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

石川　佳治† 　塚本　祐一†† 　北川　博之†

† 筑波大学電子・情報工学系　〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学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
E-mail: †{ishikawa,kitagawa}@is.tsukuba.ac.jp, ††yuichi@kde.is.tsukuba.ac.jp

あらまし　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を提案する．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を示す．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．
キーワード　時空間データベース，移動統計量，マルコフ連鎖モデル，移動分析，空間索引，制約充足問題

Extracting Mobility Statistics from Indexed Spatio-Temporal Datasets

Yoshiharu ISHIKAWA†, Yuichi TSUKAMOTO††, and Hiroyuki KITAGAWA†

† Institute of Information Sciences and Electronics, University of Tsukuba
†† Graduate School of Systems and Information Engineering, University of Tsukuba
Tennoudai 1–1–1, Tsukuba, Ibaraki, 305–8573 Japan
E-mail: †{ishikawa,kitagawa}@is.tsukuba.ac.jp, ††yuichi@kde.is.tsukuba.ac.jp

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 propose 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 proposed algorithm efficiently computes the specified Markov transition probabilities with the help of a spatial index R-tree. We reduce the statistics computation task to a kind of constraint satisfaction problem that uses a spatial index, and utilize internal representation of R-tree in an efficient manner.
Key words　spatio-temporal databases, mobility statistics, Markov chain model, mobility analysis, spatial indexes, constraint satisfaction problems

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 [6].

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．索引に関する研究がその例であり，R-木などの**空間索引** [4], [5] を用い，既存の空間軸だけでなく時間軸を含めた $d+1$ 次元で移動オブジェクトの各時点の位置を表現するアプローチや，時空間データベースの固有の要求に応じた時空間索引の研究が存在する [6].

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．データベースにおける問合せ処理の効率化には，問合せ最適化で用いられる選択率などの統計情報が重要であるが，蓄積された移動データに対し問合せの選択率を効率的に推定するための研究がすでに存在している [3], [10]. 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 [11]．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，本研究で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する．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 [11]．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を示す．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．

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．

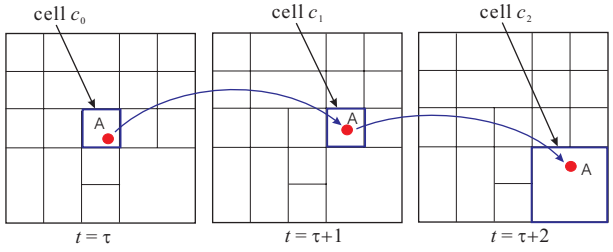

図 1　マルコフ過程モデルの概念

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．時空間データベースには多くの移動オブジェクトの移動軌跡が蓄積されているとする．ここでは履歴情報の開始時刻を $t = 0$，終了時刻を $t = T$ とする．$\Pr(c_1|c_0)$ を推定するには，各時点 $t = 0, 1, \ldots, T$ において各セルにどの移動オブジェクトが含まれているかが分かれば，

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

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．一方，図 1 のオブジェクト A のように，$t = \tau$ で c_0 内にいて $t = \tau + 1$ で c_1 内にいたオブジェクトが $t = \tau + 2$ で c_2 内に移動する確率は，2 次のマルコフ遷移確率 $\Pr(c_2|c_0, c_1)$ となる．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

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

という一般形で与えられる．$c_0, c_1, \ldots, c_n$ の中には同じセルが重複していても構わないものとする．

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．非定常的（non-stationary）な場合（たとえば，道路上の自動車の移動軌跡を扱う場合，時間帯によって移動状況が大幅に異なりうる）の扱いについては今後の課題とする．

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まとめて与えられたとき，$t = \tau + 1, 2, \ldots$ 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．なお，後述のように，提案手法では空間のセル分割は静的である必要はなく実行時に動的に決定してよく，また，マルコフ遷移の単位時間も遷移確率導出時に指定できる．このことから，本提案手法は対話的な移動分析に特に適した手法と考えられる．

