連続系列パターン照合アルゴリズム N-OPS の性能評価

李　光浩† 大森　匡† 星　守†

† 電気通信大学大学院情報システム学研究科　〒 182–8585 東京都調布市調布ヶ丘 1–5–1
E-mail: †{li,omori}@hol.is.uec.ac.jp

あらまし　近年，データストリームいう新しい情報源が出現し，我々の日常生活に重要な情報を提供してくれるようになった．データストリームの例としては，センサデータ，株価データ，ネットワークトラフィックデータなどがある．また，これらの中からユーザがほしい部分列だけを探索したいという要求がある．本論文で提案された N-OPS アルゴリズムは，系列パターン探索問題を文字列照合法でモデル化する際の枠組の探索方法である．今回は，述語間の論理関係が 2 点間文脈に依存する場合だけに限り，完備な N-OPS アルゴリズム技術について説明し，他方式と比べながら性能評価をすることを目的とする．

キーワード　データストリーム，系列パターン，照合，性能評価

Evaluation of Sequence Pattern Search Algorithm N-OPS

Guanghao LI†, Tadashi OHMORI†, and Mamoru HOSHI†

† Graduate School of Information Systems, The University of Electro-Communications　1–5–1 Chofugaoka, Chofu-shi, Tokyo, 182–8585 Japan
E-mail: †{li,omori}@hol.is.uec.ac.jp

Abstract　In recent years, a new source which is named data stream appeared and it offers important information to our daily life. Examples of data stream are sensor data, stock data, traffic data, etc. At the same time, there are a lot of demands from the users who only want to find a partial sequence which they need in a given data stream. In order to solve this problem, we propose an algorithm named N-OPS. It is a sequence pattern search algorithm which is based on text searching algorithm to solve sequence pattern search problem. We explain complete technology about N-OPS algorithm in the case that the logical relation among pair of the pattern elements is limited with only 2 sequential points. Last, we evaluate N-OPS algorithm in comparison with other searching methods.

Key words　data stream, sequence pattern, search, evaluation

1. はじめに

近年，インターネットとウェブ技術の発達によって，インターネット上でデータストリームがよく流れてくる [1][5][6][7][8][9]. このようなデータストリームの中からユーザがほしい部分列だけを抽出したいという要求が多くある．この問題は文献 [2] において扱われ，文字列照合法 [3] の考えに基づいた手法 OPS がある．しかし，系列パターンを構成するパターン要素に閉包を許す場合，OPS ではパターン要素間の論理関係が排他的でない場合に解を見逃す可能性がある [4]．この探索を完備にするよう新しくつくったものが，我々が提案した N-OPS[4] である．

N-OPS は，データストリーム中の系列パターン探索問題を文字列照合法でモデル化する際に深さ優先探索によって完備な系列パターンとの照合を行う．そのためには，OPS にはないデータ構造 HF，$count[]$ 移動規則を新しく用意する必要がある．

本稿は，文献 [4] の続編として，述語間の論理関係が 2 点間文脈に依存する場合に限り，データ構造 H,HF と $count[]$ の移動規則を含む完備な N-OPS 技術について説明し，他方式と比べて性能評価をすることを目的とする．

2. N-OPS の概要

本節では，N-OPS の概要について説明する．

2.1 系列パターン探索問題の定義

本研究では，蓄積されたデータ列から，あるいは連続的に流れてくるデータストリームからユーザが指定した系列パターンを満たすデータの部分列を効率良く取り出す問題を論じる．

ここで，系列パターンとは，データストリームを構成する 1 レコードへのブール述語を p_j としたとき，p_j の連接と閉包からなる正規表現で表されるようなパターンのことを指す．また，N-OPS では，系列パターン $P = p_{j-1}p_j^*p_{j+1}$ を

$P=\{p_{j-1},*p_j,p_{j+1}\}$ と表記する．ここで，記号「,」は連接を示し，記号「*」は閉包を示す．

例えば，系列パターン $P=\{p_{j-1},*p_j,p_{j+1}\}$ は，「p_{j-1} を満たすレコードが 1 つ来て，この直後に p_j を満たすレコードがいくつか連続して来て，次に p_{j+1} を満たすレコードが来る」という条件を満たすパターンのことを意味する．

ここで，データストリームからの系列パターン照合問題は，入力レコードが多属性であり，パターンを構成する要素がブール述語であるという点が普通の文字列照合問題と異なる．また，述語 p_j には「現レコードの属性 x の値が直前のレコードの属性 x の値より小さい」のような直近の 2 点間文脈に依存した 2 点間文脈述語と，「現レコードの属性 x の値が現時点までの全てのレコードの平均値より大きい」のような N 点間文脈に依存した N 点間文脈述語がある．

本研究では，パターン P を構成する要素間の論理関係を 2 点間文脈に依存した場合に限定して，データストリームからの系列パターン探索問題について述べる．

2.2 N-OPS の考え方

任意の時刻を t とし，時刻 t に到達したデータを x_t とすると，ストリームデータは I={...,x_{t-1},x_t,x_{t+1},...} と表現できる．例えば，系列パターン P は P=$\{p_1,*p_2,*p_3,p_4\}$ から構成され，入力レコード I は I=$\{18,19,20,21,17,23,22,27,16,14,15,...\}$ から構成されたとする．ここで，「2 点間文脈述語 p」とは，直近の 2 点 x_{t-1},x_t で真偽が決まるブール述語である．記号 , は連接を示し，記号 $*$ は閉包を示す．また，パターンの要素を p_1=$(x_{t-1}<x_t)$,p_2=$(x_{t-1}<x_t)$, p_3=$(x_t<25)$,p_4=$(x_t<16)$ とすると，上述の入力データ I から系列パターン P を探索する際の照合状況は図 1 に示したようになる．

