マルコフ連鎖モデルに基づく動的な移動ヒストグラム構築手法

町田　陽二†　　石川　佳治††,†††　　北川　博之††,†††

† 筑波大学大学院理工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学大学院システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1–1–1
††† 筑波大学計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1–1–1
E-mail: †y-machida@kde.cs.tsukuba.ac.jp, ††{ishikawa,kitagawa}@cs.tsukuba.ac.jp

あらまし　GPS や通信技術の発展に伴い，リアルタイムに移動する多数のオブジェクトの移動状況の追跡が容易になっている．こうした蓄積された大量な移動状況データを分析するには，効率よく移動統計量を求める必要がある．そこで，我々は移動データを要約する，移動ヒストグラムの構築手法の開発を進めている．このヒストグラムは，マルコフ連鎖モデルに基づいている．ユーザごとに分析・予測に役立てるために，我々は移動ヒストグラムの論理モデルとしてデータキューブを採用し，OLAP 的な分析を支援する．移動状況データはストリーム的に配信されてくるので，動的なヒストグラムの構築と管理の支援が求められる．本稿では，移動ヒストグラムの物理モデルとして木構造を採用し，限られたメモリ内で効率よく要約する木の構築手法および管理手法を提案する．さらに，提案手法の性能，精度を実験により評価する．

キーワード　移動ヒストグラム，移動パターン，マルコフ連鎖モデル，インクリメンタル処理，近似処理

A Dynamic Mobility Histogram Construction Method Based on Markov Chains

Yoji MACHIDA†, Yoshiharu ISHIKAWA††,†††, and Hiroyuki KITAGAWA††,†††

† Master's Program in Science and Engineering, University of Tsukuba
†† Graduate School of Systems and Information Engineering, University of Tsukuba
††† Center for Computational Sciences, University of Tsukuba
1–1–1 Tennoudai, Tsukuba, Ibaraki, 305–8573 Japan
E-mail: †y-machida@kde.cs.tsukuba.ac.jp, ††{ishikawa,kitagawa}@cs.tsukuba.ac.jp

Abstract　With the recent progress of spatial information technologies and communication technologies, it becomes easier to track positions of a large number of moving objects in real-time. For the interactive analysis of a huge collection of moving object trajectories, we need to compute mobility statistics in an efficient manner. For this purpose, we propose an approach for constructing a $mobility\ histogram$ to summarize moving object trajectories. The histogram is based on a mobility statistics model called the $Markov\ chain\ model$. To help an interactive analysis performed by a user, we provide a mobility histogram datacube-like logical representation and support an OLAP-style analysis. Since trajectory data is often received continuously as a $trajectory\ stream$, we have to support dynamic histogram construction and maintenace. We introduce a tree structure as the physical representation of a histogram and present a tree construction and maintenance method that efficiently works in a limited memory space. We evaluate the performance and the precision of the proposed method based on experiments.

Key words　mobility histogram, movement patterns, Markov chain model, incremental processing, approximated processing

1. ま え が き

GPS や通信技術の発展に伴い，移動する多数のオブジェクトの移動状況の追跡が容易になっている．また，こうしたオブジェクトの移動状況を蓄積するのに，時空間データベースがますます一般的になっている [3]．大量の移動オブジェクトの移動状況をリアルタイムに追跡し，分析・予測に役立てるには，ストリーム的に配信されてくる移動状況データを効率よく要約することが求められる．さらに移動オブジェクトの移動状況を要約することは，オブジェクトの移動状況を蓄積した時空間デー

タベースにおける問合せ処理でも有用である．ヒストグラムは大量なデータに対して集約することが容易でかつ要約することができる手法であり，ヒストグラム特有の統計量から移動状況データの利用へ展開できることから，本研究はヒストグラムを利用している．

ストリーム配信されるデータを継続的に集計し，コンパクトに移動状況を要約することは重要な研究課題である．そこで本研究グループは，移動データを要約する**移動ヒストグラム**（mobility histogram）の概念を提案し，ストリーム的に配信されてくる移動状況データを効率よく集計するために動的なヒストグラム構築手法の研究を進めている [5]．このアプローチでは，マルコフ連鎖モデルに基づいて移動パターンの移動統計量を集約する．移動オブジェクトの移動軌跡が送られるたびに，インクリメンタルにヒストグラムの更新を行う．精度を落とさずに効率的な処理が実現できること，また，コンパクトにヒストグラムを表現できることが課題となる．

そこで [5] では，ヒストグラムに対するデータ構造の提案と，初期段階からメモリを積極的に利用し，移動オブジェクトの移動状況を正確に反映する手法（素朴な方式）を提案した．本研究では，素朴な方式の概要を述べ，移動状況データの分布から近似的に表現する手法 (近似方式)，近似方式の精度を改善する手法（ビットマップ併用方式）の提案を行い，素朴な方式を含めた 3 方式について比較実験を行う．

2. 関 連 研 究

ヒストグラムは，データを要約するための一手法であり，データベースの分野でも，特に問合せ最適化や近似問合せにおいて盛んに研究開発が進められている [4]. [2], [9] では，蓄積された移動データに対する問合せ選択率を効率的に推定するための研究が行われている．本研究で対象とするヒストグラムは，特殊な制約はあるが多次元のヒストグラムと分類される．本研究の目的とする移動軌跡データのリアルタイムの集積を実現するためには，インクリメンタルな更新に対応するために，とくに効率性が求められる．

近年では，蓄積された移動データから効率的に全体の傾向をおさえ，移動予測を行う上でますます集約情報が重要になってきている．時制，空間データに対する集約クエリの演算に適した手法や集約クエリの演算コストを減らすモデルを提案したサーベイに [7] がある. [8] では，過去，現在，未来のクエリに対応できるように時空間ヒストグラムを用いて移動情報を集約する研究が行われている．一部の過去の移動情報や現在の移動情報における時空間集約情報をメモリ上に持たせることで，精度を犠牲にする一方で，最頻出のクエリに対応できるように効率性を実現している．

