文字列索引によるネットワーク制約下の車両軌跡の索引化

小出　智士† 　田所　幸浩† 　吉村　貴克†

† 株式会社 豊田中央研究所　〒 480–1192 愛知県長久手市横道

E-mail: †{koide,tadokoro,yoshimura}@mosk.tytlabs.co.jp

あらまし　GPS ロガーなどを通じて大量に収集される車両の軌跡データを，走行経路および走行時間帯を考慮して適切に管理することは，車両から収集されるデータを利用したアプリケーションのための基盤として重要である．本論文では蓄積された車両軌跡データに対する，完全経路クエリと呼ばれる時空間検索を行うためのデータ構造を提案する．提案手法は車両が基本的には道路上のみを走行することに着目し，軌跡をグラフ辺を文字とする文字列として扱うことで文字列パターンマッチの問題に帰着させる．さらに車両軌跡データに含まれる時間情報を取り扱うために逆接尾辞配列を用いて，時刻情報を格納する B^+ 木 と文字列索引である FM-index を統合する方法を提案する．提案手法の有効性を示すために実データを用いた比較評価を行い，特に経路クエリ長が大きな場合に従来法と比較して高速に解を得られることを確認した．

キーワード　時空間索引，ネットワーク制約軌跡，完全経路クエリ，逆接尾辞配列，FM-index

1. はじめに

近年，自動車などの移動体から収集されるデータの量および種類が爆発的に増大してきており，それらのデータを活用したアプリケーション開発への期待が高まっている．そのような応用の例としては例えば，車両軌跡データを用いてタクシードライバーの経路選択行動を学習するようなスマートナビゲーション [19] や，低コストのセンサ (車載カメラなど) のデータを統合することで地図を作成する技術 [7] などが挙げられる．このような技術の開発においては収集・蓄積された大量の車両軌跡データに対して，欲しいデータを必要に応じて高速に検索し，取り出すことができるよう適切に管理されていることが重要である．車両軌跡データの検索に特有の問題として，時空間コンテキスト (車両の走行経路および走行時間帯) を考慮した検索が必要になるという点がある．

空間索引の多くはユークリッド空間内の点として表現される位置情報に対して構築される．車両軌跡データにおいても通常，位置情報は GPS ロガーなどから得られる緯度経度の座標列として収集される．しかしながら，自動車は基本的には道路ネットワーク上のみを走行することに着目すれば，位置情報は (道路ネットワークを有向グラフ $G = (V, E)$ と考えた場合の) グラフ辺の列として表現することができる．このような軌跡の表現は**ネットワーク制約軌跡**と呼ばれる．図 1 に道路ネットワーク上を移動するネットワーク制約軌跡の例を示す．例えば，軌跡 $\mathcal{T}_0$ は $e_{ab} \to e_{bc} \to e_{ce} \to e_{ef}$ のように走行している．車両の位置を緯度経度ではなく，道路リンクの列として取り扱うことは多くのアプリケーション (ナビゲーションなど) にとって最も自然なものであると考えられる．以上のことから，我々はネットワーク制約軌跡に対する索引化・検索の問題について考える．

本論文では時空間コンテキストを考慮可能なクエリの中で，最もシンプルである **完全経路クエリ** (Strict Path Query; SPQ)

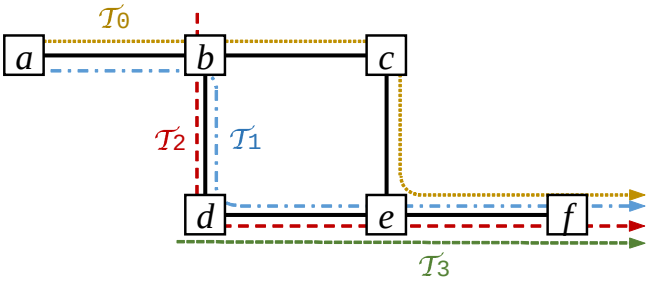

図 1　6 つの道路リンク $E = \{e_{ab}, e_{bc}, e_{bd}, e_{ce}, e_{de}, e_{ef}\}$ からなる道路ネットワーク上を移動する 4 つの軌跡 ($\mathcal{T}_0 \sim \mathcal{T}_3$) の例

に着目する．これは時刻 s から t の間にクエリ経路パターン P で走行した車両 (の ID) をすべて列挙する，という問い合わせである (厳密な定義については後述する)．ただし，クエリ経路パターン P は辺集合 E 上の記号列である．後に 3.1 節で詳しく見るように，この完全経路クエリを素朴な方法で解こうとすると，経路クエリ長 $|P|$ に比例したディスクアクセス (B^+ 木の探索) およびデータの結合 (join) が発生し，特に $|P|$ が大きな場合について高速に解くことができない，という問題がある．

ネットワーク制約軌跡は E の要素を文字とする (時刻付きの) 文字列であるとみなすことができる．従って軌跡に対する検索は，ある種の文書検索の問題とみなすことができる．我々はネットワーク制約軌跡に対する索引を全文索引を利用して構成することを試みる．しかしながら全文索引は記号列 (道路リンク列) の検索を実現するものであり，それに付随する時間的な検索条件を考慮できない．そこで我々の提案手法では全文索引として FM-index を用い，時刻などの情報は B^+ 木に格納する，というハイブリッド構造を採用する (図 2 に提案手法の概要を示す)．FM-index は与えられたパターン P に対応する

