2003－DBS－131　(30)
2003／7／17

索引付けされた移動軌跡データからの移動統計量の抽出法の評価

塚本祐一[†]，　石川佳治[††]，　北川博之[††]

概要

空間情報利用の急速な進展および携帯機器 などの普及から，時空間データベースの研究分野では，移動するオブジェクトやユーザの移動状況をデータベースに蓄積し効率よく管理するための研究が盛んに進められている．本稿では，蓄積された移動情報データから移動統計量を高速に抽出し，対話的な移動状況の分析を支援するための手法について述べる．セルの集合に分割された空間上を時間の経過につれてオブジェクトが移動する状況を捉えるための統計量として，マルコフ連鎖モデルが存在する．本研究では，空間索引R-木に蓄積された移動オブジェクトの移動軌跡データから，マルコフ連鎖モデルにおける遷移確率を効率的に求めるための手法について研究を行っている．遷移確率の導出を空間索引を用いた制約充足問題の処理に帰着させ，空間索引の内部情報を利用して効率的に処理を行う点が特徴となっている．本稿ではこの提案手法について述べ，実験のフレームワークを示す．

Evaluation of a Mobility Statistics Extraction Scheme for Indexed Spatio-Temporal Datasets

Yuichi TSUKAMOTO[†]，　Yoshiharu ISHIKAWA[††]，　Hiroyuki KITAGAWA[††]

abstract

With the recent progress of spatial information technologies and mobile computing technologies, spatio-temporal databases which store information on moving objects including vehicles and mobile users have gained a lot of research interests. In this paper, we evaluate an algorithm to extract mobility statistics from indexed spatio-temporal datasets for the interactive analysis of huge collections of moving object trajectories. We focus on a mobility statistics value called the Markov transition probability, which is based on a cell-based organization of a target space and the Markov chain model. The algorithm efficiently computes the specified Markov transition probabilities with the help of a spatial index R-tree. It reduces the statistics computation task to a kind of constraint satisfaction problem that uses a spatial index, and utilizes internal representation of R-tree in an efficient manner.

1　はじめに

今日では，地図データの電子化や GPS データなどの空間測位技術の進展などにより，空間情報利用の拡大が急速に進んでいる．また，携帯電話やモバイル PC などの普及により，モバイルユーザを対象としたデータ管理技術がより重要となってきている．これを受け，**時空間データベース**（spatio-temporal database）の研究分野では，移動するオブジェクトやユーザのためのデータ提供技術などの研究が盛んに進められている [7].

大量の移動オブジェクトに関する移動情報の蓄積や問合せを目的とした移動オブジェクトデータベースでは，問合せの効率化に関する研究が重要な課題の 1 つとなっている．

加えて，時空間データベースからの統計情報の抽出に関しても，近年いくつか提案がなされている．時空間データに関する統計量は，データベース処理の効率化のみならず，蓄積された移動状況データをもとに**移動分析**（mobility analysis）を行う際においても有用である [12]．時空間データの利用拡大により，移動分析のための統計量を効率よく抽出することがより重要となると考えられる．

以上の背景をもとに，我々は時空間データベースから移動オブジェクトの移動状況に関する統計情報を抽出する手法を提案した [14]．そこでは，移動に関する統計量として，本研究では**マルコフ連鎖**（Markov chain）モデルに基づく移動統計量を考え

[†]筑波大学システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[††]筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

ている．時空間データ分析におけるマルコフ連鎖モデルは，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，人や物の時空間的な移動傾向を把握するために用いられる [12]．この統計情報により，ある時点である位置にいるオブジェクトが次の時点でどこに移る可能性が高いといった予測が可能となる．

本研究では特に，移動オブジェクトの移動軌跡が空間索引 R-木に蓄積されている状況を想定し，R-木から効率的にマルコフ連鎖の遷移確率を推定する手法について研究を進めている．R-木から遷移確率を推定する問題は，一種の**制約充足問題**（constraint satisfaction problem）として定式化できる．

本稿では，提案した手法の詳細と評価を行うためのフレームワークを示す．

2 マルコフ過程モデルに基づく移動統計量

図 1 に示すように，空間がセルに分割されているとする．各セルは矩形でなくてはならないが，サイズは必ずしも均一でなくてよいとする．各セルにはセル番号が付けられていて，番号によりセルを特定できるものとする．図は，時刻 $t = \tau$ でセル c_0 にいたオブジェクト A が，次の時刻 $t = \tau + 1$ でセル c_1 に，そして $t = \tau + 1$ の時点でセル c_2 に移動した状況を示している．