移動状況データをヒストグラムとして表現することで，大量の移動オブジェクトの移動状況を効率よく分析予測することが可能になる．例えば，移動オブジェクトがどこにいた可能性が高いのか，移動オブジェクトが今後どこに行く可能性が高いのか，ある領域においてオブジェクトがどのように移動するのかなどは統計量をもとに推定できる. [2], [8], [9] と比較した本研究の他の特徴としては，1) 移動状況データをマルコフ連鎖モデルにあてはめて移動パターンを形成している点，2) 移動パターンを複数の解像度を用いて集積している点，3) ストリーム的な移動軌跡データのリアルタイム処理である．静的なデータを集約するのではなく，大量のストリームデータを効率的に集約する点を特徴とする．

3. マルコフ連鎖モデルに基づいた移動統計量

まずここでは，時空間移動分析を捉える上でマルコフ連鎖モデルの概念について説明する．時空間データ分析における**マルコフ連鎖モデル**（Markov chain model）は，ある地域から別の地域へある期間内にどの程度の人口が移動したなどの，移動オブジェクトの時空間的な移動傾向の把握に用いられる [10]．本研究で用いるパラメータを表 1 に示す．

表 1　パラメータとその定義

パラメータ	定義
P	各次元の分割レベル
R	セルの総数
n	マルコフモデルの遷移次数
m	最大分割レベル
N	ヒストグラムの最大ノード数
W	一定シーケンス数

準備として，本手法における移動パターンのモデリングのために用いる，2 次元平面の番号付け手法について述べる．この手法は 2 次元平面を 1 次元で順序付けする手法の一つである **Z-ordering** [6] に基づいている．本研究では，詳細度の変化にも単純なシフト演算で柔軟に対応できる点に着目して利用している．

2 次元空間が各次元ごとに 2^P（P は分割レベルを表す）個ずつ，$R = 2^{2P}$ 個のセルに分割されているとする (図 1 は $P = 2$ の場合)．この分割のことを**レベル P の分割**と呼ぶ．各セルには，$2P$ ビットのセル番号を付与する．セル番号の上付き数字は，空間分割レベルを表す．

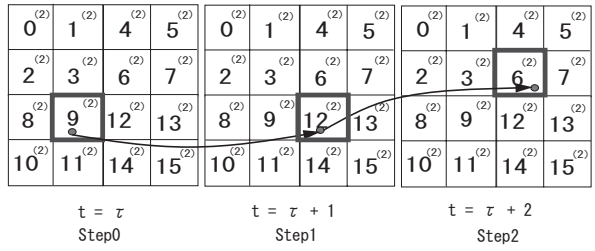


図 1　マルコフ連鎖モデルの概念

このレベルの概念を用いることで空間分割の粒度を指定できる．レベルの異なる分割を対比して，粗い分割の方を上位の空間分割，細かい分割の方を下位の空間分割と呼ぶ．

図 1 は，時刻 $t = \tau$ でセル 9 にいたオブジェクトが，次の時刻 $t = \tau + 1$ でセル 12 に，そして $t = \tau + 2$ の時点でセル 6 に移動した状況を示している．

4. ヒストグラムの論理構造

4.1 データキューブによる遷移シーケンスの集計

軌跡ストリームには $< id, x, y, t >$ のフォームを持つことを想定している．id はオブジェクトの id を，x と y は時刻 t（t：整数）における x-y 座標値を表す．軌跡ストリームが配信されると，前節のモデルに基づいて，マルコフ遷移次数が $n = 2$ なら，$9 \rightarrow 12 \rightarrow 6$ のようなセル番号の**遷移シーケンス**に容易に変換できる．これを一般的に拡張して，n 次のマルコフ遷移にも対応できる．移動ヒストグラムはこの遷移シーケンスの集約および要約をすることで構築される．集約や要約をする上で，遷移シーケンスの出現頻度を捉えるのに適切な表現モデルが望まれる．そこで本研究では，移動ヒストグラムの論理表現として，**データキューブ**を用いる．n 次のマルコフ連鎖の場合，ヒストグラムを $n+1$ 次元のデータキューブとして構成する．

$n = 2$ の場合のデータキューブ表現を図 2 に示す．これはレベル $P = 1$ の分割の場合であり，2 次元平面を $R = 2^{2P} = 4$ 個のセルに分割しているため，データキューブには $R^{n+1} = 64$ 個のセルが存在する．Step 0, 1, 2 という各次元は，それぞれマルコフ連鎖のステップに相当する．たとえば $1 \rightarrow 1 \rightarrow 2$ という遷移シーケンスが生成されると，対応するキューブセルに 1 が加算される．“ Sum ”の列は，各次元の値ごとの和を表す．

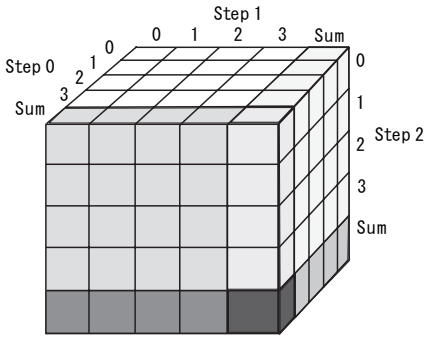


図 2　ヒストグラム論理構造

5. ヒストグラムの物理構造

データキューブ形式のヒストグラムの直接的な実装はオーバヘッドが大きい．2 次元平面を各次元ごとに 2^P 個に分割した場合，空間セルの総数は $R = 2^{2P}$ 個となり，n 次のマルコフ遷移の場合には R^{n+1} 個のキューブセルが必要となる．たとえば，$P = 5, n = 2$ の場合，$R^{n+1} = 1024^3$ となり，約 10 億個のキューブセル数となり，各セル値を 2 バイトで表しても 2GB のサイズとなる．有限なメモリを効率的に使うためには，物理構造は別のデータ構造として考える必要がある．そこで，大規模なデータキューブを近似的に表現する実装方式を以下に述べる．