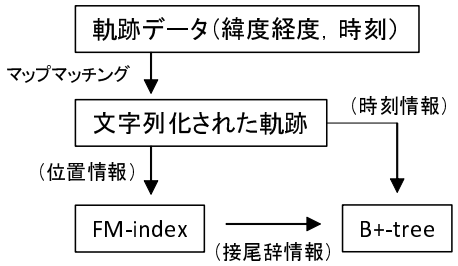

図 2　提案する索引構造の概要

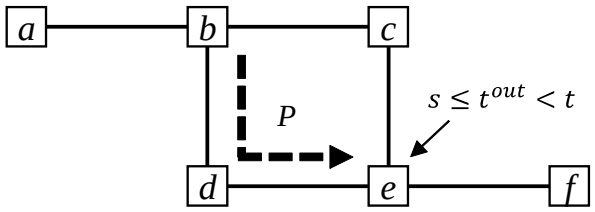

図 3　図 1 のネットワークにおける $P = e_{bd} \rightarrow e_{de}$ の完全経路クエリ．P の最終ノードでの時刻について制約する．

接尾辞配列上の範囲を高速に返すことができるが，時間に関する情報を保持できないため，時刻を含めたクエリを実行できない．そこで逆接尾辞配列を用いて，空間パターンを索引化する FM-index と時刻などを索引化する B^+ 木を対応付けることで完全経路クエリを処理する方法を提案する．この結果，ディスクアクセスのコストが経路クエリの長さ $|P|$ に依存しない方法を実現することができ，高速化を達成することができる．

本論文の構成は以下のとおりである．2 章では軌跡の表現および問題設定を述べる．3 章ではネットワーク制約軌跡に関する索引，および文字列検索に関する関連研究を述べる．4 章では提案手法を説明する．5 章では実際のプローブカーデータを用いた評価実験の結果を示す．6 章および 7 章では手法および結果に関する議論および結論を述べる．なお，本研究についての詳細は [9] を参照されたい．

2. 問 題 設 定

2.1 軌跡表現の定義

本論文では道路ネットワークを有向グラフ $G = (V, E)$ として表す．ここでは道路ネットワークは時間とともに変化しないと仮定する．道路ネットワーク上の車両の位置情報は 4 要素からなるタプル (d, e, t^{in}, t^{out}) として表現されるものとする．ただし，d は軌跡 ID, $e \in E$ は通過した道路リンク，t^{in}, t^{out} は軌跡 d が道路リンク e に入った時刻および出た時刻である．ネットワーク制約軌跡は上記の位置情報の列 $\mathcal{T}_d := \{(d, e_i, t_i^{in}, t_i^{out})\}_{i=0}^{n_d-1}$ として定義される．軌跡 $\mathcal{T}_d$ の道路リンク列のみを取り出すために，$\mathcal{T}_d.e := e_0 e_1 \cdots e_{n_d-1}$ のような表記を用いる．また，車両はネットワーク上を連続的に動くものとする．従って e_i と e_{i+1} は隣接した道路リンクであるものとし，また $t_i^{out} = t_{i+1}^{in}$ である．

先に述べたように本論文では軌跡を文字列として扱う．従って道路リンク集合 E には辞書順が定義されているとする．実際には，この順序の設定には特に制約はなく，任意の順序を用いてよい．

また，配列 A に対して，i 番目の要素を $A[i]$ と表し，$A[i, j)$ を i 番目から $j-1$ 番目までの部分配列とする．従って，上で述べたネットワーク制約軌跡に対しては $\mathcal{T}_d.e[i, j) = e_i\ e_{i+1} \cdots e_{j-1}$ となる．また本論文を通して，配列などの添字は 0 から始まるものとする．

2.2 完全経路クエリ

ここでは完全経路クエリの厳密な定式化を行う．1 章で述べたように，完全経路クエリは定性的には時刻 s から t の間にクエリ経路パターン P で走行した車両 (の ID) をすべて列挙する，という問い合わせである．

定義 1. D 個の軌跡の集合 $\mathbb{T} = \{\mathcal{T}_d \mid 0 \leqq d < D\}$ を考える．問い合わせ経路 P は長さ m の E 上の配列とする: $P := p_0\ p_1 \cdots p_{m-1}$．本論文では完全経路クエリ (Strict Path Query) を以下のように定義する：

$$SPQ(P, s, t) := \{d \mid \exists i, j\ :\ \mathcal{T}_d.e[i, j) = P \wedge s \leqq \mathcal{T}_d.t^{out}[j-1] < t\}. \quad (1)$$

すなわち，完全経路クエリ $SPQ(P, s, t)$ は部分的に P というパターンで走行し，かつ P の最後のリンク p_{m-1} を通過し終えた時刻が s と t の間に含まれるような $\mathbb{T}$ の軌跡を抽出するようなクエリである．

再び図 1 の例に戻って説明する．クエリ経路パターン $P = e_{bd} \rightarrow e_{de}$ および $[s, t) = [18, 25)$ の場合，$SPQ(P, s, t)$ は道路リンク e_{de} を出た時刻，すなわち交差点ノード e での時刻が $[18, 25)$ の間にある軌跡に対応する (図 3)．なお，時刻の制約の仕方としては P 全体が $[s, t)$ に含まれる，すなわち P の最終ノードのみではなく，最初のノード (先の例ではノード b) の時刻も制約する，というクエリも考えられる．実際，関連研究 [10] ではそのようなクエリのみを考えている．4.6 節で示すように，我々の方法は簡単な拡張によってそのようなクエリにも対応できることを強調しておく (注1)．

3. 関 連 研 究

3.1 ネットワーク制約軌跡の索引付け

ネットワーク制約軌跡に対する索引構造の研究としては FNR-tree [5] がある．これは道路リンクごとに車両が該当の道路リンク上に存在した時間区間を R 木で管理するという索引構造である．ここでは時空間矩形クエリを対象にしており，完全経路ク

(注1)：ただし両者の結果は非常に似通っている場合がほとんどであるため，厳密に時間幅を限定したい場合を除いてはあまりメリットはなく，より高速に解くことができる我々の定義 (1) で十分である．