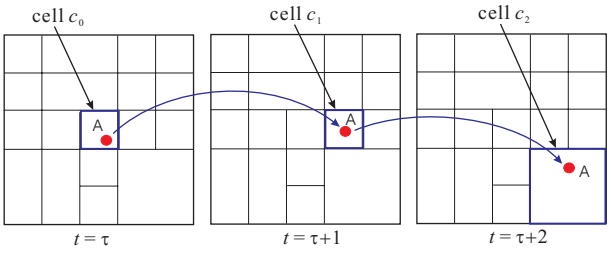

図 1: マルコフ過程モデルの概念

A のように，セル c_0 に現在いるオブジェクトが次の時点でセル c_1 に移動する確率 $\Pr(c_1|c_0)$ を時空間データベース中のデータを用いて予測したいとすると，

$$\Pr(c_1|c_0) = \frac{\sum_{t=0}^{T-1} |\mathrm{objs}(c_0, t) \cap \mathrm{objs}(c_1, t+1)|}{\sum_{t=0}^{T-1} |\mathrm{objs}(c_0, t)|} \quad (1)$$

と計算できる．ただし，$\mathrm{objs}(c_i, t)$ は，時刻 t にセル c_i の領域に含まれているオブジェクトの集合を返す関数である．この式では，各時点 $t = 0, \ldots, T-1$ で c_0 にいたオブジェクトの総数を分母とし，そのようなオブジェクトのうち，次の時点で c_1 に移ったオブジェクトの総数を分子としている．

確率 $\Pr(c_1|c_0)$ は，現在の状態（セル c_0）に依存して次の状態（セル c_1）を予測するという意味で，1 次のマルコフ過程の遷移確率に相当する．これを一般化すると，n 次のマルコフ過程による遷移確率，すなわち，各単位時間ごとに $c_0, c_1, \ldots, c_{n-1}$ とセルを移動してきたオブジェクトが次の時点でセル c_n を訪れる確率は，$\Pr(c_n|c_0, \ldots, c_{n-1})$ と表され，その推測値は

$$\Pr(c_n|c_0, \ldots, c_{n-1}) = \frac{\sum_{t=0}^{T-n} |\bigcap_{i=0}^{n} \mathrm{objs}(c_i, t+i)|}{\sum_{t=0}^{T-n} |\bigcap_{i=0}^{n-1} \mathrm{objs}(c_i, t+i)|} \quad (2)$$

という一般形で与えられる．

なお，上の議論では移動オブジェクトの移動が定常的（stationary）な過程に従い，時刻によって遷移確率が変化しないと仮定している．

マルコフ遷移確率を効率的に求めることができるならば，移動オブジェクトが次の時点でどのセルに移動する可能性が高いかという，一種の経路予測を行うことが可能となる．また，ある時刻 $t = \tau$ における移動オブジェクト集合の移動状況データが与えられたとき，$t = \tau + 1, 2, \ldots$ における移動状況がどうなるかをシミュレーションすることも可能となる．

3 関連研究

3.1 時空間データベースからの統計量抽出法について

データベースの問合せ最適化の問題については，選択率の見積りなどのためにデータベースの統計情報が用いられる．空間データベースに対しては，すでに多くの統計情報の計算手法が提案されている（例：[1]）．しかし，時空間データベースについてはいくつかの提案が見られる程度である．例えば，[4] は，移動する点オブジェクトを格納した時空間データベース上で静的な（時刻に依存しない）範囲問合せを行う際の選択率の推定法を提案している．またこれを一般化して，問合せが動的な（時刻に依存して変化する）場合の選択率の推定法を提案したものもある [11].

本研究では，時空間データベースに対する範囲問合せの選択率の推定を目的とする上記の手法とは異なり，多数の移動オブジェクトの移動状況をマルコフ遷移確率の形で要約して提示することを目的とする．この統計量は，前述のように移動オブジェクトの将来の移動状況の予測のために利用できる点を特徴とする．また，関連研究 [4, 11] と比較した本研究の他の特徴としては空間索引の利用が挙げられる．空間索引の内部情報を活用することで，効率的に統計量の計算を行うことが可能となる．

他の研究として，[13] では，交通状況を蓄積する交通データウェアハウスにおいて，通過車両台数や空間占有率などの各種統計量を高速に求めるための Σ-木を提案している．Σ-木の特徴としては，道路の空間的制約を考慮した索引ノード生成や，分析処理の際の計算コスト削減のためのデータの前処理がある．本研究との相違点としては，本研究が空間索引として一般的な R-木を対象とする点や，統計量としてマルコフ遷移確率を対象する点がある．