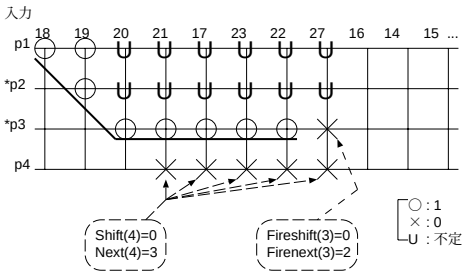

図 1　N-OPS を用いた場合の照合状況

この例では，入力レコード 27 と述語 p_4 との照合が失敗し，また，入力レコード 27 と述語 p_3 との照合が失敗して，次に，入力レコード 20 から述語 p_2 で照合再開になることを示している．

N-OPS では，パターンを構成する述語間の論理関係によって入力レコードと系列パターンとの照合を行う．述語間の論理関係は，次の式 1 と式 2 で示す 6 つの論理関係の内どれかに分類し，動的な照合状態を表現するデータ構造を用意する．($j \geqq k$)

$$\theta_{jk}=\begin{cases}1 & if\ (p_j \neq F)\wedge(p_j \Rightarrow p_k)\\ 0 & if\ (p_j \neq F)\wedge(p_j \Rightarrow \neg\ p_k)\\ U & (\text{上記以外})\end{cases} \quad (1)$$

$$\phi_{jk}=\begin{cases}1 & if\ (p_j \neq T)\wedge(\neg\ p_j \Rightarrow p_k)\\ 0 & if\ (p_j \neq T)\wedge(\neg\ p_j \Rightarrow \neg\ p_k)\\ U & (\text{上記以外})\end{cases} \quad (2)$$

また，N-OPS では深さ優先探索でデータストリームからの系列パターン探索を行う．ここで，深さ優先探索とは，パターン $P=\{p_{j-1},*p_j,p_{j+1}\}$ に対して，入力レコードと述語 p_j の照合が成功したら，述語 p_j が閉包でありなしに関らず，次の述語 p_{j+1} と次の入力レコードとの照合に進めるという探索である．

2.3 では，2 点間文脈に依存する場合，照合状態を表すデータ構造 H_j と HF_j について説明する．

2.3 2 点間文脈に依存する場合の N-OPS のデータ構造 H_j と HF_j

N-OPS では，述語と入力レコードとの照合が失敗したときに照合の再開位置を決めるためのデータ構造 H_j と HF_j を用意する．ここで，H_j は今の照合経路で述語 p_j との照合が初めて失敗した時点での照合状況を表すデータ構造であり，HF_j は閉包である述語 p_j との照合が少なくとも 1 回成功した直後に失敗した時点での照合状況を表すデータ構造である．次の式 3 と式 4 はデータ構造 H_j と HF_j を表すものである．

$$H_j=\begin{bmatrix}\theta_{1,1} & \theta_{2,1} & \cdots & \theta_{j-1,1} & \phi_{j,1}\\ & \theta_{2,2} & \cdots & \cdots & \phi_{j,2}\\ & & \ddots & \vdots & \vdots\\ & & & \theta_{j-1,j-1} & \phi_{j,j-1}\\ & & & & \phi_{j,j}\end{bmatrix} \quad (3)$$

$$HF_j=\begin{bmatrix}\theta_{1,1} & \theta_{2,1} & \cdots & \theta_{j-1,1} & \theta_{j,1} & \phi_{j,1}\\ & \theta_{2,2} & \cdots & \cdots & \theta_{j,2} & \phi_{j,2}\\ & & \ddots & \vdots & \vdots & \vdots\\ & & & \theta_{j-1,j-1} & \theta_{j,j-1} & \phi_{j,j-1}\\ & & & & \theta_{j,j} & \phi_{j,j}\end{bmatrix} \quad (4)$$

上記のデータ構造 H_j と HF_j によって，失敗した時点での照合再開位置が決まる．例えば，2.2 の例でのパターン要素間の論理関係から図 2 のような照合状態を表すデータ構造 H_j と HF_j が得られる．具体的にいうと，図 1 の照合で，入力レコード 18 と述語 p_1 との照合が成功し，次は入力レコード 19 と述語 p_2 との照合が成功し，その次は，入力レコード 20 と述語 p_3 との照合が成功する．その直後，入力レコード 21 と述語 p_4 との照合が失敗する．このとき，述語間の論理関係から，図 2 で表されたように，照合状況を表すデータ構造 H_4 が求められる．続いて，照合を行うとすると，最左照合によって，入力レコード 21 と閉包である述語 p_3 との照合が行って成功する．続いて，入力レコード 17 と述語 p_4 との照合が失敗し，述語 p_3 との照合が成功する．同じように，入力レコード 23，22 と述語 p_4 との照合が失敗し，述語 p_3 との照合が成功する．次は，入力レコード 27 と述語 p_4 との照合が失敗し，また，述語 p_3 との照合が失敗する．このとき，述語 p_3 との照合が 1 回以上成功した後に失敗したため，述語間の論理関係から，ここまでの照合状態を表すデータ構造 HF_3 が求められる (図 2 のデータ構造

HF_3).