エリのような経路に関するクエリは明示的には考慮されなかった．Vieira らによって提案された索引構造 [17] は FNR-tree と類似の構造を用いるが，対象とするクエリの範囲が拡張されている．具体的には，フレキシブルパターンクエリ (FPQ) と呼ばれる完全経路クエリを含むクエリを定義し，FPQ に対する IJP (Index-Join Pattern) アルゴリズムが提案された．しかし，IJP アルゴリズムは文書検索に対する転置インデックスと類似のアルゴリズムであり，クエリ長 $|P|$ に比例するディスクアクセスおよびマージが発生し，完全経路クエリに対しては高速であるとは言えない (注2)．

Krogh らは完全経路クエリを初めて明示的に定義し，その索引構造を提案した [10]．この論文では大別して二種類のアルゴリズムが提示されている．一つは完全経路クエリに対して厳密な結果を返す，IJP アルゴリズムに類似のアルゴリズムである．従って，クエリ長 $|P|$ に比例するディスクアクセスおよびマージ操作が発生する．もう一つは近似アルゴリズムであり，ディスクアクセスが $|P|$ に依存しない検索を実現する．しかし，この方法は軌跡に対するハッシュ関数を定義する方法であり，検索結果にはハッシュの衝突に起因する誤検出が含まれる可能性がある．本研究では Krogh らの手法では実現できない，ディスクアクセスが $|P|$ に非依存，かつ厳密な結果を返すことができる手法を提案する．

3.2 文字列索引

文字列におけるパタンマッチングは多くの既存研究がある．ここでは本研究で用いる接尾辞配列および FM-index に関連する話題について述べる．

接尾辞配列は文字列処理にとって重要なデータ構造であり，文字列 T の接尾辞を辞書式順序で並び替えたときの接尾辞のソート前の順序 (添字) として定義される．接尾辞を並び替えたことで，以下のよく知られた重要な性質が得られる：任意の部分文字列パターン P が T に出現する位置は接尾辞配列 SA 上の連続した範囲 $SA[sp, ep)$ として与えられる．

FM-index は Ferragina らによって提案された全文索引であり，圧縮された文字列上での高速なパタンマッチを可能にする [4]．これは $T_{bwt}[j] = T[SA[j] - 1]$ で定義される Burrows-Wheeler 変換 [2] をウェーブレット木 [6] (注3) に格納することで得られるデータ構造である．この索引構造を用いると与えられたパターン P に対して，上で述べた文字列 T に出現する P の対応する接尾辞配列上の範囲を文字列長 $|T|$ に依存しない $O(|P| \log \sigma)$ の時間で求めることができる．ここで σ はアルファベット集合のサイズであり，我々の問題設定においては道路リンク集合のサイズ $|E|$ となる．また空間複雑性は $|T| \log \sigma + o(|T|)$ ビットであり，元のテキストと同程度のサイズで元のテキストと索引の両方を格納できる．このサイズはデータ圧縮することで，同等の検索機能を持ったまま，さらに小さくすることができることが知られている [8], [12]．接尾辞配列および FM-index を含む文字列索引の話題については，例えば [20] が詳しい．

3.3 XML ドキュメントに対するパス検索

経路およびそれに付随する属性を検索する関連技術として XML ドキュメントに対するパス検索の技術が挙げられる．例えば Yoshikawa らによる XRel [18] では RDB を用いて XML 文書を索引化し，パス検索を実現している．その構造は非常に柔軟であり，様々な属性とパスを同時に取り扱うことができる．また原理的には (XML パスを車両経路と読み替えることで) 完全経路クエリに対しても適用可能である．しかしながら XRel では文書内に存在するパスの全ての接頭辞を格納するため，極めて多種類の接頭辞が出現し得る車両経路データに対して適用することはあまり効率的でないと考えられる．

一方で XML 文書のような構造化文書に対するデータ構造は (例えば加速度や画像など，時間以外の要素を含む) 本研究で対象とするデータよりも複雑な属性で構成される現実の車両データの格納や検索などを考慮する場合に役立つ可能性があることを強調しておく．

4. 提 案 手 法

4.1 前 処 理

通常，位置情報データは GPS ロガーなどから得られる緯度経度座標として与えられるため，これを適切に前処理し，ネットワーク制約軌跡表現を得る必要がある．本研究では，隠れマルコフモデルに基づくマップマッチング手法 [14] を用いて緯度経度を道路リンクの列に変換した．これにより，緯度経度座標 $(x_1, y_1), (x_2, y_2), \cdots$ に対応する道路リンク列 $e_1, e_2, \cdots$ が得られる．この際，e_i と e_{i+1} が隣接した道路リンクでない，ということが起こり得る．これは主に緯度経度座標の時刻の間隔が大きい時に発生する．これを修正する方法としては例えば，半教師付き学習の枠組みを用いて $(e_i, e_{i+1}) \rightarrow (e_i, e_{i+1/n}, e_{i+2/n}, e_{i+(n-1)/n}, e_{i+1})$ のようにギャップを埋める方法がある [11]．本研究ではマップマッチングアルゴリズム [14] の仮定に基づき，このようなギャップを最短経路で補間した．

時刻に関しては軌跡の経路に沿って GPS の点が射影された点の距離を用いて線形補間することで，t^{in} および t^{out} を得た．

4.2 移動パターンの索引化

ネットワーク制約軌跡の集合 $\mathbb{T} = \{\mathcal{T}_d \mid 0 \leqq d < D\}$ を考える．ここでは空間的な移動パターンのみの索引化について述べる．軌跡 d の移動パターン $\mathcal{T}_d.e$ に対して，順序を反転させた配列を $R_d := reverse(\mathcal{T}_d.e)$ とする．既に述べたように，これは道路リンク集合 E 上の文字列 (文書) とみなすことがで