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では，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．[3] 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．一方 [10] は，問合せ自体が時刻に依存して変化する場合に一般化して，選択率の推定法を提案している．

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．関連研究 [3], [10] 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，[12] 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．なお，マルコフ遷移確率を高速に導出できれば，空間占有率などの交通量に関する統計量のいくつかは確率的に導出可能である．また，セル内を通過する移動オブジェクト数については後述するアルゴリズム中で一般化して計算しているため，アルゴリズムの多少の修正で対応可能であると考える．

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，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た [8] がある．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．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その特徴である．詳しくは後述する．

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．[2] はそのような空間結合のための代表的なアルゴリズムであるが，[8] における空間索引を用いた制約充足問題の処理は，これを self-join とし，結合条件を空間的関連に一般化したものということができる．

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 [8] 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 [8] と共通するが，用いられる制約条件は [8] と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．本研究では，空間索引として一般的な R-木の利用を想定し，それに特殊な拡張を必要としない手法を特に検討する．

R-木による移動軌跡データの管理に関しては，すでにいくつかのアプローチが提案されている．[7] では，2 次元空間上を移動するオブジェクトの離散的な移動軌跡の索引付けの手法を比較している．比較された手法は，3D R-木（時間の次元を加えて 3 次元で軌跡を表す）や 2+3 R-木（現在の移動オブジェクトの位置を 2 次元の R-木で表し，軌跡の履歴を 3 次元で管理する）などである．また [9] では，移動軌跡の検索を目的とした R-木の拡張である STR-木を提案している．複数の線分により移動オブジェクトの軌跡を分割して表現する方式をとっており，木への挿入および木の分割のためのアルゴリズムを工夫して，空間的な近さだけでなく，同じオブジェクトの軌跡の一部であるかどうかも考慮して挿入・分割を行う点が特徴となっている．

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．

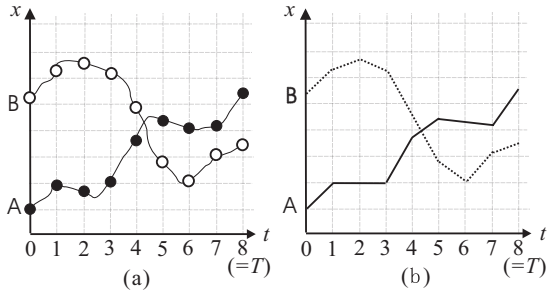

図 2　移動軌跡の表現法：(a) 点による表現，(b) 線分による表現

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る．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．点と線分による表現は相補的であり，状況に応じて適切な表現方式が選ばれる．

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 木の一般化である．一方，線分による表現の場合には，STR-木 [9] のように各線分をオブジェクト ID とともに R-木に登録することで索引が構成できる．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

以下では，R-木を用いたマルコフ統計量計算のための基本アプローチを示す．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．ここで，T の値は $T = 2^n$ と 2 の指数乗の値をとるものとする．これにより，サンプリングの粒度を粗くしたい場合にはサンプリング周期を 2^m 倍 $(1 \leqq m < n)$ ととればよい．たとえば，図 2 の例でサンプリング周期を 2 倍にした場合，$T = 0, 2, 4, \ldots, T$ という時刻が実際のサンプリング時間となる．移動データの移動経路の 1 分ごとの情報が蓄積されているが，実際の移動分析には 1 分という周期は短すぎるという場合，ユーザの分析要求に応じて 2^m 倍のサンプリング周期を用いるとよい．サンプリング周期を 2^m 倍した場合，サンプリング周期を単位時間として，あらためてサンプリング時刻を $t = 0, 1, \ldots, T$ と設定する．これにより，サンプリング周期を粗くした場合についても一般性を失わず議論が可能である．なお，線分による表現の場合は点による表現のような制約はなく，サンプリング周期を任意の値に設定することが可能である．ただし，あまりに短いサンプリング周期を用いた場合，移動軌跡の線分近似により，正確な確率値の見積もりができなくなる可能性がある．よって，移動軌跡の近似を考慮した適切なサンプリング周期の設定が必要である．

