TCS-TR-B-08-5

TCS Technical Report

Master Thesis: Efficient Algorithms for Complex Time-Series Pattern Matching Based on Bit-Parallel Method

by

Tomoya Saito

Division of Computer Science

Report Series B

July 15, 2008

Hokkaido University
Graduate School of
Information Science and Technology

Email: saito@ist.hokudai.ac.jp

Phone: +81-011-706-7680
Fax: +81-011-706-7680

修士論文：ビット並列手法に基づく
高度な時系列パターン照合アルゴリズム

斉藤 智哉
北海道大学 大学院情報科学研究科
コンピュータサイエンス専攻
札幌市北区北 14 条西 9 丁目

July 15, 2008

概要

本論文では，多次元実数ストリームに対する複雑な時系列パターンの照合問題に取り組む．ここで，時系列パターンは幾何的制約の時系列として表現され，ストリーム中でその制約の列が満たされるすべての位置を計算する問題である．幾何的に表現された時系列パターンの高速な照合のために，複数の幾何制約問合せの静的コンパイルとビット並列手法を組み合わせた機構を提案する．主結果として，1 次元整数ストリーム上の時系列パターン照合問題を $O(\frac{1}{w}mn)$ 時間で，d 次元実数ストリーム上の次元間に相関の無い時系列パターン照合問題を $O(\frac{1}{w}mn\log m)$ 時間で，$d=2$ 次元実数ストリーム上の時系列パターン照合問題を $O(\frac{1}{w}mn\log v)$ 時間で解くアルゴリズムをそれぞれ与える．ここで，m, v, n, w はそれぞれ，パターン長と，パターンサイズ，ストリーム長，マシンレジスタ長である．また，計算機実験を行い，既存手法を単純に拡張した場合に比べて 1.5 ～ 5 倍程度高速に動作することを確認した．

1 はじめに

1.1 背景

自動測定技術や高速ネットワーク技術の発展により，センサーデータや，通信記録，半構造データなどのストリームの形をとるデータが増大している．これに伴い，ストリームデータに対する大規模データ処理が重要になっている [2, 4, 10, 11, 19, 20].

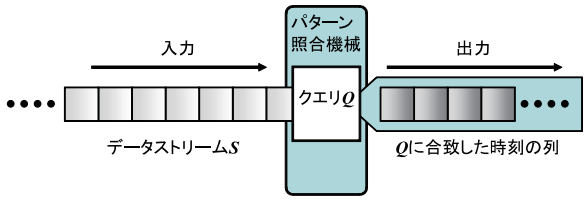

図 1: 連続データストリームに対するクエリ照合

1.2 研究目的

本論文では，連続データストリームに対して時系列パターンによる問合せが与えられる，**プッシュ型ストリーム処理**について考察する．すなわち，クエリが初めに与えられ，それを処理する機構に多次元の数値データが絶え間なく連続的に入力される場合を想定している（図 1）．これは，静的な関係データベースに対してクエリを投げかけ，それに合致するデータを得るプル型のデータ処理とは対照的な処理である．

本論文では，多次元連続データストリームに対する新しいタイプの問合せ処理として，d 次元実数ストリームに対する幾何的制約による時系列質問（geometric time-series query）を提案し，その効率よいパターン照合方法について研究する．すなわち，この照合機械は，ある単位時間毎に d 次元の実数値ベクトルが流れ込むデータストリームに対し，パターンとして与えられた幾何的制約の系列をすべて満たすベクトル列が入力されたか否かを判別する．図 2 に，ストリームの点列が時系列パターンに一致したときの概念図を示す．

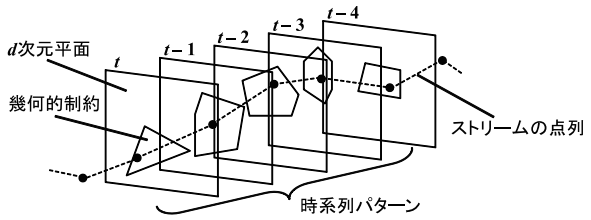

図 2: 幾何的制約の系列による時系列パターン照合

$\pi_1 = q_1 \cdot q_2 \cdot q_3 \cdot q_4 \cdot q_5$ where
$q_1 = (x > 2)$ and $q_2 = (x < 5)$ and
$q_3 = (x > 2 \wedge x < 7)$ and $q_4 = (x < 5) = q_2$ and
$q_5 = (x < 3)$.

図 3: 実数ストリームに対するパターン π_1

1.3 関連研究

Harada [6, 7] や，Sadri-Zaniolo ら [13] は，1 次元の実数データストリームに対して，図 3 のような不等式からなるパターンの照合を行う問題を定義した．また，この問題に対し，よく知られた文字列照合手法である KMP 法 [8, 5, 18] と BM 法 [3, 5, 18] を拡張した高速な時系列パターン照合アルゴリズム L2R と R2L をそれぞれ与えた．これらの手法では，時系列パターンの制約間の包含関係をあらかじめ前処理時に解析することで実行時の高速化を行う．しかし，包含関係が成り立たない制約を多く含む場合や，成立しやすい制約を含む場合，パターン長が短い場合には有効性が低いという問題点がある．

1.4 提案手法

そこで本論文では，別のタイプのよく知られた文字列照合手法であるビット並列パターン照合 [12] を拡張し，d 次元実数変数上の時系列パターンに対する高速なパターン照合アルゴリズム BPS を与える．また，次元間に相関のない制約の場合とある制約の場合について，それぞれ高速なアルゴリズムを与える．

ビット並列パターン照合とは，前処理としてパターンに対応する NFA(非決定性オートマトン) を構築し，実行時にビット演算を用いてその動作を高速に模倣する手法であり，ここではとくに高速かつ実装が容易な技法である Shift-And 手法 [12] の拡張を扱う．しかし，ビット並列手法の単純な適用では自明な手法と同じ $O(mn)$ 時間解法しか得られない．本論文で提案するパターン照合アルゴリズム BPS は，まず前処理時に，パターンに含まれる幾何学質問の集合に対して前処理を行い，実行時に入力となる d 次元実数ベクトル $\vec{x}$ 上で成立するすべての質問からなる部分集合 $P(\vec{x}) = \{\, q \in Q \mid q(\vec{x}) = 1 \,\} \subseteq Q$ を見つける**ヒット表計算**を高速に実行しながら，パターン照合を行う非決定性オートマトンを模倣する．このアルゴリズムは，特に幾何制約条件が軸並行である場合や次元数が 2 の場合において，効率の良い照合を行うことができる．本論文では，多数の幾何学質問に対するヒット表計算の高速な実現方法を与える．

1.5 主結果

本論文では，d 次元実数ストリーム上の時系列パターン照合問題に対し，次のアルゴリズムを与える．

1. 1 次元実数ストリーム上の線形制約のクラスに対するパターン照合問題を $O(1)$ 前処理時間と，$O(1)$ 領域を用いて $O(\frac{1}{w}m^2n)$ 時間で解く素朴なアルゴリズム 1-HT-naive.
2. 1 次元整数ストリーム上の不等式の論理和制約のクラスに対するパターン照合問題を $O(\frac{1}{w}cm^2)$ 前処理時間と，$O(\frac{1}{w}cm)$ 領域を用いて $O(\frac{1}{w}mn)$ 時間で解くアルゴリズム 1-HT-table.
3. 1 次元実数ストリーム上の不等式の論理和制約のクラスに対するパターン照合問題を $O(\frac{1}{w}m^3)$ 前処理時間と，$O(\frac{1}{w}m^2)$ 領域を用いて $O(\frac{1}{w}mn\log m)$ 時間で解くアルゴリズム 1-HT-bin.
4. d 次元実数ストリーム上の次元間に相関のない不等式の論理和制約のクラスに対するパターン照合問題を $O(\frac{1}{w}dm^2)$ 前処理時間と，$O(\frac{1}{w}dm^2)$ 領域を用いて $O(\frac{1}{w}dmn\log m)$ 時間で解くアルゴリズム d-HT-bin.
5. d 次元実数ストリーム上の線形制約のクラスに対するパターン照合問題を $O(1)$ 前処理時間と，$O(1)$ 領域を用いて $O(\frac{1}{w}m^2n)$ 時間で解く素朴なアルゴリズム d-HT-naive.
6. $d = 2$ 次元実数ストリーム上の線形制約のクラスに対するパターン照合問題を $O(\frac{1}{w}m^2)$ 前処理時間と，$O(\frac{1}{w}m^2)$ 領域を用いて $O(\frac{1}{w}mn\log m)$ 時間で解くアルゴリズム d-HT-slab.

7. $d = 2$ 次元実数ストリーム上の線形制約のクラスに対するパターン照合問題を $O(\frac{1}{w}m^2N^2)$ 前処理時間と，$O(\frac{1}{w}m^2N^2)$ 領域を用いて $O(T_\gamma + \frac{1}{w}mn)$ 時間で解くアルゴリズム d-HT-bucket.

ここに，$m = ||\pi||$ は，パターン π のサイズであり，$n = |S|$ はストリーム S の長さ，N はバケットの分割数，T_γ は事後チェックに要する時間である．

また，提案アルゴリズムの有効性を検証するために，人工データ上で評価実験を行った．その結果，パターン長が短い場合やその制約が緩い場合に，BPS は KMP 手法と BM 手法 [12] に基づく既存手法 L2R [7, 13] と R2L [7] と比較して，1.5 倍から 5.0 倍程度高速であり，また安定した性能で動くことがわかった．

ビット並列手法は，ハードウェア化による高速化が容易であり，将来のビット幅の増大が性能向上に結びつくと考えられる．また，正規表現照合も可能であるなどの利点を持つなど，実用面でもストリーム処理のための有用な手法と考えられる．

本論文の構成は以下の通りである．まず 2 章で，準備として用語と問題の定義を行う．3 章では，ビット並列手法を用いた提案手法である BPS アルゴリズムの概要について述べる．4 章では，1 次元実数ストリームの場合のヒット表計算について述べる．5 章では，次元間に相関のない場合の複数次元実数ストリームのヒット表計算について述べる．6 章では，次元間に相関のある場合の複数次元実数ストリームのヒット表計算について述べる．7 章では，実験として，人工データを用いた計算時間の比較実験を行い，提案アルゴリズムの性能を考察する．8 章では，実データへの応用としてモーションデータを用いた検索実験を行い，結果を考察する．

なお，4 章の内容は DEWS'07[17] と SWOD'07[14] で発表済みである．5 章の内容は FIT'07[15] で発表済みであり，6 章の内容は DEWS'08[16] で発表済みである．

2 準備

本章では，研究対象である d **次元実数ストリーム上の幾何時系列パターン照合問題**を定式化する．さらに，幾何制約の種々のクラスとこれらに基づく種々の幾何時系列パターンのクラスを導入する．また，必要な用語を導入する．

2.1 ビット演算

本論文では，計算モデルとして，以下の数値演算とビット演算をもつ RAM 機械を仮定する [1, 12]．整数レジスタの**ビット幅**を w と仮定する．長さ $m > 0$ の**ビットマスク**は，最下位ビットを右側としたビット列 $b_{m-1} \cdots b_0 \in \{0,1\}^m$ で表す．長さ $m < w$ のときは上位の残りビットに 0 を詰めることとする．**ビットごとの論理和と，論理積，否定**をそれぞれ記号 | と，&，$\sim$ で表す．非負整数 $0 \leq n \leq m-1$ に対して，ビットマスク X の n ビットの**左シフト**を $X << n$ で表す．0^m は 0 **ビットのみ**のビットマスクであり，$1 << i = 0^{m-i}10^{i-1}$ は i **番目のビットが 1** のビットマスクである．整数集合 $S \subseteq \{0, \ldots, m-1\}$ に対して，$Bit(S) \in \{0,1\}^m$ で S のビットマスク表現を表す．

例 1 ビット幅を $w = 8$ とする．$X = 00101101, Y = 10110111$ と仮定する．このとき，次にビット演算の例を示す．

1. ビットごとの論理和: $X \mid Y = 10111111$.
2. ビットごとの論理積: $X \ \& \ Y = 00100101$.
3. ビットごとの否定: $\sim X = 11010010$.
4. 1 ビットの左シフト: $X << 1 = 01011010$.
5. 0 のみのビットマスク: $0^8 = 00000000$.
6. 最下位に 1 の立ったビットマスク: $1 = 0000001$.
7. 上のビットマスクを左に 4 シフトして得られるビットマスク：$1 << 4 = 00010000$.
8. $S = \{0, 2, 3, 5\}$ のビットマスク表現: $Bit(S) = 00101101$.