(注2)：文書検索における転置インデックスでは典型的にはポスティングリストが文書 ID 順に並んでいるため，高速なマージが可能である [13]．一方で IJP アルゴリズムでは時刻に関する B^+ 木の索引で絞り込んだ複数のリストをマージするため，ソートが必要になる点も非効率である．

(注3)：本研究で取り扱うアルファベット集合は道路リンク集合 E であり，通常 $|E|$ は数万〜数百万である．これは自然言語の場合よりもかなり大きい．今回は大きなアルファベット集合に対しても適用可能なポインタのないウェーブレット木 [3] を用いた．

きる．そして得られた文書集合 $\{R_d\}$ を特殊文字 \$ を用いて結合した文字列を **軌跡文字列** $Y := R_0\$R_1\$\cdots\$R_{D-1}\$$ と定義する．ただし，\$ は道路リンク集合 E に含まれるどの道路リンクよりも辞書式順序が小さいものとする．図 1 の例では $Y = e_{ef}e_{ce}e_{bc}e_{ab}\$e_{ef}e_{de}e_{bd}e_{ab}\$e_{ef}e_{de}e_{bd}\$e_{ef}e_{de}\$$ となる．この軌跡文字列に対して FM-index を構築する．このようにして構築された索引を提案手法における空間索引と位置づける．

3.2 節で述べたように，この FM-index は与えられたパターン P に対応する接尾辞配列上の範囲 $[sp, ep)$ を高速に返すことができる．軌跡文字列 Y の接尾辞配列を $SA[0, |Y|)$ とする．このとき，P を反転させた文字列 $P_{rev} := reverse(P)$ に対応する接尾辞配列上の範囲を $[sp', ep')$ とすると $SA[sp', ep')$ は元の軌跡文字列 Y 上で P_{rev} が出現する位置のリストを過不足なく表している．このことより以下の補題が成り立つ．

補題 1. 軌跡文字列 Y に対して長さ m のクエリ経路パターン P を反転させた P_{rev} に対応する接尾辞配列上の範囲を $R(P_{rev}) := [sp', ep')$ とする．このとき，$j \in R(P_{rev})$ と $Y[SA[j], SA[j] + m) = P_{rev}$ は同値である．

ここで，接尾辞配列 SA の逆関数である逆接尾辞配列 ISA を考える．すなわち，任意の $0 \leqq j \leqq |Y|$ に対して $ISA[SA[j]] = j$ である．このとき，補題 1 を $j = ISA[i]$, $SA[j] = i$ を用いて書き換えることで以下が成り立つ．

命題 1. 軌跡文字列 Y に対して長さ m のクエリ経路パターン P を反転させた P_{rev} に対応する接尾辞配列上の範囲 $R(P_{rev}) := [sp', ep')$ とする．このとき，$ISA[i] \in R(P_{rev})$ と $Y[i, i + m) = P_{rev}$ は同値である．

この命題が示唆するのは，もし P_{rev} に対応する範囲 $R(P_{rev})$ を予め知っており (これは FM-index で高速に求められる)，その上で位置 i の $ISA[i]$ を $sp' \leqq ISA[i] < ep'$ のようにチェックすれば，位置 i が P_{rev} にマッチするかどうか (したがって逆順で見た時に P にマッチするかどうか) を一つのスカラー値に関する不等式で判定できるということである．提案手法のキーとなるアイデアは，この $ISA[i]$ の値を時刻や軌跡 ID を格納するディスク上のデータベースに格納するということである．これにより走行した経路パターンとそれ以外のデータが紐付けられ，高速な検索が可能になる．以下の節ではこれらについて詳細に説明する．

4.3 時刻情報の索引化

議論を簡単にするために，時刻その他の情報は表 1 のようなテーブルに格納されているものとする．このテーブルには図 1 で用いたものと同じ 4 つの軌跡に対応するデータが格納されている．各行は 2.1 節で定義したタプル (d, e, t^{in}, t^{out}) に対応しており，軌跡 ID d, 道路リンク ID e, 時刻 t^{in}, t^{out} の他に i, $ISA[i]$ の列を持っている．ここで i は軌跡文字列 Y 中の位置に対応し，$ISA[i]$ は Y の逆接尾辞配列 (注4) の i 番目の値である．従って，表 1 の e-列は Y に一致する．

表 1　軌跡 ID，時刻情報，逆接尾辞配列などを格納するテーブルの構造．(e, t^{out}) のペアに対して B$^+$ 索引を構築する．格納された 4 つの軌跡は図 1 の例に対応する．

i	d	e	t^{in}	t^{out}	$ISA[i]$
0	0	e_{ef}	18	23	13
1	0	e_{ce}	12	18	9
2	0	e_{bc}	9	12	6
3	0	e_{ab}	5	9	5
4	—	\$	—	—	3
5	1	e_{ef}	19	22	16
6	1	e_{de}	13	19	12
7	1	e_{bd}	10	13	8
8	1	e_{ab}	6	10	4
9	—	\$	—	—	2
10	2	e_{ef}	16	19	15
11	2	e_{de}	11	16	11
12	2	e_{bd}	8	11	7
13	—	\$	—	—	1
14	3	e_{ef}	9	15	14
15	3	e_{de}	15	21	10
16	—	\$	—	—	0

このテーブルの道路リンク e と道路リンクを出た時刻 t^{out} の二つの列のペア (e, t^{out}) に対して，B$^+$ 木 を用いて索引化しておく．このような構造化はネットワーク制約軌跡のための索引化の手法として FNR-tree [5] や Vieira らの研究 [17]，Krogh らの研究 [10] で用いられているものと本質的には同じである．ただし，既存研究では逆接尾辞配列 ISA の値は格納されていない (注5)．