3.2 空間索引を用いた制約充足問題の処理

空間索引により索引付けされた移動軌跡のデータからセル間の遷移確率を導くという本研究の目的は，後述のように，移動軌跡データからある種の時間的な制約条件を満たすオブジェクトの組を列挙し集計する処理に帰着できる．関連研究として，[9] では，空間オブジェクトに対する R-木から，指定された空間的関連を満たすオブジェクトの組をすべて列挙する手法を示している．この手法は，空間的な制約条件に基づく制約充足問題を解くことを目的としており，問合せの例としては「$\mathrm{overlaps}(x, y)$ と $\mathrm{north}(y, z)$ を満たすすべての空間オブジェクトの組 (x, y, z) をすべて列挙せよ」がある．特徴としては，空間索引を木の根から葉へ枝刈りをしながら下っていき，制約を満たすオブジェクトの組を効率的に探索する点がある．

空間索引から制約を満たす組を列挙する処理は，空間索引を用いた空間結合の一種と考えることができる．本研究は空間的関連を充足するオブジェクトの組を探索する [9] とは異なり，マルコフ連鎖から導かれる時間的制約を満たすオブジェクトの組を索引から効率的に導く点が大きく異なっている．なお，指定されたセルに関する空間的な制約条件は，時間的な制約を満たす解の探索範囲を限定するために利用される．索引を用いた一種の結合処理と捉えられる点は [9] と共通するが，用いられる制約条件は大きく異なっている．

4 空間索引による時空間オブジェクトの索引付け

4.1 移動軌跡データのための索引手法

2 節で述べたような推定を行うには，時刻 t にセル c 内に存在したオブジェクトの集合 $\mathrm{objs}(c, t)$ をいかに効率よく見つけるかが鍵となる．そのためには，索引の適切な利用が不可欠である．空間索引として一般的なものに，R-木が挙げられる．R-木による移動軌跡データの管理としてすでに提案されているアプローチとしては，3D R-木（時間の次元を加えて 3 次元で軌跡を表す）[8] や，STR-木（移動軌跡の検索を目的とする）[10] がある．本研究では，R-木の利用を想定するが，それに特殊な拡張を必要としない手法を特に検討する．

4.2 索引付けの具体例

以下では，移動軌跡を点の集合として離散的に表現する場合と線分の集まりで表現する場合のそれぞれについて，本研究のアルゴリズムが想定する索引付け方式について述べる．

例として，1 次元空間でのオブジェクトの移動の例を考える．図 2(a) は，オブジェクト A, B が時刻 $t = 0$ から $t = T\,(= 8)$ まで x 軸上を移動する様子を示している．移動経路は曲線で表現されている．実際の移動軌跡はこのように複雑であるが，計算機上での表現には何らかの近似が必要である．図 2(a) に示した経路上の点は，各時刻において移動軌跡をサンプリングしたものである．この近似により，各オブジェクトの経路は時刻と位置のペアの列で表現できる．このような表現手法を**点による表現**と呼ぶ．GPS により一定期間ごとに移動オブジェクトの位置を検出する場合は，このような表現をすることが自然である．

点による表現を用いる場合，移動オブジェクト o のある時刻 $t = \tau$ ($\tau = 0, 1, \ldots, T$) における位置が $[x_1, \ldots, x_d]$ という d 次元空間上の点で表される場合，$d + 1$ 次元空間の R-木を構築し，各時刻 t に対する o の情報を $[x_1, \ldots, x_d, t]$ という $d + 1$ 次元ベクトルと表現し，o のオブジェクト ID とペアで R-木に挿入すると索引が構築できる．これは 3D-R 木

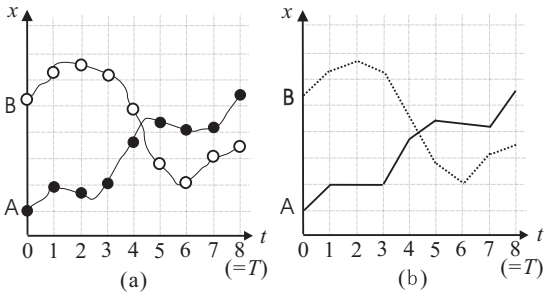

図 2: 移動軌跡の表現法：(a) 点による表現，(b) 線分による表現

の一般化である．

一方，図 2(b) は同じ経路を線分の集まりで近似したものであり，たとえば，オブジェクト B は，時刻 $t = 0$ から $t = 1$ まで x 軸上を一定方向に等速度運動をしたと近似されている．この表現方法を**線分による表現**と呼ぶ．

以上の実装手法により，点および線分による表現に対して時刻 t においてセル c に含まれている移動オブジェクトの ID を検索することが可能となる．

以下では点による表現を想定して説明を行うが，両方式で構築される R-木はリーフノードの構成が異なるものの，非リーフノードの構成はほぼ同様であるため，線分に基づく表現方式にも若干の拡張で対応可能である．