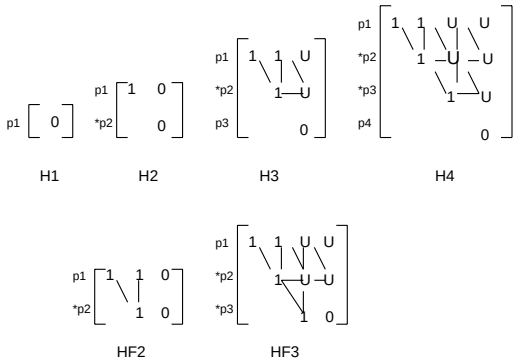

図 2　データ構造 H_j と HF_j

図 3 では，図 1 での照合と同様な照合を示す．ここで，図 3(a) の点線に囲まれた部分で述語 p_3 と入力レコード 20 から 22 までとの照合状態が同じであり，この部分を圧縮して 1 列にしたものが図 3(b) に示したデータ構造 H_4 の第 3 列の点線に囲まれた 1 列である．

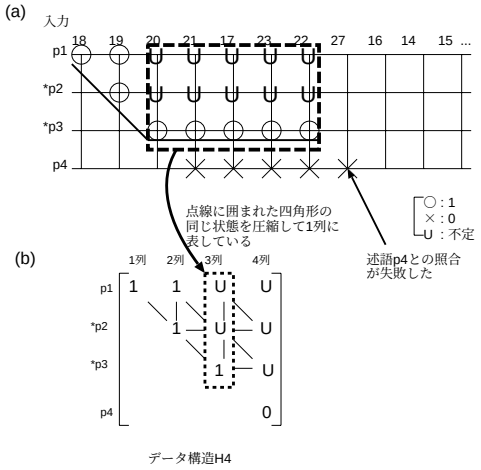

図 3　照合状況とデータ構造 H_4

図 2 に示した N-OPS のデータ構造に縦，横，斜のアークがあるが，これらはデータ構造 H_j，HF_j を構成する要素間に経路になる可能性があることを示す．アークの作り方としては，次のようになる．

もし，データ構造 H_j，あるいは HF_j を構成する要素を $[x, y](x \leqq y)$ とし，$x =$ データ構造 H_j，あるいは HF_jの行番号, $y =$ データ構造 H_j，あるいは HF_jの列番号 として横，縦，斜のアークを一般化して表すと次のようになる．

1. パターン要素 p_x と p_y が閉包でない，かつ，要素 $[x+1, y+1]$ が存在する，かつ，値が 0 でない場合 (図 4(a))：
$[x, y] \rightarrow [x+1, y+1]$
2. パターン要素 p_x が閉包であり，p_y が閉包でない，かつ，要素 $[x+1, y+1]$ が存在する，かつ，値が 0 でない場合 (図 4(b))：

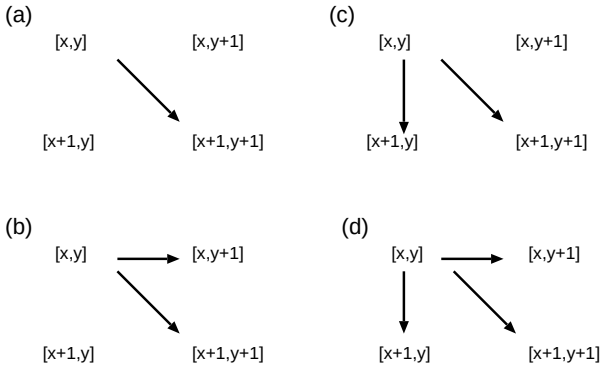

図 4　アークの描き方

$[x, y] \rightarrow [x, y+1]$
$[x, y] \rightarrow [x+1, y+1]$
3. パターン要素 p_x が閉包でない，p_y が閉包である，かつ，要素 $[x+1, y+1]$ が存在する，かつ，値が 0 でない場合 (図 4(c))：
$[x, y] \rightarrow [x+1, y+1]$
$[x, y] \rightarrow [x+1, y]$
4. パターン要素 p_x と p_y 両方が閉包である，かつ，要素 $[x+1, y+1]$ が存在する，かつ，値が 0 でない場合 (図 4(d))：
$[x, y] \rightarrow [x, y+1]$
$[x, y] \rightarrow [x+1, y+1]$
$[x, y] \rightarrow [x+1, y]$

上述のデータ構造 H_j と HF_j によって，入力レコードと述語 p_j の照合が失敗した時点での次の照合再開経路が求められる．

次の 2.4 では，再開位置情報の一部としての $shift(j)$，$next(j)$ と $Fireshift(j)$，$Firenext(j)$ について説明する．

2.4　2 点間文脈に依存する場合の $shift(j)$，$next(j)$ と $Fireshift(j)$，$Firenext(j)$

N-OPS では，データ構造ごとにそれぞれに対応する再開位置情報を用意する．すなわち，データ構造 H_j を用いた場合の再開位置情報を $shift(j)$，$next(j)$ とし，次のように定義する．$shift(j)$ は，入力レコードと述語 p_j の照合が失敗した時点での照合状態において「p_1 の照合開始位置を右シフトするシフトの回数」であり，$next(j)$ は「指定した位置へシフトした後に照合を再開する述語の番号」である．

$shift(j)$ の決定手順

データ構造 H_j において，要素 $[1, 1+s](0 \leqq s < j)$ からデータ構造 H_j の最右列 (すなわち，j 列) まで到達可能な経路が存在する s の集合を $\sigma(j)$ とする．

1. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} = 0$ なら，$shift(j) = j$ にする．
2. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} \neq 0$ なら，$shift(j) = j-1$ にする．
3. もし $\sigma(j)$ が空集合でないなら，$shift(j) = min(\sigma(j))$ にする．

$next(j)$ の決定手順

データ構造 H_j において，要素 [1, 1 + s](0 ≦ s < j) からデータ構造 H_j の最右列まで到達可能な経路が存在する s の集合を $\sigma(j)$ とする．要素 $[1, 1 + min(\sigma(j))]$ からデータ構造 H_j の最右列まで到達可能な経路集合のうち，深さ優先探索で最初に見つかる経路を $path_1$ とする．

1. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} = 1$ なら，$next(j) = 2$ にする．

2. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} \neq 1$ なら，$next(j) = 1$ にする．

3. 経路 $path_1$ 上の要素の値がすべて 1 であるとき，その終端位置を $[endx, endy]$ とすると，$next(j) = 1 + endx$ にする．