5.1 素朴な方式による構築

5.1.1 基本的アイデア

移動ヒストグラムは *quad-tree* と *k-d tree* [6] を統合したような木構造で表現する．ここでは，まず，基本的なアイデアを示すために，単純化した木構造について説明する．より具体的な構造は次の節で説明する．

まず，入力の移動軌跡データにおいて，$2^{(P)} \rightarrow 10^{(P)} \rightarrow 12^{(P)}$ のように，分割レベル P でのセルの番号付けがなされていると想定する．そこで，P がこれ以上高い精度でのデータ表現にならない場合，分割レベル P を**最大空間分割レベル**と呼び，m で表記する．

木の各内部ノードは 0 個以上 4 個以下の子ノードを有し，リーフノードはそのノードに対応する移動軌跡の数を集積するためのカウントを有する．ここでは，2 次の遷移シーケンス $a^{(m)} \rightarrow b^{(m)} \rightarrow c^{(m)}$ が木に挿入されることを考える．$a^{(m)}$, $b^{(m)}$, $c^{(m)}$ はセル番号を表す．$a^{(m)}$ をステップ 0 のセル, $b^{(m)}$ をステップ 1 のセル, $c^{(m)}$ をステップ 2 のセルと呼ぶ．基本的には以下のような処理が行われる：

（1） $a^{(m)}$，$b^{(m)}$，$c^{(m)}$ をバイナリ表記して，まず，$a^{(m)}$ の上位 2 ビットを取り出す．その内容が 00 $(= 0^{(1)})$，01 $(= 1^{(1)})$，10 $(= 2^{(1)})$，11 $(= 3^{(1)})$ のいずれかであるかに応じて，対応する辺を辿り，子ノードへと達する．ただし，そのような辺がない場合には，新たに辺を作成する．

（2） $b^{(m)}$，$c^{(m)}$ の順に上位 2 ビットを取り出し，その値に応じて同様に子ノードにアクセスする．これまでの処理は空間分割レベル 1 のおおまかな遷移に対応する．

（3） 次に，$a^{(m)}$ の次の 2 ビット，$b^{(m)}$ の次の 2 ビット，$c^{(m)}$ の次の 2 ビットをそれぞれ取り出し同様の処理を行う．このステップは空間分割レベル 2 に対応する．

（4） このようなステップを繰り返し，$2m$ ビットずつを処理した時点でリーフノードに至る．このリーフノードまで至る過程において，各ノードのカウントを 1 ずつインクリメントする．

n ステップの遷移ごとに辺を順に展開する手法は *k-d tree* の考え方に基づいている．一方，空間の 4 分割を行って粒度を変えながらデータを表現するアプローチは *quad-tree* が基礎となっている．このような意味で本手法は両者を統合したものといってよい．

図 3 に $m = 2$ の場合について，$3^{(2)} \rightarrow 6^{(2)} \rightarrow 12^{(2)}$ を追加する例を示す．図 3 の点線はノードを辿るような遷移シーケンスがまだ到着していないことを表しており，実際にはそれ以下の部分木は存在しない．実線は，ノードを辿るような遷移シーケンスが過去に挿入されたことを表している．

5.2 問合せ処理

移動ヒストグラムの構築後は，どのような移動パターンがあるのか，その移動パターンがどのくらい存在するのかに応えることが求められる．それには問合せを行えばよい．本研究での問合せは，追加と同様に遷移シーケンスを用いる．次節で問合せ処理方式について取り上げる．

5.2.1 基本的アイデア

まず，最大空間分割レベルが $m = 2$ の場合について，例を用いて処理の概要を説明する．

（1） 移動軌跡の各セルのレベルが最大空間分割レベルと一致する場合：たとえば，$3^{(2)} \rightarrow 6^{(2)} \rightarrow 9^{(2)}$ を考える．遷移シー

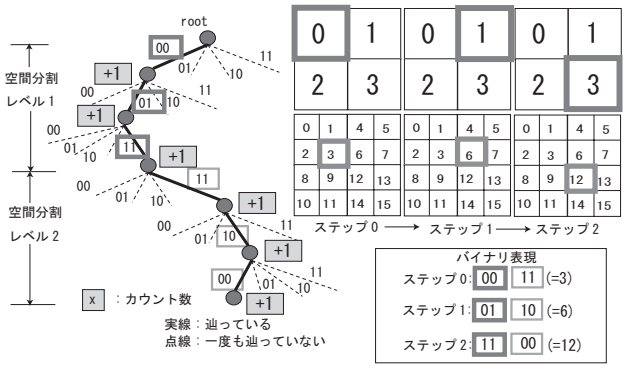


図 3　木構造 ($3^{(2)} \to 6^{(2)} \to 12^{(2)}$，$m = 2$)

ケンスの挿入と同様に，各ステップをバイナリ表記して各レベルごとに 2 ビットずつ辿っていき，目的のカウントを返す．

（2） 移動軌跡の各セルのレベルが最大空間分割レベルよりも粗い場合：例として $1^{(1)} \to 1^{(1)} \to 9^{(2)}$ を考える．(1) と同様に，各ステップをバイナリ表現にして，各レベルごとに 2 ビットずつ辿っていく．この場合は，空間分割レベル 1 の $01 \to 01 \to 10$ の後，$\{00, 01, 10, 11\} \to \{00, 01, 10, 11\} \to 01$ の $4 \times 4 \times 1 = 16$ 通りの経路をすべて辿り各カウントを調べ，総和をとる必要がある．