5 素朴な遷移確率推定アルゴリズム

本節では，用語の定義を述べた後，マルコフ遷移確率の定義を直接的に適用した素朴な遷移確率推定アルゴリズムを示す．

5.1 用語の定義

まず用語について整理する．$t = 0, 1, \ldots, T$ という集計の対象となる時刻のことを**サンプリング時刻**と呼ぶことにする．各サンプリング時刻の間の時間間隔（例：1 分，1 時間）を**サンプリング周期**と呼ぶ．

次に，セルに関する制約について述べる．セルの境界が矩形であることとセルの領域に交わりがないことを制約とするが，セルの分割は図 1 のようにサイズの異なるセルを含んでいてよいとする．セルへの分割は時空間データベースが対象としている空間全体をカバーしている必要はなく，ユーザが着目している領域に対してのみ行われていればよいとする．また，セルの分割は前もって定める必要はなく，ユーザの対話的な分析作業において動的に設定できるものとする．これにより，ユーザの興味がある領域や移動オブジェクトが密集して詳細情報を知りたい領域について，対話的に細かいセル分割をすることが可能となる．このように，ユーザの要求に応じて，遷移の時間間隔やセルの分割を任意に指定できる点は本研究の特徴である．

5.2 問題の定式化

ここで，時空間索引を用いて，式 (2) に示した n 次のマルコフ遷移確率の推定を行うことを考える．ただし，特定のセルの組合せ $c_0, \ldots, c_n$ に対する遷移確率の推定ではなく，以下のように問題を一般化する．

遷移確率の推定問題
$n + 1$ 個のセルの集合
$C_0 = \{c_{0,1}, \ldots, c_{0,|C_0|}\}, \ldots, C_n$
$= \{c_{n,1}, \ldots, c_{n,|C_n|}\}$ が与えられたとき，任意のセルの組合せ $(c_0, c_1, \ldots, c_n) \in C_0 \times \cdots \times C_n$ に対し，$\Pr(c_n|c_0, \ldots, c_{n-1})$ の値が未定義でなければその値を出力する． □

5.3 素朴なアルゴリズム

ここではまず，特定のセルの組合せ $c_0, \ldots, c_n$ について $\Pr(c_n|c_0, \ldots, c_{n-1})$ の推定を行うことを考える．この確率は，以下の 2 つの集合 S, Q を求めることが基本となる．

1. ある時刻 $t = \tau$ ($\tau = 0, 1, \ldots, T-n$) においてセル c_0 にいて，かつ，$t = \tau + 1$ においてセル c_1 にいて，...，かつ，$t = \tau + n - 1$ においてセル c_{n-1} にいたオブジェクトの集合 S

$$S = \{o \mid \exists\tau \in \{0, 1, \ldots T-n\}, o \in \mathrm{objs}(c_0, \tau) \wedge o \in \mathrm{objs}(c_1, \tau+1) \wedge \cdots \wedge o \in \mathrm{objs}(c_{n-1}, \tau+n-1)\} \quad (3)$$

2. S に含まれるオブジェクトのうち，$t = \tau + n$ においてセル c_n にいたオブジェクトの集合 Q

$$Q = \{o \mid o \in S \wedge o \in \mathrm{objs}(c_n, \tau+n)\} \quad (4)$$

このようなアイデアに基づく素朴なアルゴリズムを図 3 に示す．このアルゴリズムでは R-木の検索数が T に比例するという性質がある [14].

Procedure naive_estimation
Output: 値が未定義でない $\Pr(c_n|c_0,\ldots,c_{n-1})$ の推定値のリスト
1. **for each** $(c_0,\ldots,c_{n-1}) \in C_0 \times \cdots \times C_{n-1}$ **do**
2. $Scount := 0$; $Qcount[\,] := 0$
3. **for** $t := 0$ **to** $T - n$ **do**
4. **for** $i := 0$ **to** $n-1$ **do** $oids_i$ $:= \mathrm{range\_query}(c_i,\ t+i)$;
5. $Scount$ += $|\bigcap_{i=0}^{n-1} oids_i|$;
6. **for each** $c_n \in C_n$ **do**
7. $oids_n := \mathrm{range\_query}(c_n,\ t+n)$;
8. $Qcount[c_n]$ += $|\bigcap_{i=0}^{n} oids_n|$;
9. **end**
10. **end**
11. **if** $Scount = 0$ **then break**;
12. **for each** $c_n \in C_n$ **do**
13. **output**$(c_0,\ldots,c_n,\ Qcount[c_n]/Scount)$;
14. **end**
15. **end**