現代の実用的な計算機では，通常 $w = 32$ または $w = 64$ である．ただし，本稿では w は定数でなく，任意に大きい値をとりうると考える．ここでは，上記すべての数値演算とビット演算は，ビット幅 w の整数レジスタ上で $O(1)$ 時間で実行できると仮定する．これを複数回繰り返すことで，長さ $m > w$ のビットマスクに対しては，これらの演算は $O(\lceil \frac{m}{w} \rceil)$ 時間で実行できる [1, 12].

2.2 d 次元実数ストリーム上の幾何時系列パターン照合問題

$\mathbb{N} = \{0, 1, 2, \ldots\}$ と $\mathbb{R}$ でそれぞれ非負整数全体と実数全体を表す．A^* で集合 A の反射的推移閉包 [9] を表す．正整数 d に対して，$\mathbb{R}^d$ を d 次元実数空間とする．$\mathbb{R}^d$ の**点**または**レコード**は，ベクトル $\vec{s} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ である．変数 x のとる値域を $dom(x)$ で表す．特に指定しない場合，$dom(x) = \mathbb{R}$ と仮定する．d 次元実数ストリームは，d 次元の点列 $S = (\vec{s}_1, \ldots, \vec{s}_n) \in (\mathbb{R}^d)^*$ である．

例 2 $S_1 = (1, 5, 3, 5, 4, 2, 4, 1, 2, 2) \in (\mathbb{R}^1)^*$ は長さ 10 の 1 次元実数ストリームである．

例 3 $S_2 = ((0,0), (1,1), (2,0), (-1,-1), (0,-1), (-1,0)) \in (\mathbb{R}^2)^*$ は，長さ 6 の 2 次元実数ストリームである．

次のようなデータは d 次元実数ストリームとして扱うことができる．

1. 交通データのレコードは，2 次元ベクトル $\vec{p} = (x, y)$ である．ここに，x と y は移動点の緯度と経度である．
2. 蛋白質データのレコードは，ベクトル $\vec{p} = (x, y, z, info)$ である．ここに，x, y, z は原子の 3 次元空間座標であり，$info$ は原子の種類や電子価などの付加情報である．
3. センサーデータのレコードは，その測定値のベクトル $\vec{p} = (x_1, \ldots, x_d)$ で表される．
4. モーションデータのレコードは，$\vec{p} = (x_1, y_1, z_1, \theta_1, \phi_1, \psi_1, \ldots, x_{19}, y_{19}, z_{19}, \theta_{19}, \phi_{19}, \psi_{19})$ である．ここに，モデルは 19 個の関節を持つと仮定し，i 番目の関節に対して $x_i, y_i, z_i \in \mathbb{R}$ は 3 次元空間における関節の絶対座標を表し，$\theta_i, \phi_i, \psi_i \in (-\pi, +\pi]$ は回転角を表す．

2.3 d 次元実数空間上の幾何制約

任意の $d > 0$ に対して，d 次元実数空間 $\mathbb{R}^d$ 上の二値関数 $q : \mathbb{R}^d \to \{0, 1\}$ を，d 次元の **幾何制約（制約）** という．文脈から明らかなとき，幾何制約を $\mathbb{R}^d$ の部分集合 $\{ \vec{s} \in \mathbb{R}^d \mid q(\vec{s}) = 1 \} \subseteq \mathbb{R}^d$ と同一視する．制約 q の表現長を $||q||$ で表す．d 次元実数空間上の幾何制約のクラスを $\mathcal{C}^d_{geo}$ で表す．

$\mathcal{C}$ を幾何制約の任意のクラスとする．$\mathcal{C}$ 上の幾何制約の集合 $P = \{q_1, \ldots, q_m\}$ と d 次元の点 $\vec{s}$ に対して，$\vec{s}$ における P の**ヒット表**とは，$\vec{s}$ によって満たされる P 中の制約の集合 $P(\vec{s}) = \{ q \in P \mid q(\vec{s}) = 1 \}$ である．ここで，各制約 q_i は，添字 i で表すと仮定する．

定義 1 $\mathcal{C}$ 上の**ヒット表計算問題**とは，あらかじめ $\mathcal{C}$ 上の幾何制約の集合 $P = \{q_1, \ldots, q_m\}$ を受け取って前処理を行い，問い合わせとして d 次元の点 $\vec{s}$ を受け取り，出力として，P の $\vec{s}$ におけるヒット表 $P(\vec{s})$ を計算する問題である．

本稿では，ヒット表は長さ m のビットマスク $Bit(P(\vec{s})) \in \{0, 1\}^m$ として返されると仮定する．

定義 2 $\mathcal{C}$ 上の長さ m の**幾何時系列パターン**π は，列 $\pi = (q_1, \ldots, q_m) \in \mathcal{C}^*$ である．ここに，$1 \leq i \leq m$ に対して，$q_i \in \mathcal{C}$ は $\mathcal{C}$ 上の長さ m の幾何制約である．

上記のパターン π の長さを $|\pi| = m$ で定義し，π のサイズを $||\pi|| = \sum_{i=1}^{m} ||q_i||$ で定義する．

長さ m のパターン $\pi = (q_1, \ldots, q_m)$ が，長さ n の d 次元実数ストリーム $S = (\vec{s}_1, \ldots, \vec{s}_n) \in (\mathbb{R}^d)^*$ において**位置** $1 \leq j \leq n$ **に出現する**とは，$q_1(\vec{s}_j) \wedge \cdots \wedge q_m(\vec{s}_{j+m-1})$ が成立することをいう．

定義 3 パターンのクラス $\mathcal{C}$ に対する d **次元実数ストリーム上の幾何時系列パターン照合問題**（パターン照合問題）は，あらかじめ $\mathcal{C}$ 上の幾何時系列パターン $\pi \in \mathcal{C}$ を受け取って前処理をし，入力として d 次元実数ストリーム $S = (\vec{s}_1, \ldots, \vec{s}_n) \in (\mathbb{R}^d)^*$ を受け取り，出力として S 上の π の全ての出現位置を計算する問題である．

ここでの目的は，パラメータ $m = |\pi|$ と，$v = ||\pi||$，$n = |S|$ に対して前処理時間と，使用領域，入力ストリームにおける照合時間を小さくするアルゴリズムを設計することである．

この問題を解くアルゴリズムの計算時間の下限は，明らかに $\Omega(v + n)$ 時間である．また，実際のストリーム監視等の応用を考えたとき，パターン照合アルゴリズムは，新しく到着する度に漸増的に動く**オンライン性**をもつことが望ましい．本論文で与えるすべてのアルゴリズムは，オンライン性をもつアルゴリズムである．

2.4 幾何制約のクラス

本節では，本論文で扱う幾何制約のクラスを与える．d 次元実数空間 $\mathbb{R}^d$ において，変数集合 $(x_1, \ldots, x_d)$ を考える．点 $p = (c_1, \ldots, c_d) \in \mathbb{R}^d$ において，i 番目の変数 x_i は値 c_i をとるとする．**不等式**(atomic inequality) とは，表現 $\ell = (ax \leq c)$ であり，ここに，$a \in \{+1, -1\}$ は符号であり, $c \in \mathbb{R}$ は定数である．**線形不等式**(linear inequality) は，式 $\ell \equiv (a_1x_1 + \cdots + a_dx_d \leq c)$ である．ここに，$a_1, \ldots, a_d, c \in \mathbb{R}$ は定数である．

2.4.1 1 次元実数空間上の幾何制約のクラス

整数ストリーム上の不等式の論理積のクラス． 1 次元整数ストリーム上の不等式の論理積のクラス $\mathcal{C}^1_{ieq,int}$ を次のように定義する．$\mathcal{C}^1_{ieq,int}$ 上の幾何制約 q は，不等式の論理積 $q = \ell_1 \wedge \cdots \wedge \ell_k$ である．ここに，$(1 \leq i \leq k)$ に対して，$\ell_i = (a_i x \leq c_i)$ であり，$a_i \in \{+1, -1\}$, $c_i \in \mathbb{R}$ である．$dom(x) = \mathbb{N}$ である．

実数ストリーム上の不等式の論理積のクラス． 1 次元実数ストリーム上の不等式の論理積のクラス $\mathcal{C}^1_{ieq,real}$ を次のように定義する．$\mathcal{C}^1_{ieq,real}$ 上の幾何制約 q は，不等式の論理積 $q = \ell_1 \wedge \cdots \wedge \ell_k$ である．ここに，$(1 \leq i \leq k)$ に対して，$\ell_i = (a_i x \leq c_i)$ であり，$a_i \in \{+1, -1\}$, $c_i \in \mathbb{R}$ である．$dom(x) = \mathbb{R}$ である．

補題 1 $\mathcal{C}^1_{ieq,int}$ および $\mathcal{C}^1_{ieq,real}$ 上のすべての幾何制約は，標準形として変数 x に関する 2 つの不等式の論理積 $q = (a_1x \leq c_1) \wedge (a_2x \leq c_2)$ として表すことができる．

等式制約 $x = c$ も，2 つの不等式の論理積 $(x \leq c) \wedge (x \geq c)$ で表すことができる．本論文では，$\mathcal{C}^1_{ieq,int}$ および $\mathcal{C}^1_{ieq,real}$ 上のすべての幾何制約は標準形であると暗黙のうちに仮定する．

2.4.2 d 次元実数空間上の幾何制約のクラス：次元間に相関のない場合

d 次元実数空間上の不等式の論理積のクラス． d 次元実数空間上の不等式の論理積のクラス $\mathcal{C}^d_{ieq,real}$ を次のように定義する．$\mathcal{C}^d_{ieq,real}$ 上の幾何制約 q は，不等式の論理積

$$q = \ell^1_1 \wedge \cdots \wedge \ell^1_{k(d)} \wedge \cdots \wedge \ell^d_1 \wedge \cdots \wedge \ell^d_{k(d)}$$

である．ここで，各 $1 \leq i \leq d, 1 \leq j \leq k(d)$ に対して，ℓ^i_j は i 番目の変数 x_i に関する制約 $\ell^i_j = (a^i_j x_i \leq c^i_j)$ である．ここに，$a^i_j \in \{+1, -1\}$ であり，$c^i_j \in \mathbb{R}$, $dom(x_i) = \mathbb{R}$ である．

常に成立する線形制約を空の論理積または空集合 $\emptyset$ で表す．

2.4.3 d 次元実数空間上の幾何制約のクラス：次元間に相関のある場合

d 次元の実数空間上の線形制約のクラス． d 次元実数空間上の線形制約のクラス $\mathcal{C}^d_{lin}$ を次のように定義する．$\mathcal{C}^d_{linear}$ 上の線形制約 q は，d 次元線形不等式の論理積

$$q = \ell_1 \wedge \cdots \wedge \ell_k \ (k \leq 0)$$

である．ここに，$\ell = (a_1x_1 + \cdots + a_dx_d \leq c)$ は，d 次元線形不等式であり，$a_1, \cdots, a_d, c \in \mathbb{R}$，$dom(x_i)\mathbb{R}$ である．

例 4 線形制約の特別な場合として，次のような制約が表現可能である．

- **軸並行直方体**. $box(a_1, b_1, a_2, b_2) = [a_1, b_1] \times [a_2, b_2] = \{ (x_1, x_2) \,|\, a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2 \}$.
- **半平面**. $HS(a_1, \ldots, a_d, c) = \{ x_1 \ldots x_d \,|\, a_1x_1 + \cdots + a_dx_d \leq c \}$.
- **凸多角形**. 点列 $P = (\vec{s}_1, \ldots, \vec{s}_k)$ で表される凸包 $conv(P) = \{\alpha_1\vec{s}_1 + \ldots + \alpha_k\vec{s}_k \,|\, \alpha_1 + \ldots + \alpha_k = 1, \alpha_i$は非負実数 $\}$.