（3） 移動軌跡のセルのレベルが最大空間分割レベルよりも大きい場合：たとえば，$3^{(2)} \to 25^{(3)} \to 9^{(2)}$ を考える．$3^{(2)} \to 25^{(3)} \to 9^{(2)}$ は，空間分割レベル 2 までの木では表現できないので，カウントを近似的に推定する必要がある．そこで，まず 25 をバイナリ表記すると，(011001) となる．この上位 4 ビットが $25^{(3)}$ の上位空間に相当する．すなわち，その上位空間は $3^{(2)} \to 6^{(2)} \to 9^{(2)}$ である．$3^{(2)} \to 6^{(2)} \to 9^{(2)}$ のカウントはヒストグラムのノードに保持されている．$6^{(2)}$ の下位空間には $\{24^{(3)}, 25^{(3)}, 26^{(3)}, 27^{(3)}\}$ があるので，4 等分したカウントで近似する．

図 4 に問合せの処理概要 (1) の場合について，$3^{(2)} \to 6^{(2)} \to 12^{(2)}$ を問合せする例を示す．そのときの問合せ結果は図 4 より，**4** が返る．

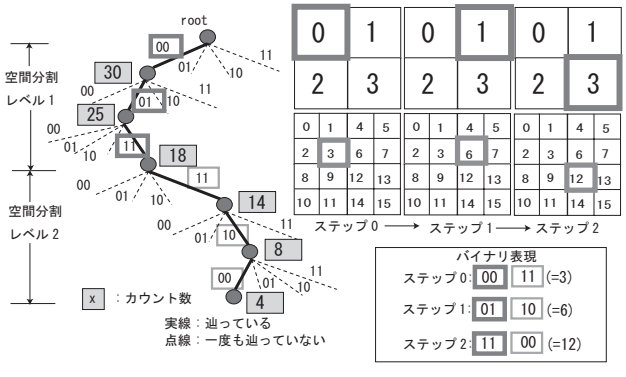


図 4　問合せ処理例：($3^{(2)} \to 6^{(2)} \to 12^{(2)}$，$m = 2$)

5.2.2　問合せ処理アルゴリズム

問合せ処理のアルゴリズムを**アルゴリズム 1** に示す．このアルゴリズムは再帰的な関数 est_count() を定義している．この関数は 4 つの引数 $node$, $qseq$, $levels$, $step$ をとる．$node$ は処理対象のノードへのポインタ，$step$ は現在処理対象の遷移ステップを表す．$qseq$, $levels$ はそれぞれ問合せ遷移シーケンス中の領域番号の配列およびレベル番号の配列である．たとえば，$3^{(1)} \to 14^{(2)} \to 15^{(2)}$ という遷移シーケンスが問合せとして与えられた場合，$qseq[0] = 3$, $qseq[1] = 14$, $qseq[2] = 15$ であり，$levels[0] = 1$, $levels[1] = levels[2] = 2$ となる．

ある遷移シーケンスの出現回数の推定を求める問合せがユーザから与えられると，まず 2 つの配列 $qseq$ と $levels$ を構築し，est_count($root, qseq, levels, 0$) という関数呼び出しを行う．$root$ はヒストグラムのルートを表し，0 はまず最初は 0 番目の遷移ステップに関する処理を行うことを意味する．

Algorithm 1 est_count($node$, $qseq$, $levels$, $step$)

```
1:  input node: 対象ノード, qseq: 問合せの遷移シーケンス, levels: 各
    遷移のレベル配列, step: 処理対象のステップ
2:  if (i = 0, ..., n について levels[i] ≡ 0)
3:      return node.count;  //問合せ終え，目的のカウントを返す
4:  else if (levels[step] > 0)
5:      shift := 2(levels[step] − 1);    //2 ビット抽出のシフト量
6:      c := (qseq[step] >> shift) & 0x3;    //探索する子ノード番号
7:      levels[step]−−;    //次レベルを表す
8:      if (node.child[c] ≡ NULL)    //探索途中で辿る辺がない
9:          count := node.count;
10:         for (i = 0; i ≦ m; i++)
11:             count := count / 2^levels[i];    //カウントの推定値を計算
12:             return count;    //近似解
13:     else
14:         return est_count(node.child[c], qseq, levels,
                    (step + 1) mod (n + 1));  //次ステップへ再帰処理
15: else
16:     count := 0;
17:     for (c = 0; c ≦ 3; c++)
18:         if (node.child[c] ≢ NULL)
19:             count += est_count(node.child[c], qseq, levels,
                        (step + 1) mod (n + 1));  //子ノードに分岐して集計
20:     return count;  //集計したカウントを返す
```

アルゴリズムの概要を説明する．このカウント処理では，ルートから 1 ノードずつ辺を辿って，対象となるカウントを有するノードまで探索し，そのカウント値を返す．木の探索処理は再帰呼出しで実現する．ただし，場合によっては 1 つの問合せに対し複数の辺に分岐して探索を行う必要も生じることに注意する．

まず，2 行目で与えられた問合せをどこまで処理したかを表現するのに $levels$ 配列の値で判別していく．典型的な処理は，4-14 行目で $node$ からその子に対し 1 つだけ辺を辿る処理を行う．8 行目の条件が偽の場合，14 行目で再帰呼び出しを実行し，子に対してカウント処理を引き継ぐことを表す．それに対して，8 行目の条件が真の場合，5.2.1 節の (3) の場合のような問合せに対してヒストグラムの解像度が高くないことを表し，11 行目で近似的なカウント値の推定を行う．続いて，15 行目では分岐処理が必要な場合で，5.2.1 節の (2) の場合に相当する．4 つの辺に対し，est_count() を 4 回再帰呼び出しし，その結果の和を