4. 経路 $path_1$ 上に U があるとき，最初に U が出た位置を $[x_u, y_u]$ とすると，$next(j) = x_u$ にする．

また，データ構造 HF_j を用いた場合の再開位置情報を $Fireshift(j)$，$Firenext(j)$ とし，次のように定義する．$Fireshift(j)$ は，入力レコードと述語 p_j の照合が失敗した時点での照合状態において「p_1 の照合開始位置を右シフトするシフトの回数」であり，$Firenext(j)$ は「指定した位置へシフトした後に照合を再開する述語の番号」である．

$Fireshift(j)$ の決定手順

データ構造 HF_j において，要素 [1, 1 + s](0 ≦ s ≦ j) からデータ構造 HF_j の最右列 (すなわち，(j + 1) 列) まで到達可能な経路が存在する s の集合を $\sigma(j)$ とする．

1. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} = 0$ なら，$Fireshift(j) = j + 1$ にする．

2. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} \neq 0$ なら，$Fireshift(j) = j$ にする．

3. もし $\sigma(j)$ が空集合でないなら，$Fireshift(j) = min(\sigma(j))$ にする．

$Firenext(j)$ の決定手順

データ構造 HF_j において，要素 [1, 1 + s](0 ≦ s ≦ j) からデータ構造 HF_j の最右列 (すなわち，(j+1) 列) まで到達可能な経路が存在する s の集合を $\sigma(j)$ とする．要素 $[1, 1+min(\sigma(j))]$ からデータ構造 HF_j の最右列まで到達可能な経路集合のうち，深さ優先探索で最初に見つかる経路を $path_1$ とする．

1. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} = 1$ なら，$Firenext(j) = 2$ にする．

2. もし $\sigma(j)$ が空集合であり，かつ，$\phi_{j,1} \neq 1$ なら，$Firenext(j) = 1$ にする．

3. 経路 $path_1$ 上の要素の値がすべて 1 であるとき，その終端位置を $[endx, endy]$ とすると，$Firenext(j) = 1 + endx$ にする．

4. 経路 $path_1$ 上に U があるとき，最初に U が出た位置を $[x_u, y_u]$ とすると，$Firenext(j) = x_u$ にする．

図 1 の例をみると，入力レコード 27 とパターン要素 p_4 との照合で失敗した場合，図 2 のデータ構造 H_4 を用いて照合再開位置を決めれば良い．データ構造 H_4 によると，次の経路は，要素 [1,1](すなわち，第 1 行第 1 列の要素) → [2,2] → [3,3] → [3,4] からなり，開始点の位置は変わらず ($shift(4) = 0$)，失敗した時点での入力とパターン要素 p_3($next(4) = 3$) との照合から再開すれば良い．

また，入力レコード 27 とパターン要素 p_3 との照合が失敗した場合，ここまでパターン要素 p_3 との照合が一回以上成功したため，図 2 のデータ構造 HF_3 を用いて照合を再開する．データ構造 HF_3 によると，次の経路は，要素 [1,1] → [2,2] → [2,3] → [2,4] からなり，開始点の位置は変わらず ($Fireshift(3) = 0$)，入力レコード 20 とパターン要素 p_2($Firenext(3) = 2$) との照合から再開すれば良い．

3. データ系列における連続 2 点間文脈に依存した N-OPS アルゴリズム

2 節までは，生じる照合状況を表現するデータ構造 H_j と HF_j の作り方とデータ構造 H_j と HF_j による照合再開要素の位置の求め方について説明した．しかし，実行時に，データ構造 H_j と HF_j の圧縮された部分に多数の照合が行う場合，照合再開位置への移動判定が必要である．そのため，3.1 では，我々が提案した N-OPS の経路情報の表記法について説明し，3.2 では照合再開位置を決めるための入力 i と各述語における $count[]$ の移動処理について説明する．

3.1 N-OPS の経路情報の表記法

データ構造 H_j と HF_j を用いて，失敗した時点での次の経路を表し，照合再開位置を決めるために，下記の提案データ構造を使って経路情報を表記する．

経路情報の提案データ構造

```
struct path_info{
  int path_x[];//H,HF における要素の列番号
  int path_y[];//H,HF における要素の行番号
  char parse[];//H,HF における要素の内容
}H_info[],HF_info[];
```

今回は，([行番号, 列番号], 要素内容) の順で，経路情報を表記する．図 2 で示した H_4 を用いた場合の次に行くべきの経路を表すと，([1,1],1) → ([2,2],1) → ([3,3],1) → ([3,4],U) になる．

次は，データ構造 H_j と HF_j から得た経路情報で縦行きの経路がある場合について述べる．

まず，系列パターン P は P=$\{p_1, *p_2, *p_3, *p_4, *p_5, p_6\}$ から構成され，入力レコード I は I=$\{x_1, x_2, x_3, ..., x_j, ..., x_n, ...\}$ から構成されたとする．そして，図 5(a) のような照合と図 5(b) のようなデータ構造 H_6 があるとする．

上記の経路情報の提案データ構造を用いて，深さ優先探索の戦略で照合を行い，述語 p_6 との照合が失敗した時点での経路情報を表現すると $([1, 4], 1) \rightarrow ([2, 5], 1) \rightarrow ([3, 5], 1) \rightarrow ([4, 6], U)$ になり，$shift$，$next$ の値は $shift(6) = 3$，$next(6) = 4$ になる．ここで，深さ優先で $([2, 5], 1) \rightarrow ([3, 5], 1)$ のような経路を「**縦行き**」の経路と呼ぶ．

しかし，図 5(a) のような照合を行うとすると，入力レコード x_{10} と述語 p_4 の照合が成功し，また入力レコード x_{11} と述語 p_5 の照合が成功し，次の入力レコード x_{12} と述語 p_6 の照合が失敗した場合，次にとる経路情報としては $([1, 4], 1) \rightarrow ([2, 5], 1) \rightarrow ([3, 5], 1) \rightarrow ([4, 6], U)$(図 5(b)) ではなく