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． □

$\Pr(c_n|c_0, \ldots, c_{n-1})$ が未定義であるとは，$c_0, \ldots, c_{n-1}$ というセルを単位時間ごとに通過した移動オブジェクトが存在しない場合（つまり，式 (2) の分母が 0 になる場合）をいう．また，セル集合 $C_0, \ldots, C_n$ の間には交わりがあってよいものとする．

5.3 素朴なアルゴリズム

ここでまず，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

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

ここでは一般に，時空間データに対する索引が，関数

> $\mathrm{range\_query}(r, t)$：引数として矩形領域 r と時刻 t を受け取り，時刻 t において r 内に含まれる移動オブジェクト（点データ）のオブジェクト ID の集合を返す．

を支援可能であると想定する．4.2 節で述べた R-木による実装方式はこの関数を支援可能である．このとき，式 (2) の定義に基づく，素朴なマルコフ遷移確率の推定アルゴリズムは図 3 のようになる．2〜14 行目は 1 つのセルの組合せ $c_0, \ldots, c_{n-1}$ に対し，各 $c_n \in C_n$ について $\Pr(c_n|c_0, \ldots, c_{n-1})$ の値を出力するための処理であり，$|C_0| \times \cdots \times |C_{n-1}|$ 回実行される．3〜10 行目がその本体で $T - n + 1$ 回実行されるが，1 回の処理ごとに 3 行目で範囲検索が n 回，7 行目で $|C_n|$ 回呼び出しされる．$\bigcap_{i=0}^{n-1} oids_i$ は集合 S に含まれるオブジェクトの ID の集合であるので，$Scount$ には，あるセルの組合せ $c_0, \ldots, c_{n-1}$ に対し，それらのセルを各時点で通過したオブジェクトの総数が入ることになる．同様に，$Qcount[c_n]$ には，セルの組合せ $c_0, \ldots, c_{n-1}, c_n$ に対する，同様のカウント結果が入る．12〜14 行目は $Scount$ が 0 でない場合，それぞれのセル組合せ $c_0, \ldots, c_{n-1}, c_n$ に対し，$\Pr(c_n|c_0, \ldots, c_{n-1})$ を求める処理である．よって，素朴なアルゴリズムによる範囲検索呼出しの総数は $|C_0| \times \cdots \times |C_{n-1}| \times \{(T-n+1) + |C_n|\}$ となる．T が n, $|C_n|$ に比べ十分大きい場合，この値は $T \times |C_0| \times \cdots \times |C_{n-1}|$ と近似できるが，T を項に含んでいるため，数多くの範囲検索の呼出しが生じることがわかる．