Vieira らが提案した IJP アルゴリズムでは，はじめにパターン $P = p_0\ p_1 \cdots p_{m-1}$ に含まれる各 p_j に対して，道路リンク ID e が p_j に一致する軌跡 ID のリストを (必要があれば時刻 t^{out} に関する条件を考慮して) $m = |P|$ 個のリストを抽出する．これは (e, t^{out}) 上に構築された索引によって比較的高速に実行できる．その後，転置インデックスにおける文書検索と同様の方法で $|P|$ 個のリストの共通部分を取ることで解を得る．この方法では B$^+$ 木 の探索回数が最大で $|P|$ となるので，大きな $|P|$ を持つクエリでは明らかに非効率である．また $|P|$ 個の各リストは軌跡 ID 順にソートされていないため，リストの結合にも大きなコストがかかってしまうという問題がある．

4.4 完全経路クエリのためのアルゴリズム

本研究ではテーブルに追加された逆接尾辞配列の値を用いて効率的な探索を行う．完全経路クエリ $SPQ(P, s, t)$ を考える．まず P の反転文字列 P_{rev} に対して，4.2 節で構築した軌跡文字列 Y の FM-index を用いて接尾辞配列上の範囲

(注4)：この例では，道路リンク集合 E 上の順序をリンク添字の辞書式順序で定義している．例えば $e_{bc} < e_{de}$ である．

(注5)：Krogh らの方法 [10] では ISA の代わりに軌跡から計算されるハッシュ値を格納することで近似的な検索を可能にする．提案手法はこのハッシュ値のフィールドを逆接尾辞配列に取り替えることで厳密な検索を実現するものとみなすことができる．

$R(P_{rev}) = [sp', ep')$ を求める．次に 4.3 節で定義したテーブルに対して，

- 道路リンク ID e がクエリ経路パターン P の最後の道路セグメント p_{m-1} と一致し，
- 時刻 t^{out} が $s \leqq t^{out} < t$ を満たし，
- $sp' \leqq ISA[i] < ep'$ を満たす

ような行の軌跡 ID を取り出す．最初の二つの e および t^{out} に関する条件は前節で説明した B^+ 木 の索引によって高速に実行可能である．それらのデータの中で最後の条件を満たすものが $SPQ(P, s, t)$ の解である．この最後の条件は後述するように，空間的な経路パターンのマッチングに対応している．この方法を表 1 (`tbl` とする) に対する SQL として表現すると以下のように非常にシンプルになる：

`SELECT` d `FROM tbl WHERE` $e = p_{m-1}$ `AND` $s \leqq t^{out} < t$ `AND` $sp' \leqq ISA < ep'$;

次節でこの方法が正しいことを示すが，ここでは表 1 の例を用いて説明する．2.2 節で用いた $P = e_{bd} \to e_{de}$ および $[s, t) = [18, 25)$ の例について再度考えてみる．まず，P_{rev} に対応する接尾辞配列上の範囲は $[sp', ep') = [11, 13)$ となる (具体的な計算は省略する)．次に P の最後の要素である e_{de} を走行し，かつ t^{out} が $[18, 25)$ に含まれる要素を取り出す．その結果，$i = 6$ 行目および，$i = 15$ 行目が抽出される．$i = 6$ のとき，$ISA[6] = 12 \in [11, 13)$ なので，この行 (軌跡 ID $d = 1$) はアルゴリズムによって抽出される．一方，$i = 15$ のとき，$ISA[15] = 10 \notin [11, 13)$ なので，この行 (軌跡 ID $d = 3$) はアルゴリズムによって抽出されない．すなわち，$d = 1$ のみが $SPQ(P, s, t)$ の解として抽出されることになる．実際，$d = 3$ の軌跡は e_{de} を $[18, 25)$ に通過するものの，$e_{bd} \to e_{de}$ という経路を走行していない．また，軌跡 ID $d = 2$ は $e_{bd} \to e_{de}$ のように走行しているものの，e_{de} を出た時刻が $16 \notin [18, 25)$ なのでやはりマッチしていない．したがって，提案アルゴリズムは正しい結果を返していることがわかる．

4.5 提案手法の性質

本節では提案手法が持ついくつかの性質を示す．はじめに，アルゴリズムの正しさを示す．

性質 1. (結果の厳密性) 4.4 節で示したアルゴリズムは厳密な $SPQ(P, s, t)$ の解を返す．

Proof. 4.4 節で示したテーブルに対する三つの検索条件のうち，最初の二つにマッチする集合を U とする．条件の設定の仕方から，$SPQ(P, s, t)$ は明らかに U に包含される．U に含まれる軌跡のうちで $SPQ(P, s, t)$ に含まれないものがあるとするならば，それは p_{m-1} 以前に走行した経路パターンが $p_0\, p_1 \cdots p_{m-2}$ にマッチしないような軌跡である．そのような false positives を取り除くために $sp' \leqq ISA[i] < ep'$ の条件を用いている．実際，この条件にマッチすることは命題 1 より $Y[i, i+m) = P_{rev}$ と等価なので，U に対して $sp' \leqq ISA[i] < ep'$ でフィルタリングしたものは $SPQ(P, s, t)$ と等しいことが言える． □

また，以下の性質はアルゴリズムの手続きより自明である．

性質 2. (ディスクアクセス) 提案アルゴリズムは任意の経路パターン P および時間幅 $[s, t)$ に対して，(e, t^{out}) を索引化する B^+ 木を一度だけ探索する．

この性質は 4.3 節で述べた IJP アルゴリズムが $|P|$ 回の探索を必要とすることと対照的である．また，Krogh らによる近似検索アルゴリズム [10] は ($|P|$ に依存しない) 二度の B^+ 木の探索で 近似的な結果 を返すものである．一方，提案手法は