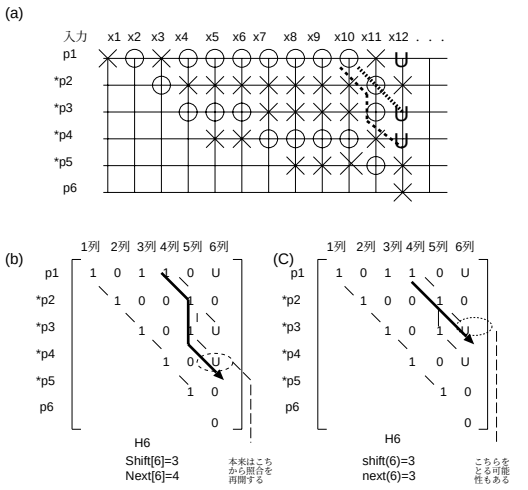

図 5　入力データと述語との照合状況とデータ構造 H_6

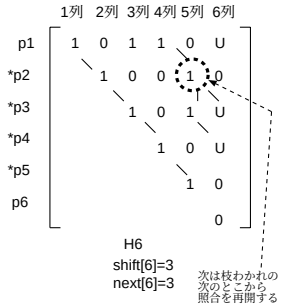

図 6　データ構造 H_6 と照合の再開位置

$([1,4],1) \rightarrow ([2,5],1) \rightarrow ([3,6],U)$ で (図 5(c)) あることが分かる．この場合，$shift$，$next$ の値は $shift(6)=3$，$next(6)=3$ になる．

このように縦行きがありうる場合，上記の例題のような問題を解決するために，次の方法が考えられる．

提案方法 データ構造 H_j と HF_j に基づいて深さ優先探索戦略によって経路になりうる情報を記録する．もし，最右列まで到達可能な縦行きの経路があれば，最初に現われた縦行き経路の一番目の要素までを経路情報とする．そして，この経路情報から $shift(j)$，$next(j)$ と $Fireshift(j)$，$Firenext(j)$ を求める．ここで，$next(j)$ と $Firenext(j)$ は，縦行き経路の一番目の要素に当る述語の次の述語の番号とする．

図 6 では図 5(a) の例を提案した方法によって照合再開位置を求めたものを示した．

今回は提案したこの方法を使って失敗した時点での次の経路を決める．また，この経路情報によって照合再開位置を決める．

3.2　照合再開位置を決めるための入力 i と $count[]$ の移動処理

N-OPS では，閉包で表された系列パターンと入力レコードとの照合を正確で効果的に行うために配列 $count[]$ を保持する．この $count[]$ は各パターン要素に 1 つの値を保持する．$count[j]$ はパターン要素 j までその系列パターンと一致した入力データの累積数を保持する．例えば，図 5(a) の場合，$count[]$ の値は，$count[1]=1, count[2]=2, count[3]=5, count[4]=9, count[5]=10$ になる．

問題は，ここまでの経路情報と各パターン要素が持っている $count[]$ 値を用いて，次の照合再開位置をどう決めるかということである．すなわち，もとの開始点からいくつの入力レコードまで移動させて照合を再開するかということである．これが入力 i と $count[]$ の移動処理である．

3.2.1　入力 i の移動

N-OPS は，次の 2 つの場合に分けて入力 i の移動処理を行う．

1. 縦行きの経路がない場合

図 7(a) は，データ構造 H_j を用いたとき，次の経路に縦行きの経路がない場合である．この場合，データ構造 H_j によると，次の照合は要素 $[k,m]$ から再開し，入力 i はもとの開始点から $count[m-1]$ を増やした位置まで移動させて，述語 p_k との照合から始まる．

2. 縦行きの経路がある場合

図 7(b) は，データ構造 H_j を用いたとき，次の経路に縦行きの経路がある場合である．この場合，データ構造 H_j によると，次の照合は要素 $[k,m]$ の次から再開し，入力 i はもとの開始点から $count[m-1]+1$ を増やした位置まで移動させて，述語 p_{k+1} との照合から始まる．

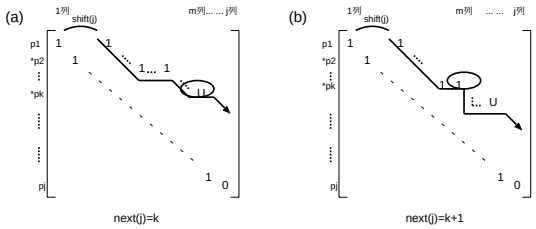

図 7　データ構造 H, HF による入力 i の移動

ここまでの処理によって，失敗した時点で次に照合を再開する入力 i の位置が求められる．しかし，次の経路の照合状況と照合位置を正確に把握するためには，もとの経路要素から次の経路要素までの $count[]$ の移動処理が必要である．

3.2.2　$count[]$ の移動

$count[]$ の移動処理には，データ構造 H_j を用いる場合とデータ構造 HF_j を用いる場合に分ける．また，各場合では，$shift(j)$，あるいは $Fireshift(j)$ の範囲，縦行き経路が有る無しの判断と経路要素の値が 1 か U かによって行われる．

次は，典型的な例で $count[]$ の移動処理について説明する．

1.「$0 \leqq Fireshift(j) < j$，且つ，縦行きの経路がない場合」の例

図 8 は，図 2 のデータ構造 HF_3 を用いた場合の $count[]$ の移動に関するものであり，縦行きの経路がない場合である．

この場合，データ構造 HF_3 における次の経路は ([1,1],1) → ([2,2],1) → ([2,3],U) → ([2,4],U) であり，移動処理の前の $count[]$ は $count[1]=1, count[2]=2, count[3]=7$ である．このとき，経路の要素 ([1,1],1) から ([2,2],1) までの論理関係が 1 であり，ここまでの経路が変えないため，この経路における $count[]$ は $count[1]=1, count[2]=2$ になる．