2.4.4 各クラス間の関係

次の包含関係が成立する．すなわち，$\mathcal{C}^1_{ieq,int} \subseteq \mathcal{C}^1_{ieq,real}$ かつ $\mathcal{C}^d_{ieq,real} \subseteq \mathcal{C}^d_{lin} \subseteq \mathcal{C}^d_{geo}$ である．

3 ビット並列手法に基づく幾何時系列パターン照合アルゴリズムの概要

本節では，d 次元の線形制約の列である時系列幾何パターン $\pi = (q_1, \ldots, q_m)$ に対して，ビット並列手法を用いて d 次元ストリーム上の時系列幾何パターン照合を効率よく行うアルゴリズム BPS を与える．

ビット並列手法は，整数レジスタ上の数値演算を用いて，ビット列上の演算を並列に実行する手法である [12]．ここでは，ビット並列手法を用いた効率良い部分文字列パターン照合手法の中で，実装が容易で高速な Shift-And 技法 [12] を採用する．

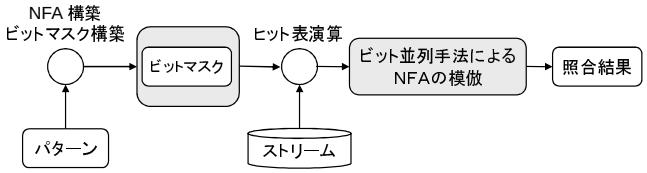

図 4: ビット並列パターン照合の概略

3.1 手法の概要

図 4 に，本論文の提案アルゴリズム BPS の構成を示す．BPS では，次の 2 つのフェイズに分けてパターン照合を行う．まず時系列パターンの静的コンパイル処理として，与えられた時系列幾何パターン π を解析し，含まれるすべての時点幾何パターンを抽出する．時点幾何パターンのクラスにしたがって，ヒット表計算のための前処理を行う．次に，ビット並列手法に基づく実行時の動的パターン照合処理として，実行時に入力ストリームから入力点を受け取りながら，ヒット表計算手続きを用いて時系列幾何パターン π の照合を行う非決定性有限オートマトン (NFA) $\mathcal{M}$ を高速に模倣する．

3.2 時系列パターンの静的コンパイル処理

3.2.1 NFA の構築 $\mathcal{C}$ を幾何制約のクラスとする．初めに最も単純な正規表現の族として，直線状の連続部分列からなるパターン $\pi = (q_1, \cdots, q_m) \in \mathcal{C}^*$ の NFA のクラスを考える．このクラスの NFA は一見単純に見えるが，部分文字列パターンだけでなく，常に真となる空制約 $q_i = \mathsf{T}$ や選言を用いることで，文字列照合におけるワイルドカードや集合文字に対応したパターン照合 [12] を表現可能であることに注意されたい．

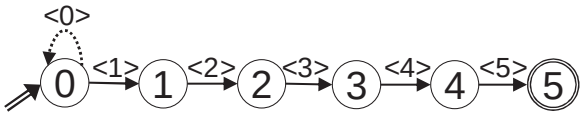

図 5: 長さ 5 の時系列幾何パターンに対する NFA $\mathcal{M}$

与えられた長さ $m \geq 0$ の時系列幾何パターン $\pi = (q_1, \cdots, q_m)$ に対する **NFA** $\mathcal{M} = (Q_\pi, \Sigma_\pi, \delta_\pi, I_\pi, F_\pi)$ は，図 5 のような 0 から m までの全ての状態を直線状につないだ NFA である．状態 0 は $\langle 0 \rangle$ で自己に遷移する自己ループをもつ．ここで，各遷移の有向辺のラベル $\langle i \rangle$ は，π の i 番目の幾何制約 q_i を表す．ここに，$q_0 = \mathsf{T}$ である．

- Q_π は，状態集合 $\{0, 1, \ldots, m\}$ である．
- Σ_π は，文字集合 $\{\langle 0 \rangle, \langle 1 \rangle, \ldots, \langle m \rangle\}$ である．
- $\delta_\pi \subseteq Q_\pi \times \Sigma_\pi \times Q_\pi$ は遷移関係 $\delta_\pi = \{(i-1, \langle i \rangle, i) \mid i = 1, \ldots, m\} \cup \{(0, \langle 0 \rangle, 0)\}$ である．
- I_π は初期状態 $I_\pi = \langle 0 \rangle$ である．
- F_π は受理状態集合 $F_\pi = \{\langle m \rangle\}$ である．

3.2.2 ヒット表計算 NFA の遷移を行うためには，入力点 $\vec{s}$ によって遷移する辺を決定する必要がある．パターン $\pi = (q_1, \ldots, q_m)$ に対して，それが含むの幾何制約すべての集合を $P = \{q_1, \ldots, q_w\}$ とおく．このとき，ヒット表 $P(\vec{s})$ を計算すればよい．

最も単純なヒット表計算の方法は，実行時にストリームの点 $\vec{s}_t$ を受け取ったときに，m 個の制約 $q_1, \ldots, q_m$ 全てに対して真理値検査 "$q_i(\vec{s}_t) = 1$?" を行うことである．しかし，ひとつの制約 q の評価に $O(||q||)$ 時間を要すると仮定すると，この方法ではヒット表計算に $O(v) = O(||\pi||)$ 時間を要する．そのため，総計算時間は $O(vn)$ 時間になってしまう．そこで，次章以降では，種々の幾何制約のクラスに対する高速なヒット表計算の実現方法を与える．

3.3 実行処理

BPS は実行時には，NFA $\mathcal{M}$ は初期状態 0 から出発し，時点 t で点 $\vec{s}_t$ 上で真となる時点幾何パターン $\langle i \rangle = q_i$ をラベルとする遷移を非決定的に実行することを繰り返し，最終状態 m で出現位置 $t - m + 1$ を出力して，照合を模倣する．図 6 に入力 d 次元ストリーム S 上でパターンを照合する NFA を模倣する手続き ScanStream を示す．このアルゴリズムは，状態集合 $D \subseteq \{1, \ldots, m\}$ を長さ m のビットマスク $D \in \{0, 1\}^m$ で表現し，ヒット表計算によって得られるヒット集合のビットマスク表現 B を用いたビット並列計算で高速に照合を行う．図 6 の状態遷移にあたる行は $O(\frac{m}{w})$ 時間で処理できる．特にパターン長が短い（$m \leq w$）とき，$O(1)$ 時間で NFA の遷移を行うことができる．

計算モデルとして，レジスタのビット幅 w で，数値演算と論理演算を定数時間で実行する RAM [12] を考える．このとき，手続き ScanStream の実行時間として次の補題が成り立つ．

補題 2 図 6 の手続き ScanStream は，時系列幾何パターン π に対して，d 次元ストリーム上の時系列幾何パターン照合問題を $O((T_{\mathcal{H}} + O(\frac{m}{w})) \cdot n)$ 時間で解く．ここで，$T_{\mathcal{H}}$ はヒット表計算に要する時間である．

procedure ScanStream(S)
入力：d 次元ストリーム $S = \vec{s}_1 \cdots \vec{s}_n$.
出力：パターン $\pi = q_1 \cdots q_m$ の S 上の全ての出現.

1: $I \leftarrow 0^{m-1}1$;
2: $F \leftarrow 10^{m-1}$;
3: $D \leftarrow I$;
4: **for** $t \leftarrow 1, \ldots, n$ **do**
5: ヒット表計算：$\vec{s_t}$ に対するビットマスク B を計算する.
6: $D \leftarrow ((D << 1) \mid I)$ & B;
7: **if** $(D$ & $F) \neq 0^m$ **then** 出現 $t - m + 1$ を出力する;
8: **end for**

図 6: NFA の模倣手続き ScanStream

3.4 複数パターンのクラス

前節の直線状の NFA のクラスに対するビット並列化を，複数パターン照合問題に拡張できる．パターン集合 $P = \{\pi_1, \ldots, \pi_k\}$ に対して，各パターン π_i のマスク B_i, I_i, F_i, D_i を直列に連結したマスク B', I', F', D' を用いて，各パターン π_i の NFA の動作の模倣をビット演算により一括して行うことができる．詳細については [12] を参照のこと.

3.5 幾何時系列パターンの正規表現への拡張

$\mathcal{C}$ を幾何制約の任意のクラスとする．有限可変長ギャップ $[low, high]$ や，省略可能幾何制約 ?，繰り返し $*$ などの演算をもつ複雑なパターンは，一般に**正規表現**[9] の部分クラスとして定義できる．$\mathcal{C}$ 上の正規表現のクラス $\mathcal{RESG}_{\mathcal{C}}$ を次のように帰納的に定義する.

(i) 線形制約 $q \in \mathcal{C}$ に対して，$\varepsilon, q \in \mathcal{RESG}_{\mathcal{C}}$ である.

(ii) もし $P, Q \in \mathcal{RESG}_{\mathcal{C}}$ ならば，$P \cdot Q, (P|Q), (P)^* \in \mathcal{RESG}_{\mathcal{C}}$ である.

(iii) 以上で生成されるものだけが $\mathcal{RESG}_{\mathcal{C}}$ の要素である.

定義 4 $\mathcal{C}$ 上の**正規表現幾何時系列パターン（正規表現パターン）**は，正規表現 $\pi \in \mathcal{RESG}_{\mathcal{C}}$ である.

また，正規表現 π の表す幾何時系列パターン $\mathcal{E}(\pi)$ を π の言語 [9] として次のように帰納的に定義する.

(i) 幾何制約 $q \in \mathcal{C}$ に対して，$\mathcal{E}(\varepsilon) = \{\varepsilon\}$ かつ $\mathcal{E}(q) = \{q\}$ である.