とる．

5.3 近似方式による構築

5.1 節の素朴な方式のような構築方法をとった場合，問題が発生する．

データ構造の近似表現　5.1 節で述べたヒストグラムの物理構造は，シーケンスのカウントを正確に表現できるが，多数のノードを必要とすることから格納コストの面でも問題がある．また，近似を行うことで単なるデータではなく要約された移動パターンの表現が可能となり，移動状況の概略を把握することが容易になる．そこで，与えられたノードの数の上限 N のもとで近似的にヒストグラムを構築することにする．

5.3.1 ヒストグラム構築アルゴリズム

遅延処理　ヒストグラムの木構造構築における基本的アイデアは**遅延処理**である: すなわち，移動シーケンスの追加に対しすぐに木構造を展開するのではなく，徐々に木を展開していく．

まず，2 次のマルコフ遷移 $(n = 2)$ を集積する例を示す．初期段階ではルートノードのみが構築され，移動シーケンスを順次入力ストリームから受け付ける．現在，最初の 100 件のシーケンスが与えられた時点であるとし，各シーケンスがステップ 0 においてレベル 0 のどのセル内に位置するかを調べたとき，

cell0$^{(0)}$: 22, cell1$^{(0)}$: 23, cell 2$^{(0)}$: 27, cell3$^{(0)}$: 28

となっていたとする．この場合，カウント数に大きな差が見られないことから，木を細分化してもあまりメリットがないと考える．そこで，引き続き移動シーケンスの集積を続ける．一方，カウント数が

cell0$^{(0)}$: 10, cell1$^{(0)}$: 20, cell 2$^{(0)}$: 50, cell3$^{(0)}$: 20

となっていたとする．このような場合，カウントにばらつきが大きいことから，木を細分化する．すなわち，ルートノードの下に 4 つの子ノードを作成する．これにより，よりパターンが集積した領域（この例では 2$^{(0)}$）を中心に木構造を展開する．ルートノードが展開されると，次にはルートノードの子ノードについて同様に見張る．ただし，今回はステップ 1 のセルに着目する．このような処理を繰り返すことで，ヒストグラムの最大数 N に達するまで，ばらつきが大きい領域を中心に細分化を続ける．N に達したら，その後はヒストグラムの木構造を固定して，カウントのインクリメントだけを行う．木構造の展開の概念を図 5 に示す．

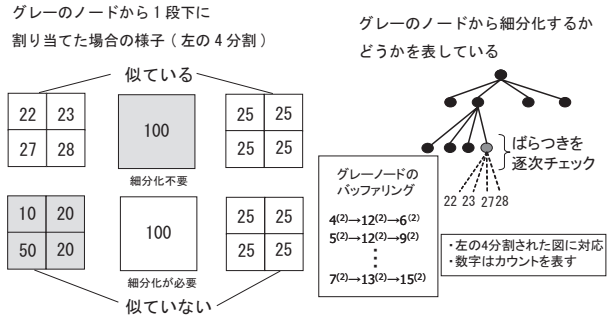


図 5　近似方式のばらつきの判定

構築アルゴリズム　ヒストグラムの構築処理のアルゴリズムを**アルゴリズム 2** に示す．

Algorithm 2 Construction algorithm

algorithm CONSTRUCT_HISTOGRAM

```
 1: root := node_gen();          //root ノードの生成
 2: count := 1;                  //割り当てノード数 (1 は root のこと)
 3: for (i = 1; i ≦ W; i++)
 4:     seq := seq_get();        //シーケンスを受け取る
 5:     buf_insert(seq);         //シーケンスをバッファに格納
 6:     (node, c) := find_node(root, seq);   //root にあるシーケンスがノードの何番目のスロットにあるかを探索
 7:     node.count[c]++; //探索したスロットのカウントをインクリメント
 8:     if (count < N) and is_non_uniform(node)
 9:         expand(node);        //木を展開
10:         count += 4;          //割り当てノード数を 4 個増やす
11: save(root);                  //ヒストグラムの木構造を保存
```

function find_node(root, seq)

```
 1: node := root;                //ノードの初期化
 2: shift := 2(m − 1);           //2 ビット抽出するためのシフト量
 3: for(i = 0; i ≦ m − 1; i++)
 4:     for(j = 0; j ≦ n; j++)
 5:     c := (seq[j] ≫ shift) & 0x3;   //探索する子ノード番号
 6:         if (has_children(node)
 7:         node := node.child[c];     //子ノードへ探索
 8:         else
 9:             return (node, c);      //ノードの何番目のスロットかを返す
10:     shift −= 2;              //次レベルの 2 ビット抽出するためのシフト量
```

まずメインアルゴリズムについて述べる．$root$ はルートノードであり，$count$ は割り当てされているノード数である．W は一定のシーケンス数であり，W 個の入力シーケンス seq についてヒストグラム構築が行われる．5 行目の buf_insert() でシーケンス seq がバッファに格納される．バッファはハッシュ表により管理され，同じシーケンスは同一エントリにマップされ，カウントのみがインクリメントされる．6 行目では seq に対応するノードが find_node() で見つけられる．c $(0 \leqq c \leqq 3)$ は，seq が $node$ の何番目のスロットに対応するかを表す．スロットは 2 つの意味で使い分けている．1 つは中間ノードの場合で，このときのスロットはポインタとして扱う．2 つめは葉ノードの場合で，このときのスロットはカウントを指す．7 行目ではヒストグラムのカウントをインクリメントする．ノードを展開する必要があるかを見つけるために，$count < N$ であれば，8 行目で is_non_uniform() 関数を呼ぶ．この関数の実現法については後で詳しく述べる．ノードの内容が一様でないならば 9 行目でノードを展開する．expand() 関数では，$node$ に新たに子ノードを追加し，値の設定を行う．新たに子ノードが 4 個追加されるので，10 行目で $count$ に 4 を追加する．save() 関数は，ヒストグラムの木構造を保存する．

find_node() 関数は，木をルートノードから辿り，シーケンス seq に対応するノードを特定し返す．$i = 0$ から m までの分割レベルのそれぞれについて，ステップ 0 から n までについてノードを辿る．5 行目は何番目の子ノードを辿るかを決定するための処理である．Z-ordering の使用により，単純なシフト演算を用いて対象の番号が特定できる．