2.「$0 \leqq shift(j) < j-1$，且つ，縦行きの経路がある場合」

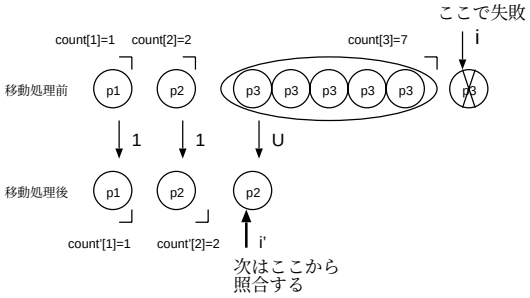

図 8　データ構造 HF_3 を用いた場合の $count[]$ 移動

の例

図 9 は，図 5 のデータ構造 H_6 を用いた場合の $count[]$ の移動に関するものであり，次の経路に縦行きがある場合である．

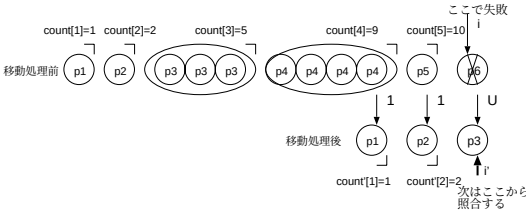

図 9　データ構造 H_6 を用いた場合の $count[]$ 移動

この場合，データ構造 H_6 における次の経路は ([1,4],1) → ([2,5],1) であり，移動処理の前の $count[]$ は $count[1] = 1, count[2] = 2, count[3] = 5, count[4] = 9, count[5] = 10$ である．ここで，述語 p_4 で 4 回 ($count[4] - count[3] = 4$) の照合が行ったが，次の経路要素 [1,1] にとって述語 p_1 が閉包でないため，この経路における $count[]$ は $count[1] = 1$ になり，$count[2] = 2$ になる．

これらの $count[]$ の移動処理によって，次の経路における各述語での $count[]$ 数が正確に求められる．

3.3　最悪場合の対応と処理

N-OPS では，述語間の論理関係が全て U である場合，述語 p_j との照合が失敗したら，次は，述語 p_j の直近の閉包である述語に戻って次の照合を行う．例えば，図 10 では，述語 p_5 との照合が失敗した場合，次は $restartpoint_1$ という位置から照合を再開したが，照合が失敗したため，$restartpoint_2$ という位置から照合を再開することを示す．この場合，述語 p_1, p_2, p_3, p_4, p_5 で行った入力レコードの最後の番号をそれぞれ $endi[1], endi[2], endi[3], endi[4], endi[5]$ とする．また，この時の各述語と $endi[1], endi[2], endi[3], endi[4], endi[5]$ からなる情報によって作られる経路 (図 10 の太線) を「**壁 (**$Wall$**)**」と呼ぶ．深さ優先探索で $restartpoint_2$ から照合を再開すると，点線で表された失敗した経路に沿って再度照合を行う．この場合，N-OPS では，最悪照合が生じ，照合回数が指数的に増加する可能性が起こる．

ここで，現在の照合位置がこの失敗した経路と接触して照合をすることを「**壁 (**$Wall$**) とぶつかる**」と呼ぶ．N-OPS では，最悪の場合を回避するために，次のように考える．

1. もし，今照合する要素の位置が壁とぶつからなければ，今の経路情報によって壁の情報を更新し，次の照合を行う．
2. もし，今照合する要素の位置が壁とぶつかるなら，直近の閉包である述語に戻って，その述語に当る壁の情報が示す位置の次の位置から照合を再開する．

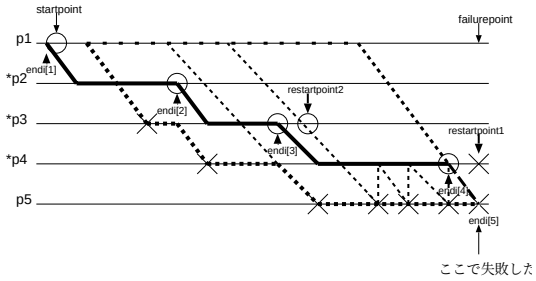

図 10　最悪時の照合状況

N-OPS の最悪の場合を回避するために，まず，次の定理 1 が成立することを示す．

定理 1 データ構造 H_j, HF_j による照合再開位置はいつも壁 ($Wall$) の上 ($over$) にある．

また，定理 1 によって，次の N-OPS の最悪場合の対応処理アルゴリズムが考えられる．

Program 3..1 N-OPS の最悪の場合の対応処理提案アルゴリズム

```
i,j ∈ proceed
LOOP:(i,j) try pattern(i,j);//パターン照合関数
if (Match Failure){
    データ構造 H，または，HF を使う;
}else{//Match Success
    i++;j++;
    if (Wall Conflict){
        直近の閉包である述語に戻って照合を再開する;
    }else{//成功処理
        goto LOOP;
    }
}
```

上述の N-OPS の最悪場合の対応処理アルゴリズムから，パターン長 $|P|$ と入力長さ N に対する計算量が理想的な場合に $O(|P| + N)$ であり，最悪でも $O(|P| \times N)$ まで抑えられることが分かる．

4.　実験と評価

本節では，2 点間文脈に依存した系列パターンを探索する際の N-OPS に関する実験評価を示す．

以下の実験では，N-OPS と最悪照合を回避する N-OPS(以下，N-OPS-Wall と呼ぶ)，$Naive$ な方法，並列照合法，並列照合法 (改) を用いた場合，それぞれのの照合回数を比較する．ここで $Naive$ な方法というのは，入れ子ループの動作原理によってデータ系列から系列パターンを探索する方法である．また，並列照合法というのは，非決定性有限オートマトンにより，入力