図 3: 素朴なアルゴリズム

6 効率的な遷移確率推定アルゴリズム

6.1 制約充足解の探索アルゴリズム

6.1.1 メインルーチン

$\Pr(c_n|c_0,\ldots,c_{n-1})$ の推定を以下の 3 ステップで行う．

1. 単位時間ごとに $c_0,\ldots,c_{n-1}$ に存在したオブジェクトの総数を R-木を用いてカウント
2. 単位時間ごとに $c_0,\ldots,c_n$ に存在したオブジェクトの総数を R-木を用いてカウント
3. ステップ 2 のカウント数 ÷ ステップ 1 のカウント数により $\Pr(c_n|c_0,\ldots,c_{n-1})$ を計算

メインルーチンのアルゴリズムを図 4 に示す．$Scount$, $Qcount$ は文字列キーのハッシュ変数であり，$c_0\#\cdots\#c_{n-1}$ は $c_0,\cdots,c_n$ をそれぞれ文字列化した後，それらを連結してできる文字列である．

後述のとおり，FC_cout 関数の呼出しでは，R-木のルートからリーフまで制約充足解を調べる探索処理が行われる．引数で指定したセルの集合に対応する領域が空間全体に比べ小さい場合は R-木の一部のノードにしかアクセスしないと考えられる．また，FC_count は 2 回しか呼び出しされないため，先の素朴な手法に比べ効率的な処理が達成できることになる．

Procedure FC_estimation(n, $root$, $level$, $(C_0,\ldots,C_n)$, max_dist)
Input: n: マルコフ遷移の次数
$root$: R-木のルートノード，$level$：R-木のレベル数
$(C_0,\ldots,C_n)$: セル集合のリスト
max_dist：単位時間に移動可能な最大距離
Output: 値が未定義でない $\Pr(c_n|c_0,\ldots,c_{n-1})$ の推定値のリスト
1. **for** $j := 0$ **to** n **do** $nodes[j] := root$;
2. $Scount := \mathrm{FC\_count}(n-1,\ level,\ nodes,$
3. $(C_0,\ldots,C_{n-1}),\ max\_dist)$;
4. $Qcount := \mathrm{FC\_count}(n,\ level,\ nodes,$
$(C_0,\ldots,C_n), max\_dist)$;
5. **foreach** $(c_0,\ldots,c_n) \in C_0 \times \cdots \times C_n$ **do**
6. **if** $Scount\{c_0\#\cdots\#c_{n-1}\} > 0$ **then**
7. **output**$(c_0,\ldots,c_n,$
8. $Qcount\{c_0\#\cdots\#c_n\}/Scount\{c_0\#\cdots\#c_{n-1}\})$;

図 4: メインルーチン

6.1.2 集計処理関数

次に，本提案手法の中心である，集計処理を行う関数 FC_count（図 5）について説明する．この関数は，[9] において提案された，R-木上で空間的な制約充足問題を解くためのアルゴリズムを拡張したものである．このアプローチでは，R-木上をルートからリーフ方向に制約を満たす解を探していく．その際，バックトラックや枝刈りをしながら充足解をもれなく探索する．

1〜7 行目では，C_i に関する制約条件の充足を検証している段階における C_j に関する解の候補の集合を保持している配列 $dom[i][j]$ に初期解集合を設定する．4 行目に現れる $\mathrm{sp\_overlap}(C_j,\ v)$ 関数は，セル集合 C_j に含まれるセルのいずれかと v が交わるかどうかを判定する述語であり，解候補の絞込みに役立つ．