ばらつきの検出　問題となるのが，メインアルゴリズム 8 行目の is_non_uniform() の実現手法である．ばらつきの検出においては，

- 統計的に明確な指標に基づいていること
- 効率的に実現できること

の二つを重視する．特に，ばらつきが統計的に明らかである場合は早い時点で分割処理を行うことが重要である．これにより，ヒストグラムの構造を早期に確定し，集計を効率的に行うことが可能となる．

そこで，本研究では χ^2 **適合度検定** (χ^2 test for goodness of fit) [11] を用いる．対象となる領域を 4 分割したときのカウントを，

x_{00}	x_{01}
x_{10}	x_{11}

のように x_{00}，x_{01}，x_{10}，x_{11} とおく．ここでの**帰無仮説** H_0 は，各セルにデータが入る確率は一様である (1/4) というものであり，χ^2 値は，

$$\bar{x} = \frac{x_{00} + x_{01} + x_{10} + x_{11}}{4} \tag{1}$$

$$\chi^2 = \sum_{c \in \{00,01,10,11\}} \frac{(x_c - \bar{x})^2}{\bar{x}} \tag{2}$$

で求められ，この値は自由度 $4-1=3$ の χ^2 分布に従う．よって，たとえば有意水準 0.05 の場合，χ^2 統計の表から 7.815 であるため，$\chi^2 > 7.815$ になった時点で分布が一様ではない．すなわち is_non_uniform() が false を返すことになる．

ただし，χ^2 適合度検定を用いる場合は，データ量が少ない場合の注意が必要である．特に (2) 式において x_c の値が非常に小さい場合が問題となる．そこで，本手法では，データのカウント数が小さい場合には χ^2 適合度検定ではなく，ロバストなノンパラメトリックな手法である 2 項検定の拡張手法を用いる．詳細については省略する．

5.4　ビットマップ併用方式による構築

近似方式では，基本的に全ての移動パターンに対して公平に推定を行う．そのため，実際に出現しない移動パターンを推定していることもあり得る．そこで多少の制約はあるものの移動パターンが実際に出現するのか出現しないのかを見極めることは，推定を行う上できわめて有効と言える．本研究ではその有効な方法として，ビットマップを用いる．ビットマップはデータキューブのバイナリ表現と位置づけることができる．素朴な方式で構築したときの移動パターンのカウントが 1 以上のものはセルに 1，それ以外ならセルに 0 と考える．今回は空間分割レベル 3 のビットマップ（32KB 相当）を利用した．図 6 にビットマップ併用方式の概要を示す．

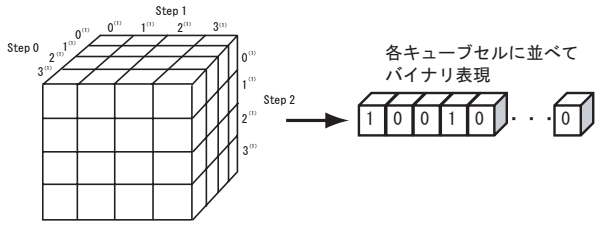


図 6　ビットマップの概要

6. 評 価 実 験

6.1　実験データと実験環境

本実験では，Brinkoff により作成された移動オブジェクトデータ生成ソフトウェアを用いる [1]．これは実際の市街地の道路ネットワーク上を自動車などが移動する際の移動の状況をシミュレーションするシステムであり，これによって生成された移動データを実験において使用する．今回扱うデータは，このシステムで提供されているドイツ Oldenburg 市の市の中心部（約 2.5 × 2.8km）の道路ネットワークにおいて，どの時点でも移動オブジェクトが常に 1000 個存在するように生成している．移動オブジェクトの生成パラメータの設定を表 2 に示す．

表 2　移動オブジェクト生成パラメータ

パラメータ (Declared parameter in generator)	値
初期の移動オブジェクトの数 (obj./begin)	1000
単位時間当たり生成する移動オブジェクトの数 (obj./time)	52
生成単位時間 (maximum time)	50
ノード数 (number of nodes)	6105
エッジ数 (number of edges)	7035
移動オブジェクトのクラス数 (classes)	6

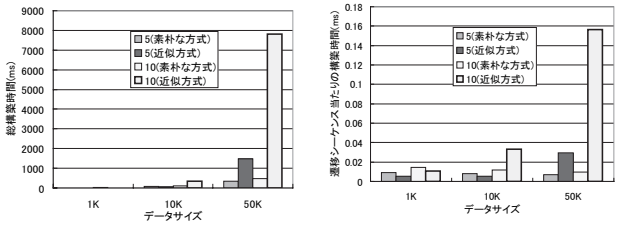
図 7 にこのシステムの移動軌跡データを示す．下の実験において，Pentium4 3.2GHz の CPU，１ GB のメモリを持つ PC を使用して実験を行った．加えて地図領域を 1024 × 1024 のセル領域に分割した場合を想定している．

図 7　移動軌跡データ

6.2　ヒストグラム構築時間

定常状態での追加処理を行う．素朴な方式では各ノードには，カウンタに 2 バイト，各ポインタに 4 バイト，計 20 バイト使用している．図 8 に構築時間について示す．図 8 の 5，10 はヒストグラムの構築の最大空間分割レベルを，さらに 1K は 1000 件，10K は 1 万件，50K は 5 万件のデータサイズを表す．

素朴な方式はデータサイズを S とすると，ヒストグラムの総構築時間は S にほぼ比例する．そして遷移シーケンス当たりの



(a) 構築時間　　(b) シーケンス当たりの構築時間

図 8　件数変化に対する構築時間

構築時間は小さくほぼ定数である．一方，近似方式ではデータサイズが大きくなるにつれ，総構築時間も大きくなっている．遷移シーケンス当たりの構築時間も同様である．ビットマップ併用方式の構築時間は近似方式と同じである．