次節では，この問題に対処するための，空間索引の内部構造を利用した効率的なアルゴリズムを示す．

6. 効率的な遷移確率推定アルゴリズム

6.1 マルコフ遷移確率から導かれる制約条件

本節では準備として必要な概念をいくつか導入し，次いでマ

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,...,c_{n-1}) ∈ C_0 × ··· × 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 := range_query(c_i, t + i);
5.      Scount += |⋂_{i=0}^{n-1} oids_i|;
6.      for each c_n ∈ C_n do
7.        oids_n := range_query(c_n, t + n);
8.        Qcount[c_n] += |⋂_{i=0}^{n} oids_n|;
9.      end
10.   end
11.   if Scount = 0 then break;
12.   for each c_n ∈ C_n do
13.     output(c_0,...,c_n, Qcount[c_n]/Scount);
14.   end
15. end
```

図 3　素朴なアルゴリズム

ルコフ遷移確率を計算する上で用いる制約条件を導く．制約条件の充足解を探索することが後述のアルゴリズムの目的となる．

P_i $(i=0,\ldots,n)$ を，セル集合 C_i の領域と移動オブジェクト o の移動軌跡が重なる時区間の集合とする．移動経路がセル集合 c の領域に複数回交わる可能性があるため，関数の結果は時区間の集合になることに注意する．なお，本稿では時区間の開始時刻，終了時刻は整数値であるものと想定する．次に，2 つの時区間 p, q が与えられたとき，その重なりの時区間を返す演算を $p \sqcap q$ で表すことにする．たとえば，$[2,6] \sqcap [3,9] = [3,6]$ である．また，空の時区間を $\perp$ で表す．時区間の集合 P_i と時区間 q の重なりについては，これを自然に拡張し，

$$P_i \sqcap q = \{p \sqcap q \mid p \in P_i, p \sqcap q \neq \perp\} \tag{5}$$

と定義する．たとえば，$\{[1,3],[4,8],[10,13]\} \sqcap [2,6] = \{[2,3],[4,6]\}$ となる．また，ある時刻 t に関して $t \in P_i$ が真であるとは，P_i 中に t を含む時区間が存在する場合をいう．

ここで，5.3 節で定義した式 (3) の集合 S および式 (4) の集合 Q について再び考える．P_i を時区間の集合，j を整数としたとき，$\mathrm{shift}(P_i, j)$ で P 中の時区間をそれぞれ j 時間だけシフトした時区間の集合を表す．たとえば，$\mathrm{shift}(\{[1,3],[5,8]\}, 1) = \{[2,4],[6,9]\}$ である．移動オブジェクトが S, Q に含まれるための条件について，以下のような命題が成立する．

［命題 1］　移動オブジェクト o が $o \in Q$ を満たすのは，

$$\forall i \in \{0,\ldots,n\},\ P_i \sqcap [i, T-n+i] \neq \emptyset \tag{6}$$

が成立し，かつ，ある整数 τ について

$$\forall i \in \{0,\ldots,n\},\ \tau + i \in P_i \sqcap [i, T-n+i] \tag{7}$$

である場合，すなわちある整数 τ について

$$\forall i \in \{0,\ldots,n\},\ P_i \sqcap [i, T-n+i] \neq \emptyset \\ \wedge\ \tau \in \mathrm{shift}(P_i, -i) \sqcap [0, T-n] \tag{8}$$

である場合である．一方，o が $o \in S$ を満たす条件は，式 (8) 内のすべての n を $n-1$ と置き換えることで得られる．　□

式 (6) の意味について説明する．$i=0$ の場合を例にとると，式 (6) の条件は $P_0 \sqcap [0, T-n] \neq \emptyset$ となる．P_0 は定義より，オブジェクト o がセル集合 C_0 に含まれているような時区間の集合である．一方，$[0, T-n]$ という時区間は，オブジェクト o がマルコフ過程の 0 番目の状態（セル集合 C_0）にあるという想定から導かれる制約である．たとえば，$P_0 = \{[T-n+1, T-n+1]\}$ の場合（$t = T-n+1$ においてのみ o が C_0 に含まれる場合），$t = T-n+1$ における o を始点とする長さ $n+1$ の遷移の列はそもそも存在しえない．存在しうるのは，$t = T-n$ の場合までで，その場合には，$t = T-n, t = T-n+1, \ldots, t = T$ という長さ $n+1$ の列が存在しうる．よって，$P_0 \sqcap [0, T-n] = \emptyset$ となった場合には，オブジェクト o はセル集合 C_0 に関して Q に含まれるための条件を満たさないことになる．

一方，C_1 に関する条件を調べる $i=1$ の場合には，式 (6) の条件は $P_1 \sqcap [1, T-n+1] \neq \emptyset$ となる．Q に含まれるオブジェクト o は C_0 に時刻 $t=0$ 以降に含まれるので，C_1 には $t=1$ 以降にしか含まれない．よって $t=0$ は排除される．時区間の最大値が $t = T-n+1$ となる理由は，$i=0$（セル集合 C_0）の場合と異なりセル集合 c_1 を対象としているため，長さ n の列が存在しうる最大の時区間を考えることによる．

式 (7) は，セル $c_0,\ldots,c_n$ に $t=\tau, t=\tau+1,\ldots,t=\tau+n$ という時刻ごとに o が含まれていること（すなわち $o \in Q$ であること）を検証するための条件である．これは，各 c_i $(i=0,\ldots,n)$ に対する時区間 $P_i \sqcap [i, T-n+i]$ に時刻 $t = \tau + i$ が含まれるようなある整数 τ が選べることを意味している．

6.2　制約充足解の探索アルゴリズム

6.2.1　メインルーチン

制約充足のアルゴリズムのメインルーチンを図 4 に示す．1 行目では，$n+1$ 要素のノードの配列 $nodes$ にルートノード $root$ への参照をそれぞれ代入している．この配列の役割については後述する．2〜4 行目で呼ばれる FC_count という関数は，本提案手法における主な処理を担当する．2〜3 行目では，$C_0,\ldots,C_{n-1}$ というセル集合を引数と，$n-1$ 次の移動オブジェクトの遷移を集計している．ただし，遷移の 0 番目では C_0 に含まれるいずれかのセル内，1 番目では C_1 に含まれるいずれかのセル内，…，$n-1$ で番目は C_{n-1} に含まれるいずれかのセル内であることを満たす解のみが集計される．FC_count の結果は連想配列 $Scount$ として返される．$Scount$ に対しては，セル番号を連結した文字列をキーとして $Scount\{c_0\#\cdots\#c_{n-1}\}$ とアクセスすることで，$c_0,\ldots,c_{n-1}$ と順に遷移した移動軌跡の総数を得ることができるものとする．4 行目は同様に，n 次の移動オブジェクトの遷移を集計している．これらの値を用いて，最終的に 5〜8 行目で遷移確率の出力を行っている．FC_count の引数として与えられる max_dist の役割については後述する．

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 := FC_count(n − 1, level, nodes, (C_0, ..., C_{n−1}),
3.                     max_dist);
4.  Qcount := FC_count(n, level, nodes, (C_0, ..., C_n), max_dist);
5.  foreach (c_0, ..., c_n) ∈ C_0 × ··· × C_n do
6.    if Scount{c_0# ··· #c_{n−1}} > 0 then
7.      output(c_0, ..., c_n,
8.             Qcount{c_0# ··· #c_n}/Scount{c_0# ··· #c_{n−1}});
```