- 厳密な結果，
- $|P|$ に依存しない B^+ 木 の探索回数，

を両立することができる．

性質 3. (時間複雑性) 提案手法の時間複雑性は $O(|P| \log |E| + \log |Y| + |U|)$ である．ただし U は性質 1 の中で定義したものである．

ここで $|P| \log |E|$ の項は FM-index の検索に対応し，これはメモリ内で実行されるため極めて高速であるため，実験の章で見るように，実践的には所要時間の $|P|$-依存性は生じない．なお，$\log |Y| + |U|$ は B^+ 木に関する時間複雑性である．一方で，IJP アルゴリズムの時間複雑性はおおよそ $O(|P| \log |Y| + |P| \cdot |U|)$ である．ただし，$|P| \log |Y|$ の項は B^+ 木の $|P|$ 回の探索に対応し，$|P| \cdot |U|$ の項はマージに対応する．

4.6 その他のクエリの取り扱い

提案した索引構造を用いて，これまでに扱ってきた完全経路クエリとは別のクエリも取り扱うことが可能である．例えば，Krogh らによって提案されたオリジナルの完全経路クエリの定義は以下のように本論文のものとわずかに異なる [10]：

$$SPQ'(P, s, t) := \{d \mid \exists i, j \,:\, \mathcal{T}_d.e[i, j) = P \wedge s \leqq \mathcal{T}_d.t^{in}[i] \wedge \mathcal{T}_d.t^{out}[j-1] < t\}. \quad (2)$$

すなわち，本論文の定義 (1) では P の最後の道路リンクを出た時刻のみを制約するのに対して，Krogh らによる定義 (2) では P 全体を出入りした時刻を制約する．式 (2) の時刻の条件は式 (1) よりも厳しいため，$SPQ'(P, s, t) \subset SPQ(P, s, t)$ が成立することがわかる．従って，まず $SPQ(P, s, t)$ を提案手法によって求めておき，次に道路リンク p_0 を時刻 $[s, t)$ 内に通過した軌跡 ID を求め，最後にそれらを適切に結合 (join) することで $SPQ'(P, s, t)$ を求めることができる．この方法では p_0 に関する処理においても B^+ 木を探索する必要があるため，合計 2 回の B^+ 木の検索が発生し，我々の完全経路クエリのアルゴリズムよりも約二倍遅くなる．

また，二つの経路 P_1, P_2 およびワイルドカードを表す経路 P_* に対して $P_1 P_* P_2$ にマッチする経路の検索なども IJP アルゴリズムの考え方と提案手法の考え方を組み合わせることで実現可能である．

図 4　マップマッチングに利用した地図 (ローマ市内，リンク数 56653)．色はデータの分布を示している．

4.7　索引の構築

最後に，提案した索引の構築方法について説明する．まず，4.1 節の方法に従って，車両軌跡データを前処理する．得られたネットワーク制約軌跡に対して軌跡文字列 Y を構成し，接尾辞配列 SA を例えば [15] の方法を用いて計算する．この接尾辞配列を用いて FM-index を構築することができる．同時に SA から逆接尾辞配列 ISA を定義通りに計算し，表 1 のように B^+ 木を構築することで提案した索引を構築することができる．

5.　実　　験

5.1　データセット

イタリアのローマ市内を走行したプローブカーデータ [1] を用いて実験を行った．このデータセットにはおよそ 320 台のタクシープローブの 2014 年 2 月の走行履歴であり，GPS のサンプリングレートは平均 7 秒である．このデータを 4.1 節で述べた隠れマルコフモデルによるマップマッチングの方法を用いて前処理し，ネットワーク制約軌跡の集合 $\mathbb{T}$ を得た．このデータには 130,000 を超える数の軌跡が含まれており，軌跡文字列の長さはおよそ 12,000,000 となった．なお，道路ネットワークとしては 56653 本の有向エッジからなる OpenStreetMap(注6) のデータを用いた (図 4)．

5.2　実　　装

提案手法および IJP アルゴリズム (Krogh らの厳密手法と同等のもの) による比較を行う．すべての手法は C++ によって実装し，g++ version 4.6.3 (`-O3` オプション) によりコンパイルを行った．ただし，ディスク上の B^+ 木としては sqlite3 を用いた．すべての実験結果は Intel Xeon W5590 CPU (3.33 GHz, 8 コア)，8 GB のメモリを搭載した Ubuntu Linux (12.04) 上で実行した．

(注6)：www.openstreetmap.org

5.3　結　　果

軌跡集合 $\mathbb{T}$ から長さ $|P| = 2, 5, 10, 20$ の走行パターンをランダムサンプリングし，500 個のクエリ走行パターン P を作成した．時間制約 $[s, t)$ に関しては 2014-02-01 0:00 から，7 日間，30 日間の 2 パターンのクエリに関して実験を行った．図 5 は 500 個のランダム軌跡クエリに対する平均応答時間を示している．横軸にはクエリ走行パターンの長さ $|P|$ をとっている．本論文における完全経路クエリの定義は Krogh らによるオリジナルの定義と異なるため，4.6 節で述べた，提案手法を応用した方法によってオリジナルの完全経路クエリに要した所要時間も同時に示している (図 5 内 “Proposed (Original definition)”)．我々の完全経路クエリの定義による結果は “Proposed (Our definition)” として図 5 に示した．

図 5 からは，提案手法においては $|P|$ が増加しても所要時間が増加しないのに対し，IJP アルゴリズムでは $|P|$ に従って処理時間がほぼ線形に増加することがわかる (図 5 中 “IJP (Original definition)”)．この IJP アルゴリズムの結果は 3.1 節で述べたように，各 $p_i \in P$ に対して B^+ 木を $|P|$ 回探索する必要があることに起因する．また，この増加量の傾きは 4.5 節で述べたように，$\log|Y| + |U|$ に比例するので，データが大きくなると提案手法との差はより大きくなると考えられる．