(ii) もし $P, Q \in \mathcal{RESG}_{\mathcal{C}}$ ならば，$\mathcal{E}(P \cdot Q) = \mathcal{E}(P) \cdot \mathcal{E}(Q)$, $\mathcal{E}(P|Q) = \mathcal{E}(P) \cup \mathcal{E}(Q)$, $\mathcal{E}((P)^*) = \mathcal{E}(P)^*$ である.

(iii) 以上で生成されるものだけが $\mathcal{E}(\pi)$ の要素である.

定義 5 言語 $\mathcal{E}(\pi)$ を正規表現パターン π の**展開**と呼び，その要素 $Q \in \mathcal{E}(\pi)$ を**展開例**と呼ぶ．展開例は $\mathcal{C}$ 上の時系列パターンである.

例 5 1 節の図 3 のパターン π_1 は，1 次元の長さ 5 の単純パターンの例である.

定義 6 正規表現パターン π がストリーム S において**位置** $1 \leq j \leq n$ **に出現する**とは，π のある展開例である幾何時系列パターン $Q \in \mathcal{E}(\pi)$ が S において位置 j に出現することをいう.

例 6 図 3 のパターン π_1 は，例 2 のストリーム S_1 に位置 1 と位置 4 で出現する.

ビット並列手法による NFA の模倣は，ヒット表計算の実現とは独立であり，NFA のクラスを拡張することで，以下に示すようなさまざまなクラスの正規表現パターンを適用可能である.

有限可変長ギャップをもつパターンは，$\pi_2 = q_1 q_2 [1..3] q_3$ のような制限された形の正規表現パターンである．ここに，有限ギャップ $q[i..j]$ は線形制約 q を満たす任意のレコードの i 個から j 個の連続を表す．これは，図 7 のように，ギャップを最大限に展開した単純パターン $\pi_2' = q_1 q_2 q_2 q_2 q_3$ の NFA に ε 遷移 $(k, \langle 0 \rangle, l)$ を追加した NFA で模倣できる．ここに，k は q_2 の位置（=2），$k - 1 \leq l \leq k + j - i - 1$ とする．ビット演算による遷移計算では，通常の状態 D の遷移の直後に，i と $j - 1$ にだけビット 1 があるマスク GI と GF を，数値減算 $-$ で組み合わせて次のビット並列演算を行い，GI のビットからのすべての ε 遷移を正しく模倣する.

$$D \leftarrow D \mid ((GF - (D \ \& \ GI)) \ \& \ \sim GF);$$

これは，例えば，$1000 - 0001 = 0111$ のように，j 番目のビットのみが 1 のマスクから，$i(< j)$ 番目のビットが 1 のマスクを算術減算すると，i から $j - 1$ まで 1 が伝播することを利用している.

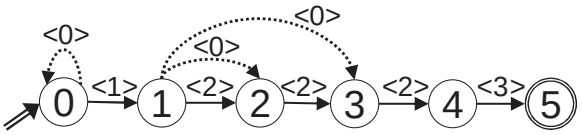

図 7: 有限可変長ギャップをもつパターン π_2 に対する NFA

さらにこの減算を用いた ε 遷移の実現技法を用いて，$\pi_3 = q_1(q_2)?q_3$ のような**省略可能幾何制約をもつパターン**や，$\pi_4 = q_1(q_2)^*q_3$ のような**幾何制約の繰り返しパターン**を，ビット並列手法で扱うことができる．模倣技法の詳細については，[12] を参照のこと．

4 1次元ストリームに対する効率よいヒット表計算

本節では，1次元実数ストリーム上の時系列パターン照合問題を考察する．3.2.2 節の $O(m)$ 時間のヒット表計算を用いた素朴な時系列パターン照合アルゴリズムでは，総計算時間が $O(mn)$ 時間になる．そのため，本節では 1 次元実数ストリーム上の不等式の論理積で表される幾何制約のクラス $\mathcal{C}^1_{ieq,real}$ に対して、二分探索を用いた $O(\frac{1}{w}m\log m)$ 時間のヒット表計算アルゴリズムを与える．さらに、1 次元実数ストリーム上のクラス $\mathcal{C}^1_{ieq,int}$ に対して、表一瞥方式を用いた $O(\frac{1}{w}m)$ 時間のヒット表計算アルゴリズムを与える．

4.1 整数ストリームに対するヒット表の構築

図 17 に，クラス $\mathcal{C}^1_{ieq,int}$ に対する，1 次元整数ストリーム上のヒット表計算アルゴリズム 1-HT-table を与える．

前処理では，1 次元整数ストリームに対して，とりうる全ての値 $\vec{s}$ に対して，あらかじめヒット表 $P(\vec{s})$ を計算しておく．具体的には，値の集合 $\Delta = \{1..c\}$ 上の全ての値 $\vec{s} \in \{1..c\}$ に対して，あらかじめヒット集合 $P(\vec{s}) \subseteq \{1,\ldots,m\}$ を表すビットマスク $Bit(P(\vec{s})) \in \{0,1\}^m$ を計算し，ビットマスク配列 $B[1..c] = (B[1],\ldots,B[c]) \in (\{0,1\}^m)^c$ に格納しておく．実行時には，配列の添字によるランダムアクセスによって，ヒット表を表すビットマスクを取り出す．

表 1: 整数ストリームに対するヒット表

$\vec{s}$	$P(\vec{s})$	$Bit(P(\vec{s}))$
8	1	00001
7	1	00001
6	13	00101
5	13	00101
4	1234	01111
3	1234	01111
2	245	11010
1	245	11010

例 7 次の不等式制約集合 P_1 を考える．

$$q_1 = (x > 2)$$
$$q_2 = (x < 5)$$
$$q_3 = (x > 2 \wedge x < 7)$$
$$q_4 = (x < 5)$$
$$q_5 = (x < 3)$$

このとき，表 1 に，値の集合 $\Delta = \{1,\ldots,8\}$ に対するヒット表をあらかじめ計算したものを示す．各欄は順に，値 $\vec{s}$ と，$\vec{s}$ が満たすヒット集合 $P(\vec{s})$，そのビットマスク表現 $Bit(P(\vec{s}))$ である．ただし，簡単のため $\{\langle 1\rangle,\langle 2\rangle,\langle 3\rangle\}$ を 123 のように表記する．

定理 1 手続き 1-HT-table は，幾何制約のクラス $\mathcal{C}^1_{ieq,int}$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(\frac{1}{w}cm^2)$ と領域 $O(\frac{1}{w}cm)$ を用いて，各入力点 $p \in \mathbb{N}1$ 個あたり，$O(\frac{1}{w}m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズである．

補題 2 から，次の系がただちに成立する．

系 1 幾何制約のクラス $\mathcal{C}^1_{ieq,int}$ 上に対して，d 次元実数ストリーム上の幾何時系列パターン照合問題は，前処理時間 $O(\frac{1}{w}cm^2)$ と領域 $O(\frac{1}{w}cm)$ を用いて，$O(\frac{1}{w}mn)$ 時間で解ける．$m = ||\pi||$ は，パターン π のサイズであり，$n = |S|$ はストリーム S の長さである．

Global：ビットマスク配列 $(B[1], \ldots, B[c]) \in (\{0,1\}^m)^c$

//前処理手続き
Build(Δ, P)
入力：制約集合 P, 値の集合 Δ.
出力：ビットマスク配列の計算.

1: **for** $x \in \{1, \cdots, c\}$ **do**
2: $B[x] \leftarrow 0^m$;
3: **for** $i \in \{1, \cdots, m-1\}$ **do**
4: **if** $q_i(x) = 1$ **then** $B[x] \leftarrow B[x] \mid 1 << i$;
5: **end for**
6: **end for**

//実行時手続き
LookUp$(\vec{s})$
入力：$\Delta = \{1, \cdots, c\}$ 上の点 $\vec{s} = x \in \mathbb{N}$.
出力：入力点 $\vec{s} = x$ に対するビットマスク $B[x]$.
return $B[x] \in \{0,1\}^m$;

図 8: 整数ストリームに対するヒット表計算アルゴリズム 1-HT-table

4.2 実数ストリームに対するヒット表の構築

図 9 に，クラス $\mathcal{C}^1_{ieq,real}$ に対する，1 次元実数ストリーム上のヒット表計算アルゴリズム 1-HT-bin を与える．実数ストリームのとりうる値は無限集合であるため先の手法が使えない．そこで，次の方法を用いる．

初めに，先の図 10 のように，π が含む総計 $||\pi||$ 個の線形不等式 $\ell_i = (x \leq c)$ の端点 c をすべて数直線 A 上にプロットする．このとき，次が成立する．

補題 3 任意の線形制約 q を充足する値の集合は A 上の端点と極小開区間の集合和として一意に表現できる．

(証明) A 上の端点と極小開区間の集合和のうち，どの区間も，その区間内で満足される制約は同じである．よって補題が成り立つ． □

これより，節 4.1 がすべての値に対するビットマスクを計算したのと同様に，あらかじめ A 上のすべての端点と極小開区間に対してビットマスクを計算し，表 2 のようなヒット表を構築する．実行時に，入力点 $\vec{s} \in \mathbb{R}$ が含まれる区間をこの表から探索することで，$\vec{s}$ に対応するビットマスク $Bit(P(\vec{s}))$ を計算できる．

Global：ビットマスク配列 $(B[1], \ldots, B[c]) \in (\{0,1\}^m)^c$ [6pt]

//前処理手続き
Build(P)
入力：制約集合 P.
出力：ビットマスク配列の計算.

1: P のすべての線形不等式の端点を，数直線 A 上にプロットする．
2: A 上のすべての端点と極小開区間に対してビットマスクを計算し，表 2 のようなヒット表の配列を構築する．この配列の要素は，端点の昇順にソートする．

//実行時手続き
LookUp$(\vec{s})$
入力：1 次元実数空間上上の点 $\vec{s} = x \in \mathbb{R}$.
出力：入力点 $\vec{s} = x$ に対するビットマスク $B[x]$.