図 4　メインルーチン

6.2.2　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，[8] 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，引数として与えられた $n+1$ 要素のノード配列 $nodes$ の各要素 $nodes[j]$ について，$nodes[j]$ が非リーフノードの場合 $nodes[j]$ の子のノードの集合を，$nodes[j]$ がリーフノードの場合 $nodes[j]$ 中に含まれる移動軌跡のエントリの集合を，$dom[0][j]$ に代入する．dom は $(n+1) \times (n+1)$ の集合を要素とする配列であり，$dom[i][j]$ 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．なお，ここでは指定された $C_0, \ldots, C_{n+1}$ に関する $n+1$ 個の制約条件をこの順に処理していくと想定している．最初に $dom[0][j]$ $(j = 0, \ldots, n)$ を初期化することは，C_0 に関する制約を処理している時点での，$n+1$ 個の制約に対する解の初期候補を与えていることに相当する．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る．C_j と空間的に交わりを持たない v は探索の候補に含まれないことになる．以降の処理では，これらの解の初期候補を R-木の内部情報を用いて絞り込んでいく．なお，メインルーチンから FC_count が最初に呼ばれたときには，$nodes[j] = root$ $(j = 0, \ldots, n)$ と初期化されているため，各 $dom[0][j]$ にはルートノードの 1 レベル下の子ノードの集合が初期解集合として代入される．

8 行目で i を 0 に初期化している．i は何番目の制約条件の検証を行っているかを保持する変数であり，9〜33 行目の while ループ内でこの値を増減することで充足解の探索を制御する．

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．while ループ中では，まず $inst[i]$ に i