近似方式の構築時間がデータサイズに対して大きくなる理由は，各遷移シーケンスごとに一様性の判定を行っているので，データサイズが大きいと χ^2 適合度検定を行う頻度も増え，時間を要していると考えられる．さらに，近似方式の特徴であるノードの分割処理に時間を要していることも考えられる．ただし，多少精度は落ちるがヒストグラムのサイズを縮小することにより，構築時間を短縮することができる．これにより分割処理や分割処理時に要するバッファ上のスキャンの回数を減らすことができ，構築時間が急激に上昇することは無くなると期待できる．また，現状でも遷移シーケンス当たりの構築時間が 0.16(ms) なので，遅いとはいえない．すなわち，本研究の本質的に問題になるのはサイズと精度のトレードオフである．

6.3　ヒストグラムのサイズ

次に構築したときのヒストグラムのサイズを図 9 に示す．データサイズ 1K は 1000 件，10K は 1 万件，50K は 5 万件を表す．素朴な方式の 1 個のノードは実装上 20 バイト，近似方式の 1 個のノードは実装上 16 バイト使用している．ビットマップ併用方式は近似方式のヒストグラムサイズに対してビットマップを加えている．

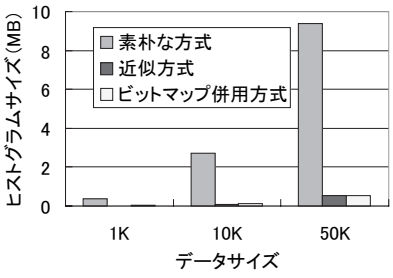


図 9　ヒストグラムのサイズ

どの方式も，データサイズを S とするとヒストグラムサイズは S にほぼ比例している．素朴な方式のヒストグラムサイズは他の方式に比べて非常に大きくなっており，実用性に欠けると判断できる．よって，ヒストグラムサイズでは近似方式とビットマップ併用方式が有効といえる．

6.4　問合せ処理時間

複数の問合せパターン 100 個を与えた際の問合せ処理時間を調べた．問合せのパターンには (1) 移動軌跡の各セルのレベルが最大空間分割レベルと一致する場合（問合せ例：$909876^{(10)} \rightarrow 397555^{(10)} \rightarrow 399468^{(10)}$），(2) 移動軌跡の各セルのレベルが最大空間分割レベルよりも粗い場合（問合せ例：$2^{(2)} \rightarrow 2^{(2)} \rightarrow 2^{(2)}$，$53662^{(9)} \rightarrow 66816^{(9)} \rightarrow 109748^{(9)}$) を用いた．分割された地図領域上の問合せ例 $3^{(2)} \rightarrow 6^{(2)} \rightarrow 12^{(2)}$ を図 10 に示す．太枠は空間分割レベル 2 を，細枠は空間分割レベル 4 を表している．

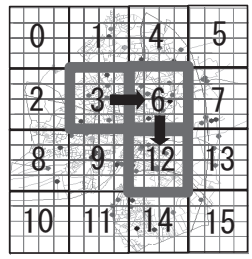
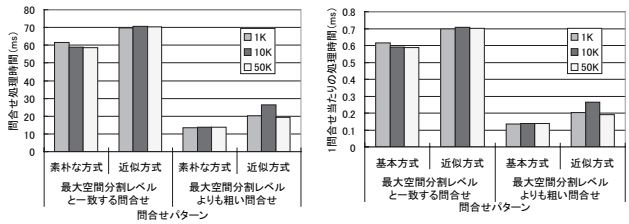


図 10　問合せ例

いずれの問合せも全て 2 次のマルコフ遷移の遷移シーケンスで行った．問合せパターンごとの問合せ処理時間を図 11 に示す．



(a) 問合せ処理時間　　(b) 1 問合せ当たりの処理時間

図 11　問合せパターンにおける問合せ処理時間

素朴な方式の問合せ処理時間では，データサイズに依存せずにほぼ一定の処理時間を要することがわかる．上記の場合の問合せでは葉ノードの数を K（$K \ll S$）とすると，処理時間はほぼ K に比例する．すなわち，詳細なヒストグラムほど問合せ処理時間が大きくなる傾向にある．

近似方式では，素朴な方式の問合せ処理時間と傾向はほとんど変わらない．敢えて言えば，近似方式の問合せ処理時間が素朴な方式よりも微量であるが大きい．これは，素朴な方式と近似方式における処理アルゴリズムの実装レベルの差と考えられる．ビットマップ併用方式では，近似方式とほぼ同じである．

6.5　精　　度

近似方式のヒストグラムの精度評価を行う．ここで，真のカウントとして素朴な方式のノードカウント ACT_i，推定カウントとして近似方式のノードカウント EST_i を定義する．

精度を表す指標に以下の式を用いる．

$$\sqrt{\frac{1}{(2^{2P})^{(n+1)}} \sum_{i=1}^{(2^{2P})^{(n+1)}} \left(\frac{ACT_i - EST_i}{ACT_i} \right)^2} \quad \text{(相対誤差)} \quad (3)$$

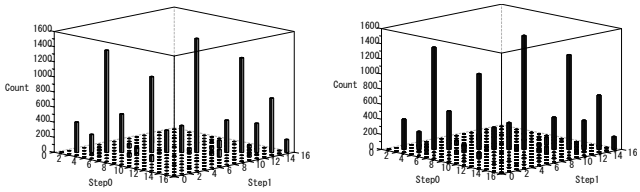
(3) 式の分母が 0 になることもありうる．それは，実際にその遷移シーケンスが出現しなかったことを表す．そこで，上記の精度指標式にラプラス推定を考慮した補間を行うことにする．