ごとにパターンを構成する全ての述語と並列的に照合を行う方法である．並列照合法 (改) は，並列照合法からの無駄な照合を回避した照合法である．

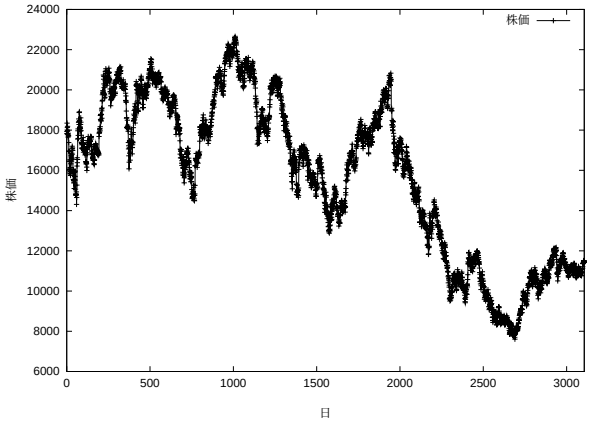

図 11　実験データ：1992/05/27 から 2005/01/07 までの日経平均株価の一日の終値 3107 点

実験データとしては，日経平均株価 [10] の一部である 1992/05/27 から 2005/01/07 までの日経平均株価の一日の終値 3107 点を使用した (図 11).

4.1　実験１：述語間の論理関係が排他的でない場合

今回は，2 点間文脈に依存した連続系列パターン要素間の論理関係が排他的でない場合，N-OPS と N-OPS-Wall，$Naive$ な方法，並列照合法，並列照合法 (改) を用いた場合，それぞれのの照合回数を比較する．

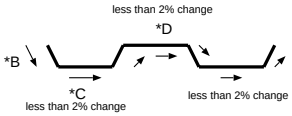

図 12　$double\quad bottom$

パターン P は，株価市場でよく現われる $double\quad bottom$ というものとする (図 12). すなわち，パターン P を構成する述語は A=$\{x_t \geqq 0.98 \times x_{t-1}\}$，$B$=$\{x_t < 0.98 \times x_{t-1}\}$，$C$=$\{0.98 \times x_{t-1} < x_t < 1.02 \times x_{t-1}\}$，$D$=$\{x_t > 1.02 \times x_{t-1}\}$，$E$=$\{x_t \leqq 1.02 \times x_{t-1}\}$ である．次は，パターン P を P=$\{A,*B,*C,D\}$,P=$\{A,*B,*C,*D,*C,*B,*C,D\}$,$P$=$\{A,*B,*C,*D,*C,*B,*C,*D,*E,*A,*B,C\}$,$P$=$\{A,*B,*C,*D,*C,*B,*C,*D,*E,*A,*B,*C,*D,*B,*D,C\}$ のように，パターン長を 4,8,12,16 として実験した．

図 13 は，N-OPS と N-OPS-Wall，$Naive$ な方法，並列照合法，並列照合法 (改) を用いた場合の照合回数の比較グラフである．

図 13 に示した照合回数から分かるように，同じパターンに対して，並列照合法で照合をした場合，計算量が $O(|P| \times N)$ になる．また，並列照合法 (改) で照合を行った場合，照合回数は並列照合法より少ない．また，$Naive$ な方法を使った方が並列照合法 (改) より照合回数が少なくて，照合成功までの処理が速かったと言える．N-OPS を用いて照合を行う場合，照合回数が $Naive$ な方法より少なく，処理が速かったと言える．N-OPS-Wall を用いて照合を行う場合，照合回数が一番少なく，処理が一番速かったと言える．

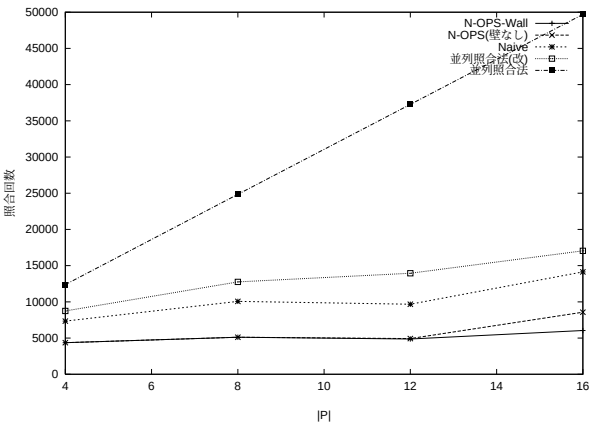

図 13　照合回数の比較グラフ (実験 1)

4.2　実験 2：述語間の論理関係が 0,1 の場合

今回は，2 点間文脈に依存した連続系列パターン要素間の論理関係が 0，1 である場合，N-OPS(この場合には，N-OPS-Wall と一致する)，$Naive$ な方法，並列照合法，並列照合法 (改) を用いた場合，それぞれのの照合回数を比較する．

パターン P は述語 A, B から構成され，A, B を $A = \{x_t < 0.98 \times x_{t-1}\}$,$B = \{x_t \geqq 0.98 \times x_{t-1}\}$ とする．次は，パターン P を P=$\{A,*B,*A,B\}$,P=$\{A,*B,*A,*B,*B,*A,*A,B\}$,$P$=$\{A,*B,*A,*B,*B,*A,*A,\ *B,*B,*A,*A,B\}$,$P$=$\{A,*B,*A,*B,*B,*A,*A,\ *A,*B,*A,*A,B\}$ のように，パターン長は 4,8,12,16 として実験した．