Function FC_count(n, $level$, $nodes$, $(C_0, \ldots, C_n)$, 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
$(C_0, \ldots, C_n)$：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: $count$：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 := 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  // リーフノードの場合：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

番目の制約 C_i を満たす解を追加しようとする．10 行目で集合 $dom[i][i]$ が空であるかどうかを調べているが，それが空である場合，C_i に関する制約を満たす解の候補がなくなったことを表す．そのとき，もし $i = 0$ であれば，$n+1$ 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．new_value は，$level \geqq 1$ の場合（非リーフノードのレベルを現在処理している場合）には R-木のノードであり，$level = 0$ の場合（リーフノードのレベルを処理し

ている場合）にはリーフノード中の移動軌跡データのエントリ（$d+1$ 次元の点オブジェクト）である．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．$level \geqq 1$ の場合，new_value.trange は，R-木のノード new_value の最小包囲矩形（minimum bounding rectangle, MBR）の時間軸における範囲を表している．19 行目では，これと式 (8) で示した $[i, T-n+i]$ という時区間の交わりの区間をとり $inst[i]$.trange に代入している．これは，ノード new_value の下に位置するリーフノードに含まれる移動軌跡のエントリのうち，この時区間内に入るもののみが解の条件を満たすことを表す．$level = 0$ の場合は，20 行目で単に new_value（この場合は点オブジェクト）の時間軸の値を $inst[i]$.trange に代入する．点オブジェクトの場合は時間軸の値が時区間ではなく時刻であるため，$inst[i]$.trange の値は $[5, 5]$ のような時間幅が 0 の時区間となる．

22 行目では，いま対象としている制約条件が n 番目かどうかを調べている．まず $i = n$ のときには 23〜28 行目が実行される．現在非リーフノードのレベルを処理している場合（$level \geqq 1$）には，現在 $inst[0]$.value, ..., $inst[n]$.value に束縛されているノードを引数として，$level$ の値を 1 つ減らして FC_count を再帰呼出しする．リーフノードのレベルを処理している場合（$level = 0$）には，見つかった解のカウントを増やすため，連想配列 $count$ をインクリメントする．関数 cell は，与えられた点オブジェクトが属するセルの番号を返す関数であるとする．

$i < n$ の場合，すなわち，まだ残りの制約条件がある場合には 30〜32 行目が実行される．30 行目で呼ばれる関数 check_forward は，現在までの部分解 $inst[0], \ldots, inst[i]$ に対し，それ以降の制約に対する解が存在するかどうかをあらかじめチェックするための関数である．このような処理は *forward checking* と呼ばれ，制約充足問題の解探索で用いられる効率化のアプローチの 1 つである．check_forward の値が真であれば i をインクリメントし，$i+1$ 番目の制約に対する処理に移る．check_forward の値が負の場合，現在の部分解 $inst[0], \ldots, inst[i]$ については，それ以上処理を進めても充足解はないことになる．この場合は 10 行目に戻り次の $inst[i]$ の候補について同様の処理を行う．

6.2.3 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．

まず 1〜18 行目のループでは，j 番目 $(j = i+1, \ldots, n)$ の制約について解の候補集合 $dom[i+1][j]$ の値を求める．ただし，$dom[i+1][j] = \emptyset$ となった場合には，現在の $inst[0], \ldots, inst[i]$ に基づく制約充足解が存在しないため，17 行目で false を返し関数を終了する．2 行目では，まず $dom[i+1][j]$ の候補集合値を初期化し，3〜16 行目のループの中で，そこから充足解の構成要素となりえない要素を 1 つ 1 つ取り除いていく．

4 行目では，現在リーフノードのレベルを処理している場合を扱っている．この場合，$inst[0]$.value, ..., $inst[i]$.value には，ある移動オブジェクトの長さ $i+1$ の移動軌跡が保持されている．現在対象としている移動オブジェクトの移動軌跡データ（点オブジェクト）v のオブジェクト ID とこれまでの移動軌跡のオブジェクト ID が一致しない場合には，同一の移動オブジェクトの軌跡とはならないため，候補から排除する．