1: 入力点 $\vec{s}$ を含む区間を配列から二分探索し，対応するビットマスク B を見つける．
2: **return** $B \in \{0,1\}^m$;

図 9: 実数ストリームに対するヒット表計算アルゴリズム 1-HT-bin

例 8 例 7 の不等式制約集合 P_1 に対して，図 10 に，P_1 のすべての不等式制約をプロットしたものを示す．この図では，数直線 A がすべての端点と極小開区間に分割されていることがわかる．表 2 に，すべての端点と極小開区間に対するヒット表をあらかじめ計算したものを示す．各欄は順に，端点または極小開区間の値 $\vec{s}$ と，$\vec{s}$ が満たすヒット集合 $P(\vec{s})$，そのビットマスク表現 $Bit(P(\vec{s}))$ である．ただし，簡単のため $\{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$ を 123 のように表記する．

A は高々$||\pi||$ 個の端点と極小開区間しか持たない

表 2: 実数ストリームに対するヒット表

$\vec{s}$	$P(\vec{s})$	$Bit(P(\vec{s}))$
$\vec{s} > 7$	1	00001
$\vec{s} = 7$	1	00001
$5 < \vec{s} < 7$	13	00101
$\vec{s} = 5$	13	00101
$3 < \vec{s} < 5$	1234	01111
$\vec{s} = 3$	1234	01111
$2 < \vec{s} < 3$	12345	11111
$\vec{s} = 2$	245	11010
$\vec{s} < 2$	245	11010

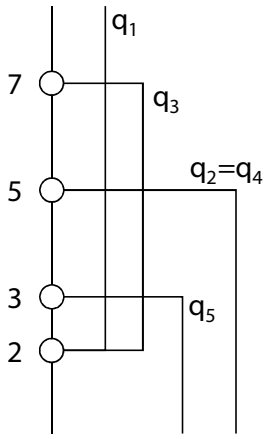

図 10: 制約集合 P_1 の数直線上の表現

ので，節 4.1 と同様の議論で，計算量は次のようになる．

定理 2 手続き 1-HT-bin は，幾何制約のクラス $\mathcal{C}^1_{ieq,real}$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(\frac{1}{w}m^3)$ と領域 $O(\frac{1}{w}m^2)$ を用いて，各入力点 $p \in \mathbb{R}1$ 個あたり，$O(\frac{1}{w}m\log m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズである．

補題 2 から，次の系がただちに成立する．

系 2 幾何制約のクラス $\mathcal{C}^1_{ieq,real}$ 上に対して，d 次元実数ストリーム上の幾何時系列パターン照合問題は，前処理時間 $O(\frac{1}{w}m^3)$ と領域 $O(\frac{1}{w}m^2)$ を用いて，$O(\frac{1}{w}mn\log m)$ 時間で解ける．$m = ||\pi||$ は，パターン π のサイズであり，$n = |S|$ はストリーム S の長さである．

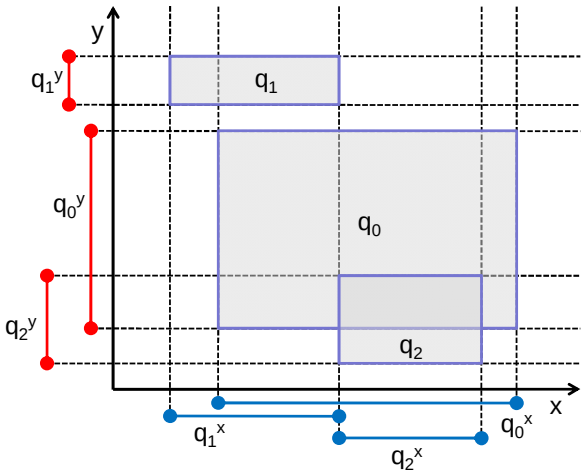

図 11: 軸平行な幾何制約と x 軸と y 軸に分離された制約

5 多次元ストリームに対する効率よいヒット表計算：次元間に相関のない幾何制約の場合

本章では，ストリームを d 次元に拡張し，d 次元実数ストリーム上の時系列パターン照合問題を考察する．クラス $\mathcal{C}^d_{ieq,real}$ 上の，パターンが次元間に相関のない制約からなる場合のヒット表計算アルゴリズムを与える．説明の簡便のため，入力点の空間として 2 次元実数空間 $\mathbb{R}^2$ を考える．

前章の手法を d 次元ストリームに単純に適用すると，d に関して指数サイズの領域のビットマスクを要する．そこで，次元間に相関がない制約に特化した手法を与える．

5.1 変数分離形のパターン

制約 q 中の線形不等式 $\ell_i (1 \le i \le |q|)$ がすべて各次元間に相関のない制約 $\ell_i = (a_i x_j \le c_i)$ であるとき，**変数分離形**であるという．ここに，$1 \le j \le d$ である．即ち，$|\ell_i| = 1$ であり，$d = 2$ 次元空間上では，図 11 のように x 軸または y 軸に平行な直線からなる制約である．変数分離形の制約のみからなるパターンを変数分離形のパターンという．

5.2 変数分離形のパターンに対するヒット表計算

図 12 に，変数分離形のヒット表計算アルゴリズム d-HT-bin を与える．幾何制約集合で用いられる線

procedure d-HT-bin($\vec{s} = (x, y) \in \mathbb{R}^2$)

1: 端点のリスト $a_1 < \cdots < a_\alpha$ 上で x 座標 x に関する探索を行い，$\vec{s}$ を含む垂直な帯状の領域 S_i^x $(0 \leq i \leq \alpha + 1)$ を見つける．
2: 端点のリスト $b_1 < \cdots < b_\beta$ 上で y 座標 y に関する探索を行い，$\vec{s}$ を含む水平な帯状の領域 S_j^y $(0 \leq j \leq \beta + 1)$ を見つける．
3: それぞれのビットマスク B_i^x と B_j^y を取り出す．
4: **return** $B = (B_i^x \ \& \ B_j^y) \in \{0, 1\}^w$;

図 12: d-HT-bin のヒット表計算

形制約が変数分離形の制約のみで定められる場合には，図 11 のように，軸平行な幾何制約集合 $lin(P)$ を x 軸と y 軸に平行な制約からなる部分集合に分離し，各次元毎に前処理を行い，実行時に入力点 $\vec{s}_t = \{x_1, \ldots, x_d\}$ に対して各次元のヒット表の論理積を求めることで扱える．これは，ヒット表のビットマスク表現のビット積 $P(\vec{s}) = \wedge_i \ell_i(x_i)$ で高速に求めることができる．これにより，ビットマスクの検索処理の著しい簡素化と高速化が可能である．

まず，$lin(P)$ 中の線形不等式で y 軸に平行なもの（垂直線）の全体を考える．これらの垂直線全体を x 座標で昇順にソートしたリストを $L_x = \{\sigma_i x_i \leq a_i\}_{i=1}^{\alpha}$ $(\sigma_i \in \{+1, -1\})$ とおく．次に，これらの垂直線を用いて，平面 $\mathbb{R}^2$ を互いに交わらない，垂直な帯状の領域 $\mathbb{R}^2 = S_0^x + \cdots + S_\alpha^x$ に分割する．各領域 S_i^x $(0 \leq i \leq \alpha)$ に，その領域との共通部分が存在する制約 $p \in P$ すべての集合を表すビットマスク $B_i^x \in \{0, 1\}^w$ を関連付ける．同様に，x 軸に平行な直線（水平線）の全体を用いて，互いに交わらない水平な帯状の領域 $\mathbb{R}^2 = S_0^y + \cdots + S_\beta^y$ に平面 $\mathbb{R}^2$ を分割し，各領域 S_j^y $(0 \leq j \leq \beta)$ に，それを満たす制約集合を表すビットマスク $B_j^y \in \{0, 1\}^w$ を関連付ける．

d-HT-bin は，与えられた入力点 $\vec{s} = (x, y)$ に対して，x 座標と y 座標それぞれに対して領域のリスト上の探索を行い，それぞれに対応づけられたビットマスクをビット積で結合したものを返す．

各次元に z 個の制約が含まれているとする．各次元に関して，$O(z)$ のリストを探索してビットマスクを返すので，変数分離形のアルゴリズムの計算量は次のようになる．

定理 3 手続き d-HT-bin は，幾何制約のクラス $\mathcal{C}_{ieq,real}^d$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(\frac{1}{w}dm^2)$ と領域 $O(\frac{1}{w}dm^2)$ を用いて，各入力点 $p \in \mathbb{R}^d$ *1* 個あたり，$O(\frac{1}{w}dm \log m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズである．

補題 2 から，次の系がただちに成立する．

系 3 幾何制約のクラス $\mathcal{C}_{ieq,real}^d$ 上に対して，d 次元実数ストリーム上の次元間に相関のない幾何時系列パターン照合問題は，前処理時間 $O(\frac{1}{w}dm^2)$ と領域 $O(\frac{1}{w}dm^2)$ を用いて，$O(\frac{1}{w}dmn \log m)$ 時間で解く．$m = ||\pi||$ は，パターン π のサイズであり，$n = |S|$ はストリーム S の長さである．

5.3 変数分離形でない制約集合への適用

$ax + by \leq c$ というような変数分離形でない制約は，新しい仮想的な変数 $z = ax + by$ を各時点で計算することで，変数分離形の制約 $z \leq c$ に帰着することができる．また，変数の前の値との比較 $x - x.prev \leq c$ のような制約についても，変数 $z = x - x.prev$ の導入によって実現できる．

5.4 まとめ

本節で述べた変数分離形の場合のヒット表計算は，$d \geq 3$ の場合にもそのまま適用できる．これは次章で述べるスラブ法と比べて，構造と前処理の簡単さに利点がある．とくに，3 次元以上の高次元データの場合に有利である．

6 多次元ストリームに対する効率よいヒット表計算：次元間に相関のある幾何制約の場合

本節では，複数次元間に相関がある場合のクラス $\mathcal{C}_{linear}^d$ の幾何制約集合に対するヒット表計算の効率良いアルゴリズムを与える．

6.1 手法の概要

議論の簡便のため，入力点の空間として 2 次元実数空間 $\mathbb{R}^2$ を考える．いま，2 次元平面 $\mathbb{R}^2$ 上の線形幾何制約の集合 $P = \{p_1, \ldots, p_w\} (w \geq 1)$ が与えられたと仮定する．以後，P 中に含まれるすべての線形不等式からなる集合を $lin(P) \subseteq \mathcal{L}$ で表し，その総数を $v = |lin(P)|$ で表す．すなわち，$O(v) = O(||\pi||)$ である．

//実行時手続き d-HT-Naive($\vec{s} = (x, y) \in \mathbb{R}^2$)

1: $B \leftarrow 0^w$;
2: **for** $i \leftarrow 1, \ldots, w$ **do**
3: **foreach** $\ell \in p_i$ **do**
4: **if** $\ell(\vec{x})=1$ **then** $B \leftarrow B \mid (1 << i-1)$;
5: **end foreach**
6: **end for**
7: **return** B;

図 13: 素朴な方法によるヒット表計算アルゴリズ d-HT-Naive

6.2 素朴な方法

素朴な方法として，前処理を行わず，入力が与えられるたびに，集合 P の制約を逐次的に評価する方が考えられる．図 17 に，素朴なヒット表計算手続 d-HT-Naive を示す．

定理 4 手続き d-HT-naive は，幾何制約のクラス $\mathcal{C}^d_{linear}$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(1)$ と領域 $O(1)$ を用いて，各入力点 $p \in \mathbb{R}^d$ 1 個あたり，$O(m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズである．

補題 2 から，次の系がただちに成立する．

系 4 幾何制約のクラス $\mathcal{C}^d_{linear}$ 上に対して，d 次元実数ストリーム上の次元間に相関のない幾何時系列パターン照合問題は，手続き d-HT-naive を用いて前処理時間 $O(dm^2\frac{1}{w})$ と領域 $O(dm^2\frac{1}{w})$ で，$O(\frac{1}{w}dnm\log m)$ 時間で解ける．ここに，$m = ||\pi||$ はパターン π のサイズであり，$n = |S|$ はストリーム S の長さである．

6.3 スラブ法

本節では，スラブ法を用いたヒット表計算アルゴリズム d-HT-slab を与える．スラブ法は，あらかじめ前処理でビットマスクを計算しておき，入力毎のビットマスクを表引きで高速化する方法である．アルゴリズム d-HT-slab は，時点幾何パターン集合 P が与えられると，次のように前処理を行う．