図 14 は，N-OPS と N-OPS-Wall，$Naive$ な方法，並列照合法，並列照合法 (改) を用いた場合の照合回数の比較グラフである．

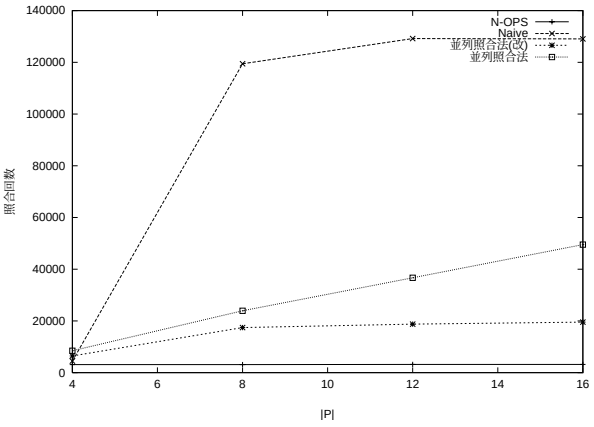

図 14　照合回数の比較グラフ (実験 2)

図 14 から次のことが分かる．述語数が 4 の場合では並列照合法を用いた場合の照合回数が一番多い．述語数が 8 を越えると，$Naive$ な方法のほうの照合回数が急に増えて照合回数が 11 万回を越える．次に照合回数が多いのは並列照合法であり，

その次が並列照合法 (改) である．N-OPS を用いた場合の照合回数は 3107〜3182 回であり，照合回数が一番少なくて処理が一番速かったと言える．

4.3 実験 3：最悪照合が生じる場合

今回は，実験 2 の環境で，N-OPS を用いた場合に最悪照合が生じることを示し，N-OPS-Wall を使った場合の評価をする．

もし，実験 2 のパターンに対する N-OPS のデータ構造 H, HF 中に強制的に U を 100%入れると照合回数が指数的に増加し，最悪照合が起こる．

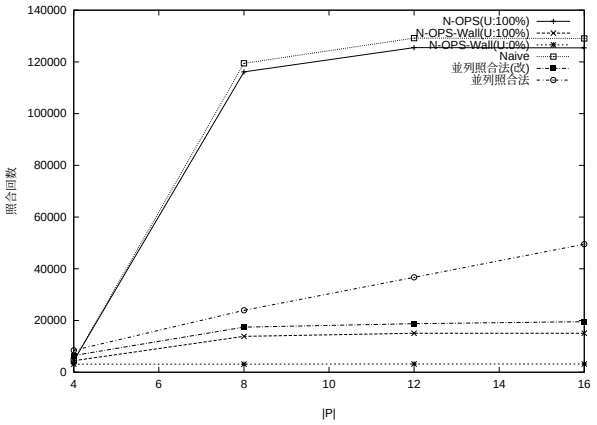

図 15　最悪場合の N-OPS と各方法による照合回数の比較グラフ (実験 3)

この場合，N-OPS-Wall を用いて照合すると，照合回数は並列照合法 (改) と並列照合法を用いた場合の照合回数より少ないように制御できることが分かる (図 15).

5. おわりに

本論文では，データストリーム，あるいはデータベース中に蓄積された系列データから 2 点間文脈に依存した連続系列パターンを探索する際に，文字列照合法の考え方を利用した完備な N-OPS の動作原理を述べ，N-OPS のアルゴリズムを提案し，N-OPS の性能を評価した．本研究を通して，2 点間文脈に依存した連続系列パターンの述語間の論理関係が排他的である場合の計算量が $O(|P|+N)$ であり，$Naive$ な方法，オートマトンの考えを利用した並列照合法，並列照合法 (改) より，処理が速いという結論が得られる．

2 点間文脈に依存した連続系列パターンの述語間の論理関係が非排他的である場合，計算量が排他的な場合より多い．また，データ構造 H, HF からの U の割合が増加すると照合回数が増加し，計算量が $O(|P|+N)$ から指数的に増加して，最悪の照合が行う可能性がある．

この最悪時の照合を回避するために，壁を作り，壁とぶつかるかどうかを判断しながら照合を行う．この最悪場合の対処ができたため，最悪時の計算量を $O(|P| \times N)$ に抑えることができ，$Naive$ な方法，オートマトンの考えを利用した並列照合法，並列照合法 (改) より，処理が速いという結論が得られる．

文　献

[1] B.Babcock, S.Babu, M.Datar, R.Motwani, J.Widom.“ Models and issues in data stream systems ”, in ACM PODS, pp.1-16, 2002.

[2] R.Sadri, C.Zaniolo, A.Zarkesh, J.Adibi.“ Optimization of sequence queries in database systems ”, in ACM PODS, pp.71-81, May 2001.

[3] D.E.Knuth, J.H.Morris, V.R.Pratt.“ Fast pattern matching in strings ”, SIAM Journal of Computing, 6(2):323-350, June 1977.

[4] 大森 匡，七海 嘉仁，星 守,“ データストリームにおける系列パターン探索アルゴリズム N-OPS ”, FIT2003, D-019, 2003 年 9 月.

[5] L.Harada, Y.Hotta, N.Akaboshi, K.Kubota, T.Ohmori, R.Take.“ Event Analyzer: A Tool for Sequential Data Processing ”, in ACM CIKM 2003, pp.172-174, Nov.2003.

[6] D.Terry, D.Goldberg, D.Nichols, B.Oki.“ Continuous queries over append-only databases ”, SIGMOD Record, 1992,21(2):321-330.

[7] R.Avnur, J.Hellerstein.“ Continuously adaptive query processing ”, in ACM SIGMOD, pp.261-272, 2000.

[8] J.Hellerstein, M.Franklin, S.Chandrasekaran, A.Deshpande, K.Hildrum, S.Madden, V.Raman, M.Shah.“ Adaptive query processing: Technology in evolution ”, IEEE Data Engineering Bulletin, 2000,23(2):7-18.

[9] D.Carney, U.Cetinternel, M.Cherniack, C.Convey, S.Lee, G.Seidman, M.Stone braker, N.Tatbul, S.Zdonik.“ Monitoring streams-A new class of DBMS applications ”, Technical Report, CS-02-01, Providence: Departmentof Computer Science, Brown University,2002.

[10] 日経平均株価， http://quote.yahoo.co.jp/