while ループ内の説明に移る．まず，$n+1$ 要素の配列 $inst$ について説明しておく．これは，制約を充足するある解を探索している途中の状況で，部分的な探索解を保持するための配列である．i 番目の制約 C_i を処理している状況では，$inst[0],\ldots,inst[i-1]$ に部分解が入っている．10 行目で集合 $dom[i][i]$ が空であるかどうかを調べ，$i = 0$ であれば，$n+1$

```
Function FC_count(n, level, nodes, (C_0, ..., C_n), max_dist)
Input: n：遷移の次数,
       level：現在処理している R-木のレベル
       nodes：n + 1 要素のノード配列で nodes[j] は
              セル c_j に対応
       (C_0, ..., C_n)：セル集合のリスト
       max_dist：単位時間に移動可能な最大距離
Output: count：集計結果を入れたハッシュ表
1.  for j := 0 to n do  // 各制約に対する初期解集合を設定
2.    child_set := ∅;
3.    foreach v ∈ nodes[j].children do
4.      if sp_overlap(C_j, v) then
          //v は C_j と空間的な交わりを持つ
5.        child_set := child_set ∪ {v};
6.    dom[0][j] := child_set;  // 子ノードの集合を代入
7.  end
8.  i := 0;  // 現在着目している制約に対するインデックス
9.  while true do
10.   if dom[i][i].isempty then  // 空集合になった
11.     if i = 0 then return count;  // 手続きの終了
12.     else
13.       i−−;  continue;  // バックトラック
14.     end
15.   else
16.     new_val := get_next(dom[i][i]);
            // 次の要素を取り出す
17.     inst[i].value := new_val;
            // C_i の制約に対する解の候補とする
18.     if level ≥ 1 then
            // inst[i] の値がとり得る有効な時区間を設定
19.       inst[i].trange
            := t_overlap(new_val.trange, [i, T − n + i]);
20.     else inst[i].trange := new_val.trange;
21.   end
22.   if i = n then  // 現在の解候補で制約が充足された
23.     if level ≥ 1 then  // 非リーフノードの場合
24.       for k := 0 to n do refs[i] := inst[k].value;
25.       FC_count(n, level − 1, refs, {C_0, ..., C_n});
26.     else  // リーフノード：充足解に対するカウントを増やす
27.       count{cell(inst.[0].value)#
            ··· #cell(inst[n].value)}++;
28.     end
29.   else  // 制約充足の途中の場合
30.     if check_forward(i, n, level, dom, inst, (C_{i+1}, ..., C_n),
31.                      max_dist) then
32.       i++;  // 充足解が存在した．次は C_{i+1} の制約充足へ
33. end
```

図 5: Forward Checking に基づく集計関数

個の制約全体を満たす解の候補が存在しえなくなった（C_0 を満たす解候補がなくなれば，全体の制約を満たすことは不可能なため）ので関数を終了する．$i > 0$ の場合には i をデクリメントして while ループを続ける．これは，バックトラックを行って C_{i-1} の制約に関する別の解候補からあらためて調べなおすことを意味している．

$dom[i][i]$ が空でない場合，17 行目で $inst[i]$.value に新しい解の候補を代入する．18〜20 行目では，$inst[i]$ に代入された要素が i 番目の制約を満たす上でとり得る時区間を $inst[i]$.trange に設定している．19 行目の t_overlap(P, Q) は，時区間の集合 P, Q の交わりをとる関数である．例えば，t_overlap $([1,3],[4,8],[10,13],[2,6]) = [2,3],[4,6]$ となる．

22 行目で，今対象としている制約条件が n 番目かを調べ，まだ残りの制約条件がある場合 ($i < n$ の場合)，30〜32 行目が実行される．関数 Check_forward は次節で説明する．

6.1.3 Forward Checking の処理

関数 check_forward を図 6 に示す．この関数では，現在の i 番目の制約までの解候補 $inst[0], \ldots, inst[i]$ をもとに $i+1$ 番目から n 番目までの制約を満たす解候補があるかどうかを調べ，存在する場合には $dom[i+1][j]$ $(j = i+1, \ldots, n)$ に解候補集合を入れて true を返す．そうでない場合は false を返す．

こうすることで，解候補を調べる際に，いくつかの枝刈り条件に応じてあらかじめ解候補を絞り込むためのものである．この関数を用いることで，解の構成要素を含まない枝を探索するということがなくなるため，集計の効率化を図ることができる．このような処理を [9] では *forward checking* と呼んでいる．

枝刈り条件としては，(1) 対象としているオブジェクト ID と移動軌跡のオブジェクト ID が一致しているか，(2) 移動軌跡が対象とするセル内に含まれているか，(3) 移動軌跡が対象とするセル内に含まれているか，(4) オブジェクトが空間的に単位時間で移動できる距離であるか，という 4 つがある．

詳細については論文 [14] を参照していただきたい．

7 実験

本節では，移動オブジェクトのシミュレーションデータと R-木を用いて，素朴な遷移確率推定アルゴリズムと提案手法との処理時間の比較を行うためのフレームワークを示す．

7.1 実験用データ

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 [2]．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