一方で提案手法のクエリ処理時間は $|P|$ と共に増加することはないが，このことは以下のように説明できる．提案手法の時間複雑性は $O(|P| \log|E| + \log|Y| + |U|)$ であったが，$|P| \log|E|$ の項は FM-index の検索 (接尾辞配列上の範囲 $[sp', ep')$ を求めるアルゴリズム) にかかるものである．これはメモリ上で処

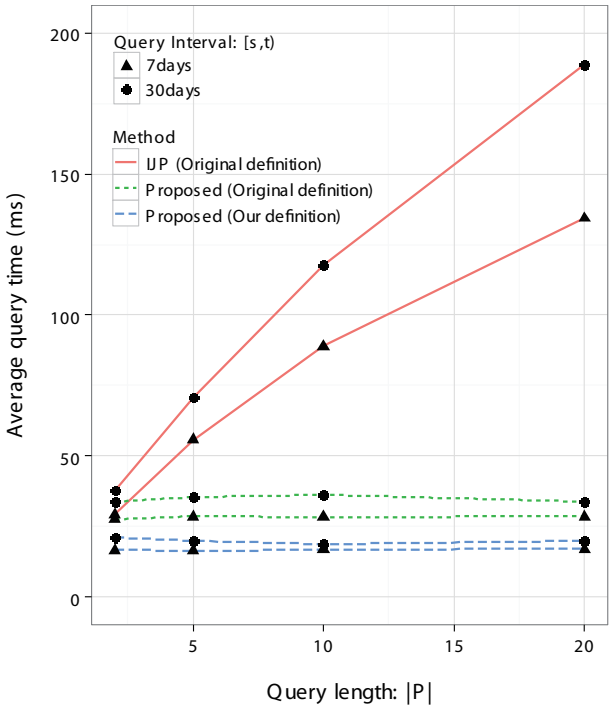

図 5　完全経路クエリに対する所要時間 (ミリ秒, 500 クエリの平均)

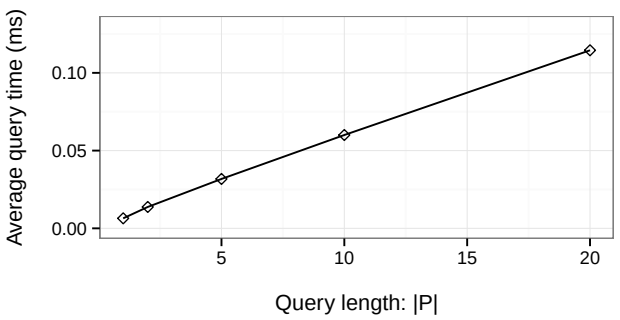

図 6　軌跡文字列 Y に関する FM-index に対して接尾辞配列上の範囲を求めるのに要した時間 (ミリ秒, 500 クエリの平均)

理されるために，ディスクアクセスの発生する B^+ 木の探索にかかる時間と比較してほとんど無視できる程度のものである．このことを示すために，FM-index の検索に実際に要した処理時間を図 6 に示す．この結果より，FM-index の検索に要する時間は $|P|$ に比例して大きくなるものの，高々数十マイクロ秒程度であり，全体の処理時間よりも二桁以上小さいことがわかる．

また，同じ提案索引構造を用いた場合でも，本研究における完全経路クエリの定義式 (1) と既存研究 [10] での定義式 (2) では所要時間がおよそ二倍ほど異なる (それぞれ図 5 内 “Proposed (Our definition)” および “Proposed (Original definition)” に対応)．これは既に 4.6 節で述べたとおり，オリジナルのクエリを提案手法を用いて解く場合には二回の B^+ 木の探索が必要になるためである．しかしながら，処理時間が $|P|$ に依存しない，という性質は同様である．

さらに，図 5 からは時刻制約 $[s,t)$ の期間が長くなると，既存手法，提案手法の両方において所要時間は増加するという傾向が見てとれる．これは期間 $[s,t)$ が長くなるに従って，返される結果の数 (もしくは 4.5 節で述べた $|U|$ のサイズ) が多くなることによるものと考えられる．6.2 節において，この点について議論を行う．

6. 議　　論

本節では前節での数値実験などを踏まえ，索引サイズ，クエリ最適化，データ構造の最適化，索引の更新の 4 つの観点から提案手法について議論をする．

6.1 索引のサイズ

FM-index はメモリ上に構築されることを前提とした索引であるため，そのサイズはコンパクトであることが望ましい．今回は圧縮手法を用いていないが，そのサイズは 33 MB であった．従って数十ギガバイトのメモリを搭載した計算機を用いることで今回のデータセットよりも数百倍大きなものに関しても容易に扱うことができる．一般に，データ分布 (文字の出現分布) には図 4 に示すように偏りがあるため，索引の圧縮 [8], [12] などの技術を用いることで，より大きなデータセットも取り扱うことができると考えられる．

6.2 クエリ最適化

前節において，提案手法は既存技術と比較して高速に完全経路クエリを処理することを見たが，ここではクエリ最適化について簡単に議論を行う．例えば 4.4 節で述べた 3 つのフィルタリング条件のうち，時刻に関する条件 $s \leqq t^{out} < t$ および逆接尾辞配列に関する条件 $sp' \leqq ISA[i] < ep'$ の適用順序を逆にすることが考えられる．ISA に対してインデックスを作成しておけばこの順序でも高速にフィルタリング可能である．高速化のためには $s \leqq t^{out} < t$ および $sp' \leqq ISA[i] < ep'$ の条件のうちで，ヒットするレコードの件数がより少なくなると見積もれるものを最初に用いてフィルタリングすればよい．後者の条件の件数は $ep' - sp'$ であり，前者の件数は時間幅 $t - s$ に比例するため，これらの件数をおおよそ見積もることが可能であり，これによりクエリ最適化を実現できる．