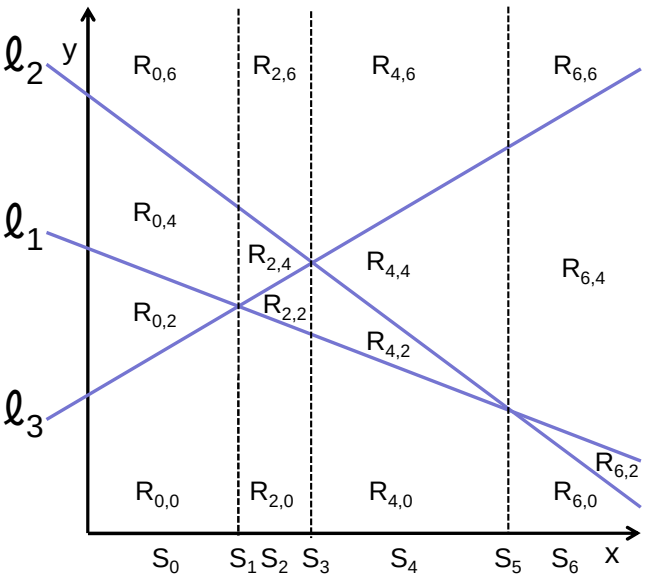

図 14: スラブ法：帯状領域と台形領域

前処理 初めに，$lin(P)$ 中のすべての線形不等式から二つずつとった組み合わせ $\ell_1, \ell_2 \in lin(P)$ を考えて，それらの交点 $\ell_1 \cap \ell_2$ 全体の集合 $I(P) = \{\, \ell_1 \cap \ell_2 \mid \ell_1, \ell_2 \in lin(P) \,\} \subseteq \mathbb{R}^2$ を求める．次に交点の x 座標を $x_1 < \cdots < x_\alpha$ と昇順にソートして並べる．これらの点を通って y 軸に平行な直線（垂直線）を引いて区切り，平面 $\mathbb{R}^2$ を互いに交わらない垂直な帯状の領域と，垂直線のみからなる領域 $\mathbb{R}^2 = S_0 + \cdots + S_{2\alpha}$ に分割する．ここに，$0 \le i \le \alpha$ に対して，$2i$ 番目の帯は $S_{2i} = (x_i, x_{i+1}) \times \mathbb{R}$ であり，$2i+1$ 番目の帯は $S_{2i+1} = [x_{i+1}, x_{i+1}] \times \mathbb{R}$ である．ただし，$x_0 = -\infty, x_{\alpha+1} = +\infty$ と定義する．

各 $i = 0, \ldots, 2\alpha$ に対して，垂直な帯 S_i と，S_i を横切る $lin(P)$ のすべての直線を考える．S_i を横切る直線で区切り，台形領域と，直線のみからなる領域 $S_i = R_{i,0} + \cdots + R_{i,2\beta}$ に分割する．S_i の内部ではどの $lin(P)$ の直線も交わらないので，台形領域 $R_{i,0}, \ldots, R_{i,2\beta}$ は y 軸の向きに一意に順序づけることができる．各台形領域 $R_{i,j}$ の内部では明らかに満足される制約は同じなので，各 $R_{i,j}$ に，満足される制約 $p \in P$ すべての集合を表すビットマスク $B_r \in \{0,1\}^w$ を関連付ける．図 14 に，3 つの線形制約 $\ell_1, \ldots, \ell_3$ による帯状領域と台形領域の分割を示す．

検索処理 図 6.3 に，スラブ法を用いて実行時に入力点に対するビットマスクを計算する手続きを与える．このアルゴリズムは，入力点 $\vec{s} = (x, y)$ が与えられると，x 座標に関して二分探索を行い，$\vec{s}$ を

procedure algd-HT-Slab($\vec{s} = (x, y) \in \mathbb{R}^2$)

1: x 座標 $x_1 < \cdots < x_\alpha$ に関する二分探索を行い $\vec{s}$ を含む帯状の領域 S_i $(0 \le i \le 2\alpha)$ を見つけ
2: S_i に含まれる台形領域に関する二分探索を行い $\vec{s}$ を含む台形領域 $R_{i,j}$ $(0 \le j \le 2\beta)$ を見つけ
3: **return** $B_r \in \{0,1\}^w$;

図 15: スラブ法におけるヒット表の計算

含む帯状の領域 S_i を見つける．次に，S_i の内部 y 座標に関して二分探索を行い，$\vec{s}$ を含む台形領 $R_{i,j}$ を見つける．最後に，台形領域 $R_{i,j}$ に対応づられたビットマスクを返す．

計算量 集合 $lin(P)$ 中の v 本の直線の交点数は大で，$M = v^2$ 箇所なので，領域の数 α, β は高々 で抑えられる．計 2 回の $O(\log M) = O(\log v)$ 時の二分探索を行うので，計算量は次のようになる

定理 5 手続き d-HT-slab は，幾何制約のクラス $\mathcal{C}^d_{linear}$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(\frac{1}{w}m^3)$ と領域 $O(\frac{1}{w}m^3)$ を用いて，各入力点 $p \in \mathbb{R}^d$ 1 個あたり，$O(\frac{1}{w}m \log m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズである．

補題 2 から，次の系がただちに成立する．

系 5 幾何制約のクラス $\mathcal{C}^d_{linear}$ 上に対して，d 次元実数ストリーム上の次元間に相関のない幾何時系列パターン照合問題は，手続き d-HT-slab を用いて前処理時間 $O(\frac{1}{w}m^3)$ と領域 $O(\frac{1}{w}m^3)$ で，$O(\frac{1}{w}mn \log m)$ 時間で解ける．ここに，$m = ||\pi||$ はパターン π のサイズであり，$n = |S|$ はストリーム S の長さである．

これは素朴な場合に比べて，指数的なスピードアップである．

次元が $d \ge 3$ の高次元データの場合にも，再帰的な拡張でスラブ法を適用可能である．しかし，索引構造の複雑さから，実際には $d \le 2$ が実用的だと思われる．

6.4 バケット法

実際的な高速化技法として，式の定数や交点の座標が，微小長さ $\Delta \in \mathbb{R}$ を単位として離散化されてお

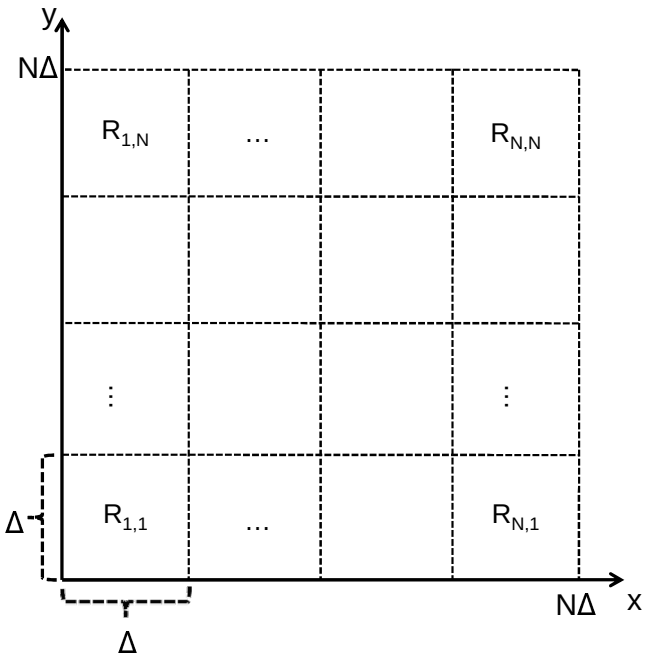

図 16: バケット法

り，入力値が上下限をもつ有限な実数区間 $[a, b] \subseteq \mathbb{R}$ から与えられる場合が重要である．この場合は，スラブ法における二分探索を行わず，2 次元配列上のバケットを用いた表引きによる高速化が可能である．本節では，バケット法を用いたヒット表計算アルゴリズム d-HT-bucket を与える．

本節で述べる手法は，4.1 節の 1 次元整数ストリームのヒット表計算を，d 次元実数ストリームへ拡張した手法と考えることができる．

前処理 問合せ可能な領域を $[0\Delta, N\Delta) \times [0\Delta, N\Delta) \subseteq \mathbb{R}^2$ とする．2 次元配列 $A[1..N, 1..N]$ 上に，一辺が Δ の直方体状の小領域 $R_{i,j} = [i-1\Delta, i\Delta) \times [j-1\Delta, j\Delta)$ を表すバケット $A[i,j]$ $(1 \le i, j \le N)$ を配置する．その小領域 $R_{i,j}$ 上で満足される制約集合 $p \in P$ を表すビットマスク $B_{i,j} \in \{0,1\}^w$ をバケットに直接格納する．直線がバケットを斜めに横断する場合には，$B_{i,j}$ を検索後に個々の制約を順に検査する事後チェックが必要である．

検索処理 図 6.4 に，実行時に入力点に対するビットマスクを計算する手続きを示す．d-HT-bucket は，入力点 $\vec{s} = (x, y)$ が与えられると，$i = \lceil x/\Delta \rceil$ と $j = \lceil y/\Delta \rceil$ に対して，バケット $A[i,j]$ を直接表引きし，関連付けられたビットマスク $B_{i,j} \in \{0,1\}^w$ を返す．

procedure HTConvexBucket($\vec{s} = (x, y) \in \mathbb{R}^2$)

1: $i = \lceil x/\Delta \rceil$;
2: $j = \lceil y/\Delta \rceil$;
3: バケット $A[i, j]$ を直接表引きする;
4: 関連付けられたビットマスク $B_{i,j} \in \{0, 1\}^w$ を返す.

図 17: バケット法におけるヒット表の計算

計算量

定理 6 手続き d-HT-bucket は，幾何制約のクラス $\mathcal{C}^d_{linear}$ に対して，制約集合 P に対するヒット表計算問題を，前処理時間 $O(\frac{1}{w}m^2N^2)$ と領域 $O(\frac{1}{w}m^2N^2)$ を用いて，各入力点 $p \in \mathbb{R}^d$ 1 個あたり，$O(T_\gamma + \frac{1}{w}m)$ 時間で解く．ここに，$m = ||P||$ は，P のサイズ，T_γ は事後チェックに要する時間である．

補題 2 から，次の系がただちに成立する．

系 6 幾何制約のクラス $\mathcal{C}^d_{linear}$ 上に対して，d 次元実数ストリーム上の次元間に相関のない幾何時系列パターン照合問題は，手続き d-HT-bucket を用いて前処理時間 $O(\frac{1}{w}m^2N^2)$ と領域 $O(\frac{1}{w}m^2N^2)$ で，$O(T_\gamma + \frac{1}{w}mn)$ 時間で解ける．ここに，$m = ||\pi||$ は，パターン π のサイズであり，N はバケットの分割数，$n = |S|$ はストリーム S の長さ，T_γ は事後チェックに要する時間である．

T_γ の時間計算量は理論的には $O(v)$ だが，ランダムな制約データの場合は定数とみなせると期待できる．また，事後チェックを行わないことで，$T_\gamma = O(1)$ となる近似的なパターン照合の実装とすることができる．

6.5 まとめ

本節では，一般的な線形制約に対する高速なヒット表計算のアルゴリズムとして，バケット法とスラブ法を与えた．バケット法はスラブ法と比べて，2 次元空間上で検索領域が離散的かつ有界な場合に，構造の簡単さと検索速度で著しい利点がある．一方で，3 次元以上の高次元では，データが疎になりほとんどのバケットが無駄になることから実用性は低いと思われる．

7 実験

本節では，提案アルゴリズム BPS の性能を評価するための計算機実験を行う．

7.1 実験 1

4 章の 1 次元実数ストリームと 1 次元整数ストリームに対するアルゴリズムと，5 章の変数分離形に対するアルゴリズムの性能を評価する実験を行った．

実験 1 では，以下に示される 6 種類のアルゴリズムを C++言語で実装し用いた．

NAIVE: $O(mn)$ 時間の素朴なパターン照合アルゴリズム [12] の実数ストリーム版.