Function check_forward($i, n, level, dom, inst, (C_{i+1}, \ldots, C_n), max\_dist$)
Input: i：現在処理している制約の番号，n：遷移の次数
$level$：現在処理している R-木のレベル
dom：解候補集合の配列，$(C_{i+1}, \ldots, C_n)$：セル集合
max_dist：単位時間に移動可能な最大距離
Output: 現在の $inst[0], \ldots, inst[i]$ に対し，
$inst[i+1], \ldots, inst[n]$ の候補があれば $true$，
そうでない場合 $false$
Note: 副作用として dom の値を変更

```
1. for j := i + 1 to n do  // 未チェックの各制約について
2.   dom[i+1][j] := dom[i][j];  // 解の候補集合を初期化
3.   foreach v ∈ dom[i+1][j] do  // 各候補について
4.     if (level = 0) and (inst[0].value.id ≠ v.id)
         then goto 15;
5.       // v は別のオブジェクトに対する移動軌跡であった
6.     vrange := t_overlap(v.trange, [j, T − n + j];
         // 有効な時区間を計算
7.     if vrange = ⊥ then goto 15;  // vrange が空であった
8.     if t_overlap(shift(vrange, −j), inst[0].trange = ⊥)
9.       then goto 15;    // 時間的な制約条件を満たさない
10.    if not sp_overlap(C_j, v) then goto 15;
11.      // v はセル集合 C_j の領域と空間的に重ならない
12.    if sp_dist(inst[i].value, v) > max_dist × (j − i)
         then goto 15;
13.    // inst[i].value から v まで j − i 時間内に移動不可能
14.    continue;
         // v は条件を満たした．次の v のチェックへ進む
15.    dom[i+1][j] := dom[i+1][j] − {v};
         // v を候補から削除
16.  end
17.  if dom[i+1][j] = ∅ then return false;
         // 充足解の候補がない
18. end
19. return true;
```

図 6: Forward Checking のための関数

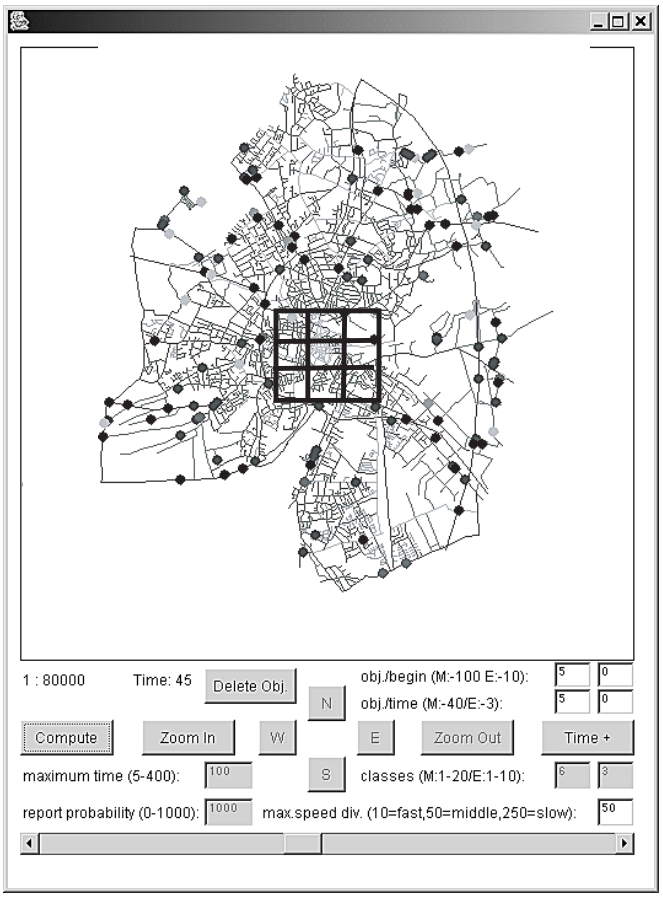

図 7: 移動オブジェクトデータの様子

おいて，パラメータを適宜設定して生成したものである．図 7 にこのシステムのインタフェース画面を示す．

移動データ生成では，初期移動オブジェクトの個数を 5 件とし，1 分ごとに 5 件ずつ移動オブジェクトが地図上にランダムに生成・配置されるものとした．各移動オブジェクトは目的地が設定されており，目的地に達した場合は地図上から削除される．また，移動オブジェクトが地図領域の外に出た場合にも，地図上から削除される．その結果，定常状態では平均して 100 件強の移動オブジェクトが地図上を移動することとなった．本実験では，このようにして生成される多数の移動オブジェクトの移動状況を，1 分刻みで $T = 1000$ 分の期間に対して生成した．この結果，(オブジェクト ID，時刻，x 座標，y 座標) という形式のデータが 124752 件生成された．これらのデータを 3 次元の R-木に登録した．

7.2 遷移確率の推定

ここでは，実際にマルコフ遷移確率を求める具体例について示す．地図領域を 20×20 のセル領域に分割し，あるセルから自分を含め近辺のセルに遷移する確率（1 次のマルコフ過程遷移確率）を調べた（図 7 の中央）．その部分を拡大した図を図 8 に示す．

