 1/3 程度の実行時間になる．軌跡データの長さが 6 の場合，NaiveM は NaiveS の約 1/6 程度の実行時間になり，GrowthM は GrowthS の約 1/3 程度の実行時間になる．この結果により，MSDBSCAN の効率性が検証された．また，軌跡データの長さが 3 の場合，GrowthS は NaiveS の約 1/3 程度の実行時間になり，NaiveM とほぼ同じ実行時間になるが，軌跡データの長さが 6 になると，GrowthS は NaiveS の約 1/25 程度の実行時間になり，NaiveM の約 1/4 程度の実行時間になる．この結果により，成長型の発見手法の効率性が検証された．

5.2.3 パラメータに関する実験

最小支持度 $MinSup$ と距離閾値 η を変化させて実験を行った．$\eta = 2.0, MinObjs = 1$ の条件のもとで，最小支持度 $MinSup$ による発見された軌跡パターンの数の変化を図 9 に示す．系列パターンと同じように，最小支持度が大きく

なると，発見された軌跡パターンの数が少なくなる．また，$MinSup = 50, MinObjs = 1$ の条件のもとで，距離閾値 η による発見された軌跡パターンの数の変化を図 10 に示す．距離閾値が大きくなると，発見された軌跡パターンの数はほぼ変わらないことがわかる．原因は以下のように考えられる．Brinkhoff 生成器により生成された軌跡データは道路ネットワークと合わせる軌跡データであるので，生成された軌跡は道路セグメントの近くに集積する．そのため，軌跡データの密度が高く，発見された軌跡パターンの数は距離閾値によりあまり影響されない．

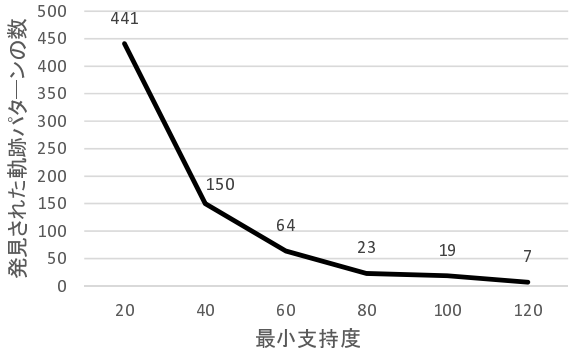

図 9　最小支持度 $MinSup$ により発見された軌跡パターンの数

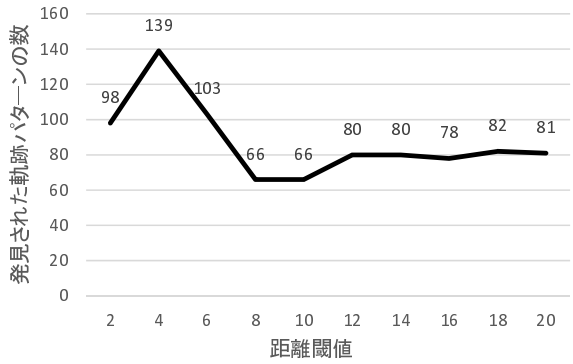

図 10　距離閾値 η により発見された軌跡パターンの数

6. まとめと今後の課題

本研究では，[1] で提案された意味的な軌跡パターンの発見の問題について述べ，改良手法を提案した．具体的には，軌跡の距離関数を提案して，それを用いて密度に基づく意味的な軌跡パターンを定義した．また，距離に基づく空間索引構造である M-tree で軌跡に拡張した DBSCAN 手法を効率化した．さらに，密度に基づく意味的な軌跡パターンが Apriori 性質を持つことを示した．最後に，この Apriori 性質に基いて，意味的な軌跡データベースから密度に基づくすべての意味的な軌跡パターンを効率よく発見する手法を提案した．実験を行うことで，本手法は軌跡データが長ければ長いほど有効性があると示した．

今後の課題として，まず，実際に SNS にある意味的な軌跡データ，すなわち，チェックインデータを収集して，実データ上で実験を行い，有効性を検証する．また，今の段階でクラスタリングと系列パターンの発見は二つのステップで行われるが，両方とも Apriori 性質を満たすので，クラスタリング部分の処理を系列パターンの発見の処理に組み込むことが考えられる．それにより，より効率を上げられると予想している．最後に，本研究で意味的な軌跡データに対して，カテゴリ情報と空間情報を利用するが，時間情報を取り扱っていない．軌跡パターンの特性である空間的な近さに対して，時間的に近いという制限の追加も考えられる．そうすると，カテゴリ情報，空間情報，時間情報の三つともを含めることができる．これにより，さらに実用性がある意味的な軌跡パターンをユーザに提供することが可能になる．

謝　　辞

本研究の経費の一部は科学研究費（25280039）および地球環境情報統融合プログラム（DIAS-P）による．

文　　献

[1] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas F. La Porta. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *PVLDB*, Vol. 7, No. 9, pp. 769–780, 2014.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, pp. 3–14, 1995.

[3] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pp. 0215–0215, 2001.

[4] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pp. 426–435, 1997.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, Vol. 96, pp. 226–231, 1996.

[6] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, pp. 593–604, 2007.

[7] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *KDD*, pp. 330–339, 2007.

[8] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *ICDE*, pp. 900–911, 2011.

[9] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M Ni. Finding time period-based most frequent path in big trajectory data. In *SIGMOD*, pp. 713–724, 2013.

[10] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *PVLDB*, Vol. 1, No. 1, pp. 1068–1080, 2008.

[11] Patrick Laube and Stephan Imfeld. Analyzing relative motion within groups oftrackable moving point objects. In *Geographic information science*, pp. 132–144. 2002.

[12] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *PVLDB*, Vol. 3, No. 1-2, pp. 723–734, 2010.

[13] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. On discovery of gathering patterns from trajectories. In *ICDE*, pp. 242–253, 2013.

[14] Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, Vol. 6, pp. 153–180, 2002.