6〜7 行目では，v（$level \geqq 1$ なら R-木のノード，$level = 0$ なら移動軌跡の点データ）に対する有効な時区間を計算する．これが空の時区間となる場合，v は解の候補とはなりえない．8 行目では式 (8) における整数値 τ が実際に存在しうるかを調る．10 行目では，セル集合 C_j と v が空間的に交わるかどうかを調べる．最後に 12 行目では，i 番目の制約に対する候補である $inst[i]$.value（$level \geqq 1$ なら R-木のノード，$level = 0$ なら点オブジェクト）と v の空間的な距離 sp_dist で求める．空間的な距離は，R-木のノード同士については MBR の最近点間の距離を用いる．この値が与えられた $max\_dist \times (j-i)$ より大きいとき，$inst[i]$.value の領域（もしくは点）から v の領域（もしくは点）まで $j-i$ 時間では移動できないことになる．これにより，無駄な解候補を削減できる．以上のチェックをすべてパスした場合にのみ v は $dom[i+1][j]$ に残ることになる．

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 := 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 shift(vrange, −j) ⊓ inst[0].trange = ⊥ then goto 15;
9.       // 式 (8) を満たす τ は存在しえない
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 までは j − i 時間内に移動できない
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

7. 問合せ処理の例

図 2 のデータに対し R-木を構築した例を図 7 に示す．1〜6 はリーフレベルの MBR であり，a, b, c はそれぞれノード 1 と 2，3 と 4，5 と 6 の親ノードの MBR である．ノード a, b, c の

親ノードはルートノード root である．図で示すように，セル c_1 と c_2 の領域は，それぞれ [1, 3) と [3, 6) であるものとする．

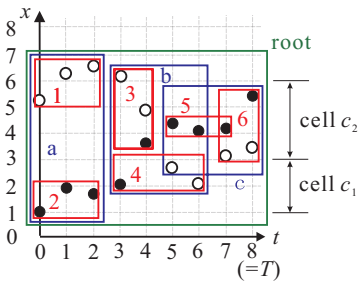

図 7　R-木の構成例

ここで，$C_0 = \{c_1\}, C_1 = \{c_1, c_2\}, C_2 = \{c_2\}$ について FC_estimation(2, root, 2, (C_0, C_1, C_2), 2.5) を実行することについて考える．すなわち，推定するマルコフ過程の次数を $n = 2$，R-木のレベル数を $level = 2$，移動オブジェクトの単位時間あたりの最大移動距離を $max\_dist = 2.5$ とする．

ここで，特に FC_estimation（図 4）の 4 行目で FC_count を呼び出した場合の処理を述べる．FC_count（図 5）の 1〜7 行目により，$dom[0][0] = dom[0][1] = dom[0][2] = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ となる．$i = 0$ として while ループに入り，16〜20 行目で $inst[0].\mathrm{value} = \mathrm{a}$, $inst[0].\mathrm{trange} = [0, 2]$, $dom[0][0] = \{\mathrm{b}, \mathrm{c}\}$ となる．ここで 30 行目の check_forward を呼び出す．check_forward（図 6）では，まず $j = i + 1 = 1$ について処理する．2 行目で $dom[1][1] = dom[0][1] = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ と初期化する．$v = \mathrm{a}$ および $v = \mathrm{b}$ はその後の条件をすべて満たすため，$dom[1][1]$ から排除されない．しかし，$v = \mathrm{c}$ の場合，6 行目で $vrange = [5, 7]$ となり，8 行目で $\mathrm{shift}([5, 7], -1) \sqcap [0, 2) = \bot$ となり排除され，$dom[1][1] = \{\mathrm{a}, \mathrm{b}\}$ となる．$j = 2$ の場合も同様して，$dom[1][2] = \{\mathrm{a}, \mathrm{b}\}$ となる．すなわち，$inset[0].\mathrm{value} = \mathrm{a}$ 内に 0 番目の制約を満たす移動軌跡データがあるとしたとき，1, 2 番目の制約を満たす移動軌跡データはノード c の下にはないことになる．最後に check_forward は true を返す．