6.3 データ構造の最適化: 道路 ID e-列の除去

実は逆接尾辞配列の情報を保持しておけば，表 1 における e の列をデータベースに格納する必要がないということが言える．この理由は以下のとおりである．まず，命題 1 より任意の道路リンク $a \in E$ に対して $Y[i] = a$ と $C[a] \leqq ISA[i] < C[a+1]$ は同値であることがわかる．ただし，$C[a]$ は a よりも辞書順が小さい道路リンクが軌跡文字列 Y において出現した回数である．つまり表 1 のある行が a にマッチすることは $C[a] \leqq ISA[i] < C[a+1]$ を調べればよいので e を格納しておく必要はないことがわかる (注7)．すなわち (ISA, t^{out}) の組に対して B^+ 木 を構築すれば本論文で提案したアルゴリズムを実行するのに十分であることが言える．

逆に e-列が除去された後の表 1 のある行を見た時に対応する道路 ID を $ISA[i]$ から復元するには，上で述べた配列 C を二分探索すればよい．もしくは [16] で提案されている簡潔ビットベクトルを用いてこの探索を実現することも考えられる．

6.4 索引の更新

提案手法は過去の履歴データに対する索引化を対象としており，データの動的な更新はサポートしていない．一つの解決法として索引をある時間幅ごとに逐次的に構築することが考えられる．これによってすべてのデータを構築し直す必要がなくなる．より動的なデータの取り扱いは今後の課題である．

7. おわりに

本論文ではネットワーク上を移動する車両の軌跡に対する完全経路クエリに対する効率的な応答を可能にする索引構造の提案を行った．提案手法では，車両軌跡 (道路リンクの列) を文字列とみなし，全文索引である FM-index と時刻の索引である

(注7)：また，ISA は一意であるので主キーとしても用いることができる．

B^+ 木を逆接尾辞配列を用いて統合した．その結果，B^+ 木の探索回数がクエリ経路長 $|P|$ に依存しない，かつ厳密な結果を保証する手法であることを示された．また，実データを用いた実験において，提案手法が実際に既存手法よりも高速であることを示した．

提案手法はディスク上に格納する方法には強い制約を持たない．つまり基本的な B^+ 木 による索引が利用可能であれば基本的にはどのようなデータベースシステム上にも逆接尾辞配列のフィールドを追加することで実装することができることも大きな利点の一つである．このことは提案手法の高い拡張性を意味しており，今後は車両走行の時空間パターンとそれ以外の車載センサなどを合わせたデータマイニングへの応用などが期待される．

文　献

[1] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. CRAWDAD data set roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/, July 2014.

[2] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. In *Technical Report 124*. Digital Equipment Corporation, 1994.

[3] F. Claude and G. Navarro. Practical rank/select queries over arbitrary sequences. In *In Proc. 15th SPIRE, LNCS 5280*, pages 176–187, 2008.

[4] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 390–, 2000.

[5] E. Frentzos. Indexing objects moving on fixed networks. *Proc. of the 8th Intl. Symp. on Spatial and Temporal Databases (SSTD)*, pages 289–305, 2003.

[6] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 841–850, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[7] C. Guo, J. Meguro, Y. Kojima, and T. Naito. Automatic lane-level map generation for advanced driver assistance systems using low-cost sensors. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3975–3982, 2014.

[8] J. Kärkkäinen and S. J. Puglisi. Fixed block compression boosting in fm-index. In *Proceedings of String Processing and Information Retrieval (SPIRE '11)*, pages 174–184, 2011.

[9] S. Koide, Y. Tadokoro, and T. Yoshimura. Snt-index: Spatio-temporal index for vehicular trajectories on a road network based on substring matching. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Smart Cities and Urban Analytics*, UrbanGIS'15, New York, NY, USA, 2015. ACM.

[10] B. Krogh, N. Pelekis, Y. Theodoridis, and K. Torp. Path-based queries on trajectory data. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 341–350, New York, NY, USA, 2014. ACM.

[11] M. Li, A. Ahmed, and A. J. Smola. Inferring movement trajectories from gps snippets. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 325–334, New York, NY, USA, 2015. ACM.

[12] V. Mäkinen and G. Navarro. Implicit compression boosting with applications to self-indexing. In *Proceedings of String Processing and Information Retrieval (SPIRE '07)*, pages 229–241, 2007.

[13] C. D. Manning and P. Raghavan. **情報検索の基礎**. 共立出版, 2012.

[14] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 336–343. ACM, 2009.

[15] G. Nong, S. Zhang, and W. H. Chan. Two efficient algorithms for linear time suffix array construction. *IEEE Trans. Comput.*, (10):1471–1484, 2011.

[16] D. Okanohara and K. Sadakane. Practical entropy-compressed rank/select dictionary. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 60–70, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[17] M. R. Vieira, E. Frías-Martínez, P. Bakalov, V. Frías-Martínez, and V. J. Tsotras. Querying spatio-temporal patterns in mobile phone-call databases. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pages 239–248, 2010.

[18] M. Yoshikawa and T. Amagasa. Xrel: A path-based approach to storage and retrieval of xml documents using relational databases. *ACM Trans. Internet Technol.*, 1(1):110–141, Aug. 2001.

[19] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 99–108, New York, NY, USA, 2010. ACM.

[20] 岡野原大輔. **高速文字列解析の世界: データ圧縮・全文検索・テキストマイニング**. 岩波書店, 2012.