L2R: KMP 手法 [8] の実数ストリーム版である Harada [7] と Sadri ら [13] の L2R アルゴリズムの実装.

R2L: BM 手法 [3] の実数ストリーム版である Harada [7] の R2L アルゴリズムの実装.

BPS_bin: 4 章の 1-HT-bin を用いた BPS アルゴリズムで，ヒット表計算を配列上の二分探索 [1] で行うもの.

BPS_lin: 4 章の 1-HT-bin の二分探索を線形探索 [1] に置き換えた BPS アルゴリズム.

BPS_tab: 4 章の 1-HT-table を用いた整数ストリームに対する BPS アルゴリズムで，ヒット表計算を表引きで実現したもの.

$d > 1$ 次元の実験については，BPS_bin, BPS_lin, BPS_tab を 5 と同様の議論で d 次元に拡張したアルゴリズム d-BPS_bin, d-BPS_lin, d-BPS_tab をそれぞれ代替として用いた.

7.1.1 データ $N^d = [1, c]^d \subseteq \mathbb{N}^d$ 上の一様分布で独立に生成した n 個の d 次元上の点からなるストリームデータ S と，ランダムパターン $\pi = q_1 \cdots q_m$ を用いた．ここにパラメータ $0 \leq \varepsilon \leq c$ に対して，独立なランダム整数 $a^i_j \in [1, c - \varepsilon]$ に対し，$q_i = \alpha^i_1 \wedge \cdots \wedge \alpha^i_d$ $(1 \leq i \leq m)$ かつ $\alpha^i_j = (x_j > a^i_j) \wedge (x_j \leq a^i_j + \varepsilon)$ ととった．とくに指定しない限り，$d = 1$ とし，$c = 100, n = 10^7, m = 3$ とした．

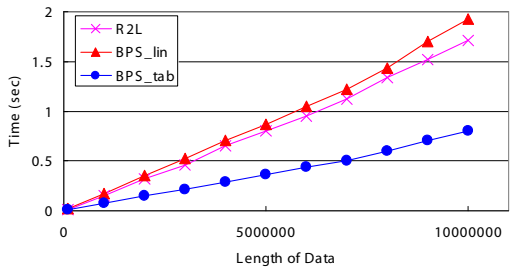

図 18: データサイズと計算時間

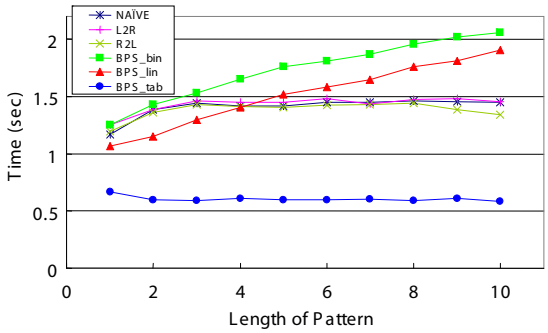

図 19: パターン長と計算時間

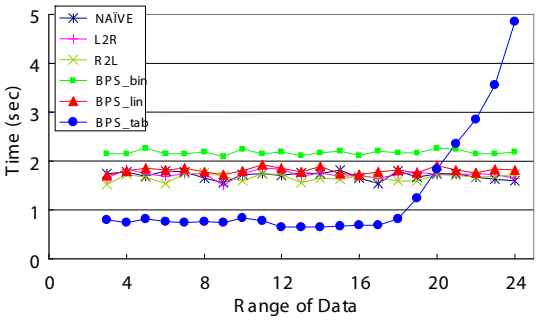

図 20: データ幅と計算時間

7.1.2 方法

実験 1 は，PC（Pentium M 1.30GHz，レジスタ長 32bit，RAM 1GB，Cygwin, g++3.4.4）上で行った．パラメータを変えて生成したストリームデータに対して，各アルゴリズムの計算時間を計測した．ストリームデータは主記憶上に読み込み実験を行った．計算時間には，データの読み込み時間は含めていない．

7.1.3 実験結果

データサイズに対する規模耐性． データサイズ n に対するアルゴリズムの規模耐性を評価するため，R2L と，BPS_lin，BPS_tab に対して，n を 10^5 個から 10^7 個まで変化させて，それぞれの計算時間を測定した．計算時間は 30 回の平均である．結果を図 18 に示す．BPS アルゴリズムの計算時間は n に比例し，3.3 節の定理 2 から予測されるふるまいと整合する．具体的には，$n = 10^7 = 10,000,000$ のとき，BPS_tab の計算時間は 0.8 秒程度であり，十分高速だった．

パターン長と計算時間． パターン長 m の変化に対するふるまいを評価するため，実装した 6 つのアルゴリズムに対して，m を 1 から 10 まで変化させたときの計算時間を測定した．計算時間は 30 回の平均である．結果を図 19 に示す．BPS_tab の計算時間は，m の値によらず 0.6 秒前後であり，6 つのアルゴリズムの中で最も高速であった．NAIVE と，L2R，R2L の計算時間はパターン長によらず 1.5 秒前後で一定だった．BPS_lin は，$m \leq 4$ では NAIVE と，L2R，R2L より高速だったが，$m \geq 5$ ではこれらのアルゴリズムより低速だった．BPS_bin は，同様の傾向を持つがより低速だった．

BPS_tab のデータ幅に対する耐性． 整数ストリームに対するアルゴリズム BPS_tab は，ストリームの値域 c に比例するサイズのマスク配列を用いるため，c が大きい場合の有効性は未知である．そこで，パターンの相対幅 $h = 25(\%)$ を一定に保ったまま，c を $2^3 = 8$ から $2^{24} = 16,777,216$ まで変化させて計算時間を測定した．結果を図 20 に示す．16 ビット整数程度のデータ幅までは，BPS_tab は他のアルゴリズムの 2.5 倍高速であり，十分使用に耐えることがわかった．

複数パターン照合． 3 つの BPS アルゴリズムを複数パターン照合へ拡張し（3.4 節参照），パターン数 l を 1 から 8 まで変化させて計算時間を測定した．NAIVE と，L2R，R2L は複数パターンへの拡張が自明でないため，各パターンの照合にかかった計算時間の和を用いた．結果を図 21 に示す．BPS_tab の計算時間は，パターン数によらず一定で，0.6 秒程度だった．BPS_bin と BPS_lin の計算時間は，パターン数の増加にともない増加したものの，$l \geq 2$ では，NAIVE と，L2R，R2L よりも高速だった．たとえば，パターン数 6 では，R2L と比較して，BPS_tab は 10 倍程度，BPS_bin と BPS_lin は 2.5 倍程度高速であった．

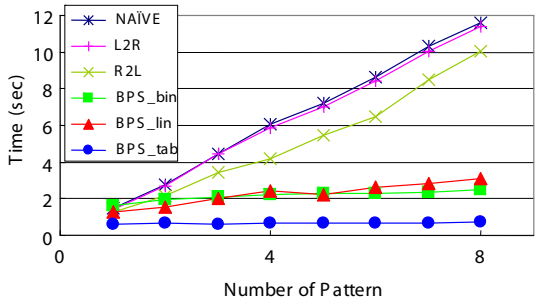

図 21: パターン数と計算時間

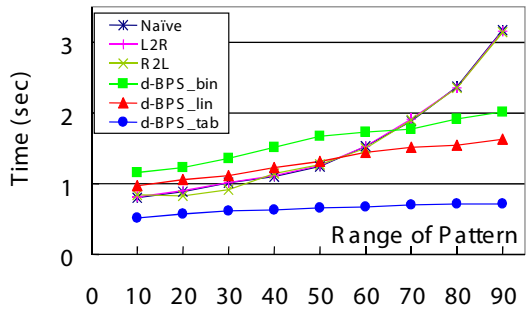

図 22: パターン幅 ε に対する計算時間

パターン幅に対する計算時間． パターンに対する依存性を知るために，ε を 10 から 90 まで変化させたときの各アルゴリズムの計算時間を測定した．計算時間は 30 回の平均である．$d = 3$ とした．結果を図 22 に示す．d-BPS_tab と，d-BPS_lin，d-BPS_bin の計算時間は相対幅によらず安定して高速だった．一方，NAIVE と，L2R，R2L の計算時間は，相対幅の増加にともない 3 倍程度増加した．たとえば，$\varepsilon = 70(\%)$ のとき，d-BPS_tab と d-BPS_lin はそれぞれ，NAIVE と，L2R，R2L と比較して 4 倍と 2 倍程度高速であった．

d に対する計算時間． 図 23 に，d を 1 から 8 まで変化させたときの各アルゴリズムの計算時間を示す．$m = 4$ とした．次元 d によらず制約が空間 R^d を満たす体積を一定割合 $\gamma = 0.25$ とするため，$\varepsilon = \gamma^{1/d}$ とした．全アルゴリズムの計算時間は d に比例し，とくに，d-BPS_tab は他のアルゴリズムと比較して 2.5 倍程度高速だった．

実験のまとめ． $c \leq 2^{16}$ 程度の整数ストリームでは BPS_tab が最も高速だった．実数ストリームに対

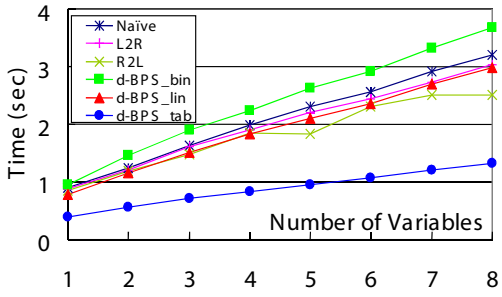

図 23: 次元数 d に対する計算時間

しては，$m \leq 4$ では BPS_lin が，$m \geq 5$ では L2R や R2L が高速だった．また，BPS アルゴリズムはパターンの内容によらず比較的安定した性能であった．

8 実験 2

6 章の，d 次元空間上の線形制約クラスの時系列幾何パターンを扱うためのヒット表計算の実行速度を比較するため，合成データ上で実験を行った．ここでは，節 2.4.3 の例 4 で述べたような凸多角形からなるパターンに対応するヒット表構築アルゴリズムを，自明なアルゴリズム d-HT-naive と，スラブ法を用いたアルゴリズム d-HT-slab，バケット法を用いたアルゴリズム d-HT-bucket として実装した．なお，今回実験で用いた d-HT-bucket は，ひとつのバケット内に複数の領域がある場合の事後チェックを行わない簡易な実装である．また，バケットの分割は 512×512 とした．

実験 2 は，PC（Core 2 Duo T7200 2.00GHz，レジスタ長 32bit，RAM 1GB，VisualC++ 2008）上で行った．

8.1 データ

二次元平面 $R = [0, 1] \times [0, 1] \subseteq \mathbb{R}^2$ 上にランダムに生成した $e = 10$ 個の点 $p_1, \ldots, p_e \in R$ に対する凸包を計算した．図 24 に凸包の例を示す．生成した凸包を m 個連結したものを長さ m のパターンとした．ランダムに生成した 2 次元座標上の長さ n の点列 $S = (\vec{s}_1, \ldots, \vec{s}_n) \in R^*$ を 2 次元実数ストリームとした．