ラプラス推定を考慮した真のカウントおよび推定値のカウントの補間は以下の式である．

$$A\hat{C}T_i = \frac{SeqC_i + 1}{S + B} \times S, \quad E\hat{S}T_i = \frac{SeqC'_i + 1}{S + B} \times S \qquad (4)$$

（4）式の $SeqC_i$，$SeqC'_i$ は i 番目の遷移シーケンスのカウント，$S(= \sum SeqC_i)$ はデータサイズ，$B(= R^{n+1})$ はバケット数を表している．データサイズが非常に大きい場合は，補間後のカウント値 $A\hat{C}T_i$，$E\hat{S}T_i$ が (補間前のカウント値+1) に収束する．

素朴な方式と近似方式の精度評価の前に，それぞれの方式によるカウントを図 12 に示す．図 12 は 1 次マルコフ連鎖の遷移シーケンス 10000 件を対象とした空間分割レベル 2 の全問合せ (256 個) の結果である．素朴な方式と近似方式のカウントにほ



(a) 素朴な方式によるカウント　　(b) 近似方式によるカウント

図 12　1 次マルコフ連鎖のカウントイメージ ($S = 10000$ 件，$P = 2$)

とんど差がないことがわかる．同一セル内（3 → 3, 9 → 9 など）の移動パターンが頻繁に出現している．続いて，素朴な方式によるカウントから近似方式によるカウントを引いて絶対値をとった差を図 13 に示す．

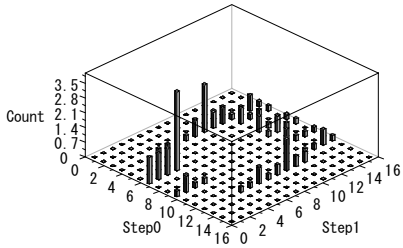
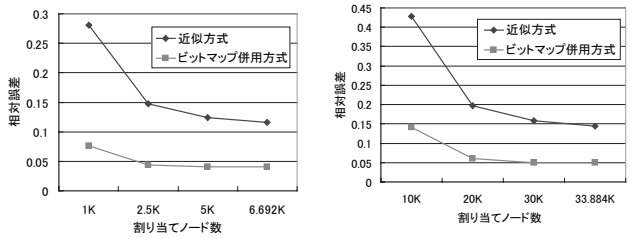


図 13　カウントの誤差イメージ

差をとった場合も，もともとの各移動パターンは数百〜千数百というカウントを持つことを考慮すれば，誤差が小さいものと解釈できる．

以下に近似方式とビットマップ併用方式のラプラス推定による補間を行った相対誤差について図 14 に示す．2 次のマルコフ連鎖の遷移シーケンス 10000 件，50000 件を対象とした空間分割レベル 3 の全問合せ (262144 個) の結果である．割り当てノード数の 6.692K，33.884K は最大割り当てノード数である．

いずれの方式も相対誤差は割り当てノードが増えるにつれて，精度が向上していることがわかる．そして，近似方式の誤差率は 10% から 15% ぐらいに収束することが言える．それに対して，ビットマップ併用方式の誤差率は 5% とかなり精度がいい



(a) $S = 10000, n = 2, P = 3$　　(b) $S = 50000, n = 2, P = 3$

図 14　精　　度

ことがわかる．移動パターンの出現の有無を表した少量の情報でも推定を行うのに大いに貢献していたといえる．

7. ま と め

本稿では，木構造による移動統計量の推定手法を素朴な方式，近似方式，ビットマップ併用方式の 3 種類のヒストグラムについて提案した．各方式について，ヒストグラムの構築時間，ヒストグラムのサイズ，問合せ処理時間，近似方式およびビットマップ併用方式の精度評価を行った．評価実験から，近似方式やビットマップ併用方式がサイズと精度のトレードオフに効果的であることが検証できた．

今後の課題として，非定常のデータや実データを用いて本手法の有効性を検証していくことが必要である．さらに，効率的な実装，ビットマップを用いたカウントの推定手法の検討が挙げられる．

謝辞

本研究の一部は，日本学術振興会科学研究費基盤研究 (C)(2)(16500048)，旭硝子財団研究助成，稲森財団研究助成及び文部科学省科学研究費特定領域研究 (2)(16016205) による．

文　　献

[1] T. Brinkhoff. A framework for generating network based moving objects. In *GeoInfomatica*, No. 6(2), pp. 153–180, 2002.

[2] Y. Choi and C. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proc. ACM SIGMOD*, pp. 440–451, 2002.

[3] Ralf Hartmut Güting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[4] Yannis Ioannidis. The history of histograms (abridged). In *Proc. VLDB*, 2004.

[5] 町田陽二, 石川佳治, 北川博之. 移動軌跡ストリームデータのためのインクリメンタルなヒストグラム管理手法. 日本データベース学会 Letters, Vol. 4, No. 2, pp. 45–48, 2005.

[6] Shanshi Shekhar and Sanjay Chawla. *Spatial Databases*. Prentice Hall, 2002.

[7] Richard T. Snodgrass, Ines Fernando Vega Lopezd, and Bongki Moon. Spatiotemporal aggregate computation: A survey. In *Proc. IEEE Transactions on Knowledge and Data Engineering 17(2)*, pp. 271–286, 2005.

[8] J. Sun, D. Papadias, Y. Tao, and B. Liu. Querying about the past, the present, the future in spatio-temporal databases. In *Proc. ICDE*, 2004.

[9] Y. Tao, J. Sun, and D. Papadias. Selectivity estimation for predictive spatio-temporal queries. In *Proc. ICDE*, 2003.

[10] G. J. G. Upton and B. Fingleton. *Spatial Data Analysis by Example, Volume II: Categorical and Directional Data*. John Wiley & Sons, 1989.

[11] 矢島美寛, 廣津千尋. 自然科学の統計学. 東京大学出版会, 1996.