図 5 の FC_count に戻ると，check_forward が true であったので，32 行目で i を増やし while ループを繰り返す．16〜20 行目で今度は $inst[1].\mathrm{value} = \mathrm{a}$, $inst[1].\mathrm{trange} = [1, 2]$, $dom[1][1] = \{\mathrm{b}\}$ となる．再び check_forward を呼び出すと true が返り，$dom[2][2] = \{\mathrm{a}, \mathrm{b}\}$ となる．ここで再び FC_count に戻り，i を増やす．次の while ループにより，今度は 25 行目で $\mathrm{FC\_count}(2, 1, refs, \{C_0, C_1, C_2\})$ を再帰呼び出しする．ただし，$refs[0] = \mathrm{a}, refs[1] = \mathrm{a}, refs[2] = \mathrm{a}$ である．

FC_count の再帰呼出しでは，まず C_0, C_1, C_2 の制約を考慮し，$dom[0][0] = \{2\}, dom[0][1] = \{1, 2\}, dom[0][2] = \{1\}$ とおいて同様の処理を繰り返す．まず $inst[0].\mathrm{value} = 2$ して次に check_forward を呼び出すと結果は true となり，$dom[1][1] = \{2\}, dom[1][2] = \{1\}$ が得られる．$dom[1][1]$ からノード 1 が排除される理由は，$inst[0].\mathrm{value}$ のノード 2 から単位時間内にノード 1 内に移動できない（$sp\_dist(2, 1) > max\_dist$）ことによる．この結果，今度は $\mathrm{FC\_count}(2, 0, refs, \{C_0, C_1, C_2\})$ を再帰呼出しする．ただし，$refs[0] = 2, refs[1] = 2, refs[2] = 1$ である．詳細は省略するが，この場合は充足解は得られない（$t = \tau$ でセル 2 にいて，$t = \tau + 1$ でセル 2 にいて，$t = \tau + 2$ でセル 1 にいたオブジェクトは存在しない）．最終的にはバックトラックとなり，今度はレベル 1 に戻り，$inst[0].\mathrm{value} = \mathrm{a}, inst[1].\mathrm{value} = \mathrm{a}, inst[2].\mathrm{value} = \mathrm{b}$ の場合を探索して失敗しバックトラックする．そして $inst[0].\mathrm{value} = \mathrm{a}, inst[1].\mathrm{value} = \mathrm{b}, inst[2].\mathrm{value} = \mathrm{a}$ で再び失敗し，今度は $inst[0].\mathrm{value} = \mathrm{a}, inst[1].\mathrm{value} = \mathrm{b}, inst[2].\mathrm{value} = \mathrm{b}$ を行う．この場合には充足解 $inst[0].\mathrm{value} = (id = \mathrm{A}, t = 2, x = 1.8)$, $inst[1].\mathrm{value} = (id = \mathrm{A}, t = 3, x = 2.2), inst[2].\mathrm{value} = (id = \mathrm{A}, t = 4, x = 3.6)$ がが 1 つ得られる．この結果，$\mathrm{count}(c_1 \# c_1 \# c_2)$ が増やされる．以上の処理を繰り返すことで，制約を満たす解をもれなく集計する．

なお，この場合の FC_estimation の結果は，$\Pr(c_2|c_1, c_1) = 2/4 = 0.5, \Pr(c_2|c_1, c_2) = 2/2 = 1.0$ となる．

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のアルゴリズムを示した．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) 実験に基づく評価，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

読者の方々からは貴重なコメントをいただきました．ここに感謝いたします．本研究の一部は，日本学術振興会科学研究費基盤研究 (B)(12480067)，若手研究 (B)(14780316)，および文部科学省科学研究費特定領域研究 (14019009) による．

文　　献

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

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

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

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

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

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

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

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

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

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

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

[12] 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．