素朴なアルゴリズムで遷移確率を求めた時，$\Pr(c_1|c_5), \Pr(c_2|c_5), \cdots, \Pr(c_9|c_5)$ の値はそれぞれ図に示すような値になる．このとき，確率を求めるために要した総ページアクセス数はのべ約 2500 万回であり，581.2 秒間で実行された．平均すると，一つの遷移確率を求めるために必要なページアクセス数は 2772994 回であり，64.57 秒で実行される．提案手法についても同様に評価を行う．

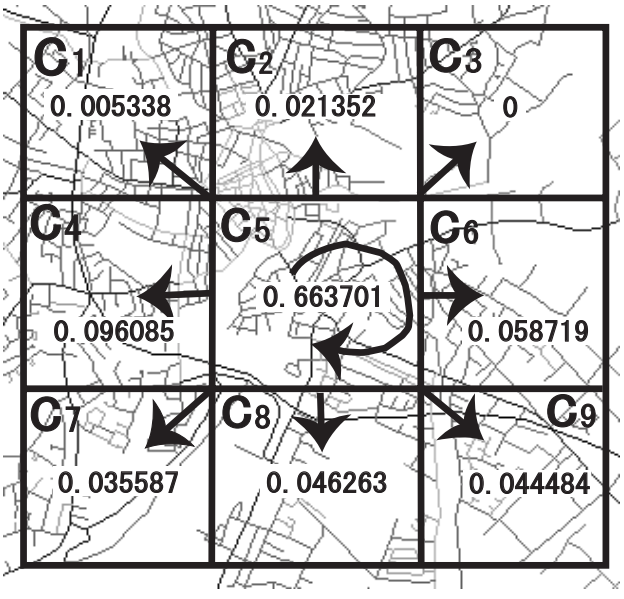

図 8: セル間のマルコフ過程遷移確率

8 まとめ

本稿では，時空間データベースの情報からマルコフ過程モデルに基づく移動統計量を推定するために提案されたアルゴリズムの実験について述べた．空間索引の内部情報を利用し，計数処理の効率化を図った点が特徴となっている．

今後の課題として，1）7 節で示したシミュレーションデータを用いたアルゴリズムの詳細な性能評価，2) データベース中のデータの分布に合わせたセルの適合的な分割，3) 非定常なマルコフ過程への拡張などが挙げられる．

謝辞

本研究の一部は，セコム科学技術振興財団研究奨励金，文部科学省科学研究費特定領域研究 (2)(15017207)，日本学術振興会科学研究費基盤研究 (B)(15300027)，若手研究 (B)(14780316) による．

参考文献

[1] S. Acharya, V. Poosala, and S. Ramaswamy, Selectivity Estimation in Spatial Databases, *Proc. of ACM SIGMOD*, 1999.

[2] T. Brinkhoff, A Framework for Generating Network Based Moving Objects, *GeoInfomatica*, 6(2), pp. 153-180, 2002.

[3] T. Brinkhoff, H.-P. Kriegel, and B. Seeger, Efficient Processing of Spatial Joins Using R-trees, *Proc. ACM SIGMOD*, 1993.

[4] Y. Choi and C. Chung, Selectivity Estimation for Spatio-Temporal Queries to Moving Objects, *Proc. of ACM SIGMOD*, pp. 440–451, 2002.

[5] V. Gaede and O. Günther, Multidimensional Access Methods, *ACM Computing Surveys*, 30(2), pp. 170–231, 1998.

[6] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching, *Proc. of ACM SIGMOD*, pp. 47–57, 1984.

[7] C.S. Jensen (ed.), Special Issue: Indexing of Moving Objects, *IEEE Data Engineering Bulletin*, 25(2), Jun. 2002.

[8] M.A. Nascimento, J.R.O. Silva, and Y. Theodoridis, Evaluation of Access Structures for Discretely Moving Points, *Proc. STDBM*, pp. 171–188, 1999.

[9] D. Papadias, N. Mamoulis, and Vasilis Delis, Algorithms for Querying by Spatial Structure, *Proc. of VLDB*, 1998.

[10] D. Pfoser, C.S. jensen, and Y. Theodoridis, Novel Approaches to the Indexing of Moving Object Trajectories, *Proc. of VLDB*, 2000.

[11] Y. Tao, J. Sun, and D. Papadias, Selectivity Estimation for Predictive Spatio-Temporal Queries, *Proc. of ICDE*, 2003 .

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

[13] 岸浩史，田名部淳，河野浩之，Σ-tree による時空間 OLAP 技術の交通データへの適用，データ工学ワークショップ（DEWS2002），2002 年.

[14] 石川佳治，塚本祐一，北川博之，索引付けされた移動軌跡データからの移動統計量の抽出法について，第 14 回データ工学ワークショップ（DEWS2003），2003 年.