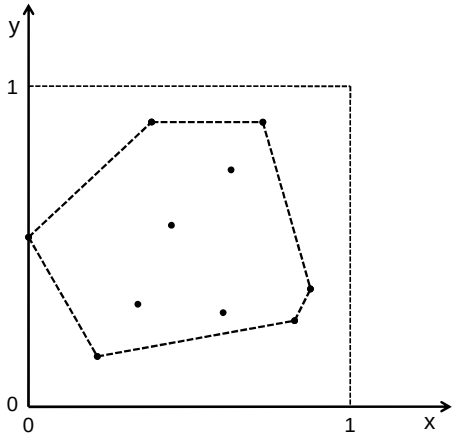

図 24: ランダムな e 点の凸包

8.2 方法

生成したパターンに対応するヒット表を，各アルゴリズムを用いて構築した．これらのヒット表に対して点列を入力として与え，ビットマスクを計算するために必要とした総計算時間を計測した．

8.3 実験結果

パターン長 m を 1 から 8 まで変化させたときの各アルゴリズムの計算時間を図 25 に示す．d-HT-naive の計算時間は m に比例して増大した．d-HT-slab の計算時間は m が大きくなるにつれ対数的に増大したものの，$m = 4$ で d-HT-naive に比べて 4 倍程度高速であった．d-HT-bucket の計算時間は m の値に関わらず一定であり，$m = 4$ で d-HT-naive に比べて 25 倍程度高速であった．これらの振舞いは，6 章で述べた計算量に合致している．また，d-HT-bucket が事後チェックを行わないことにより誤ったビットマスクを返す割合を調べたところ，$m = 4, e = 10$ のとき 0.5% 程度であった．

9 d 次元実数ストリーム上の時系列パターン照合のモーション検索への応用

本節では，アルゴリズム BPS の実データへの応用実験として，モーションデータからの検索実験を行う．

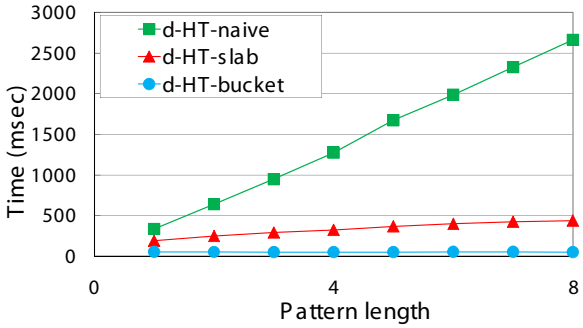

図 25: パターン長に対する計算時間

9.1 データ

モーションデータとして，mocapdata(http://www.mocapdata.com) で bvh 形式で公開されている歩行モーションデータ walk-normal-azumi.bvh を用いた．

今回データに用いた bvh 形式は，スケルトン階層記述部と各関節のモーションデータ記述部からなるファイル形式である．スケルトン階層記述部には，モデルの関節と関節間の長さが階層構造で記述されている．モーションデータ記述部には，各時点での姿勢が，各関節について 3 次元の絶対座標と 3 次元の回転角による相対座標の計 6 次元データで記述されている．

モーションデータファイル walk-normal-azumi.bvh のスケルトン階層記述部の一部抜粋を図 26 に，モーションデータ記述部の一部抜粋を図 27 にそれぞれ示す．walk-normal-azumi.bvh では，腰を根とした 19 関節の木構造で人体モデルを記述している．このモーションデータは，5 歩程度歩いて 180 度方向転換することを 2 往復繰り返した動きのデータである．

今回の実験では，19 関節 114 次元の実数ストリームのうち，root 関節と絶対座標の次元を除いた 54 次元を用いた．これは，人体モデルが同様の姿勢をとっていれば絶対座標や向きに関わらずマッチする結果を意図したためである．

パターンとして，モーションデータの時点 $t = 206$ から切り出した長さ $m = 16$ の $d = 54$ 次元ストリーム $S[t..t + m - 1]$ に区間幅を加え，各次元の制約がすべて論理積で結合した変数分離形のパターン $\pi = q_1 \cdots q_m$ を用いた．すなわち，制約 $q_i = \alpha_1^i \wedge \cdots \wedge \alpha_d^i, (1 \leq i \leq m)$ であり，各次元について，$\alpha_i^j = (x_j < p_j[i] + \varepsilon_j/2) \wedge (x_j \geq p_j[i] - \varepsilon_j/2)$ である．

```
HIERARCHY
ROOT Hips
{
  OFFSET 15.509 39.6767 29.6418
  CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
  JOINT LeftHip
  {
    OFFSET 3.43 -1.56319e-013 0
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT LeftKnee
    {
      OFFSET -6.39488e-014 -18.47 -6.66134e-014
      CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
      JOINT LeftAnkle
      {
        OFFSET -3.55271e-014 -17.95 -4.9738e-014
        CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
        End Site
        {
          OFFSET 0 -3.12 0
        }
      }
    }
  }
  JOINT RightHip
  {
    OFFSET -3.43 -2.70006e-013 0
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT RightKnee
    {
      OFFSET -2.13163e-014 -18.47 -4.9738e-014
      CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
      JOINT RightAnkle
      {
        OFFSET 3.55271e-014 -17.95 -7.81597e-014
        CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
        End Site
        {
          OFFSET 0 -3.12 0
        }
```

図 26: bvh 形式のスケルトン階層記述部 (抜粋)

```
MOTION
Frames: 747
Frame Time: 0.0333333
18.6248 37.3733 -11.8532 -2.65337 -10.5432 1.98998 3.43 -9.9476e-014
18.6156 37.3758 -11.8453 -2.66078 -10.4967 2.00046 3.43 -7.16488e-01
18.6047 37.3825 -11.8184 -2.73248 -10.5234 2.02125 3.43 -2.93184e-01
18.5917 37.3856 -11.8122 -2.76174 -10.444 2.0335 3.43 2.02636e-014 5
18.5951 37.3925 -11.807 -2.70346 -10.3744 2.15962 3.43 6.98457e-014
18.5932 37.396 -11.8038 -2.72764 -10.3429 2.1452 3.43 1.12176e-013 5
18.5916 37.4024 -11.7995 -2.70331 -10.3133 2.14501 3.43 1.40003e-013
18.5904 37.4069 -11.8025 -2.72153 -10.311 2.1719 3.43 1.46076e-013 5
18.5919 37.4107 -11.804 -2.68207 -10.2907 2.14413 3.43 1.23141e-013
18.5896 37.4109 -11.8025 -2.69692 -10.2652 2.17604 3.43 6.39488e-014
18.5869 37.4056 -11.797 -2.70635 -10.2277 2.20049 3.43 -1.13687e-013
18.5881 37.4132 -11.7983 -2.67474 -10.253 2.22709 3.43 -2.62901e-013
18.5832 37.4229 -11.7989 -2.63964 -10.2441 2.22661 3.43 -4.26326e-01
18.5883 37.416 -11.8014 -2.64148 -10.1717 2.25759 3.43 1.49214e-013
18.5869 37.4181 -11.8018 -2.61146 -10.1778 2.25279 3.43 2.15461e-013
18.584 37.4174 -11.8018 -2.62401 -10.1601 2.25511 3.43 2.51923e-013
18.5746 37.4225 -11.8071 -2.60332 -10.1176 2.28643 3.43 2.62316e-013
18.5655 37.426 -11.8013 -2.61954 -10.1768 2.29847 3.43 2.50355e-013
18.5626 37.4272 -11.8005 -2.61947 -10.1672 2.33935 3.43 2.19757e-013
18.5654 37.4285 -11.8025 -2.57974 -10.1278 2.36644 3.43 1.74236e-013
18.5602 37.4273 -11.7992 -2.58022 -10.1519 2.33497 3.43 1.17508e-013
18.5555 37.4322 -11.7994 -2.59663 -10.1455 2.30291 3.43 5.32907e-014
18.5545 37.4279 -11.7964 -2.62233 -10.1322 2.31339 3.43 -1.47018e-01
18.5555 37.4249 -11.7861 -2.6501 -10.1341 2.3438 3.43 -8.27532e-014
18.5508 37.424 -11.778 -2.71981 -10.1114 2.28403 3.43 -1.47148e-013
18.5514 37.4276 -11.7646 -2.68871 -10.1182 2.24901 3.43 -2.0417e-013
18.556 37.428 -11.7563 -2.67509 -10.0994 2.22267 3.43 -2.50104e-013
18.5624 37.4338 -11.7363 -2.65141 -10.1806 2.26485 3.43 -2.81233e-01
18.5618 37.4263 -11.7261 -2.6942 -10.1483 2.25878 3.43 -2.93843e-013
18.5669 37.426 -11.7212 -2.70766 -10.0714 2.26682 3.43 -2.84217e-013
18.5685 37.4246 -11.708 -2.63535 -9.9786 2.32847 3.43 -2.13163e-013
18.5657 37.4218 -11.6892 -2.60361 -10.0148 2.34771 3.43 2.13163e-013
18.5618 37.4219 -11.6777 -2.63759 -10.0758 2.34864 3.43 2.34479e-013
18.5618 37.4286 -11.6642 -2.64911 -10.1025 2.29191 3.43 1.7053e-013
18.5677 37.4373 -11.6515 -2.61452 -10.1561 2.29384 3.43 -1.42109e-01
18.5625 37.4418 -11.641 -2.64288 -10.1465 2.36367 3.43 -2.72423e-013
18.5648 37.4524 -11.6401 -2.59729 -10.0659 2.36456 3.43 -3.65906e-01
```

図 27: bvh 形式のモーションデータ記述部 (抜粋)

ただし，$p_j[i]$ は $\vec{p}_j$ の i 番目の要素とする．また ε_j は j 番目の次元に定めた区間幅である．今回の実験では，j 番目の次元のストリーム上の最大値と最小値の差 $range_j$ とパラメータ h を用いて $\varepsilon_j = range_j \times h$ とした．これは各次元がとりうる値の幅を考慮して区間幅を加えるためである．

9.2 方法

パターンを walk-normal-azumi.bvh から生成し，walk-normal-azumi.bvh をストリームとして，d − BPS_bin を用いて照合を行った．h を 0 から 0.05 ずつ大きくしていき，どのような検索結果が得られるか観察した．また，照合にかかった計算時間を測定した．

今回の実験を行うにあたって，BPS の GUI を実装した．概観を図 28 に示す．各部位の説明は次のとおり．

(1): パターンのプレビュー画面

(2): モデルの関節リスト

(3): ストリームのプレビュー画面

(4): 照合結果

(5): パターンの始点 t を指定するスクロールバー

(6): パラメータ h の入力部

(7): パターン長 m の入力部

(8): 検索ボタン

(9): アニメーションの再生・停止ボタン

パターンとストリームとして，それぞれ bvh ファイルを読み込み，パターンのパラメータ t と，h，m を GUI で指定できる実装となっている．また，パターンと照合結果の動きをアニメーションで確認できる．

9.3 結果

目視によれば，このストリーム中に，パターン π(図 29-(a)) と似た動きは π 自身を含め 9 箇所に存在した．h を 0 から 1.35 照合結果を表 3 に示す．ここに，π に似た 9 箇所の動きを時系列順に $A_1, \ldots, A_9$ とする．また，π に似ていないと判断される動きを時系列順に $B_1, \ldots$ とする．

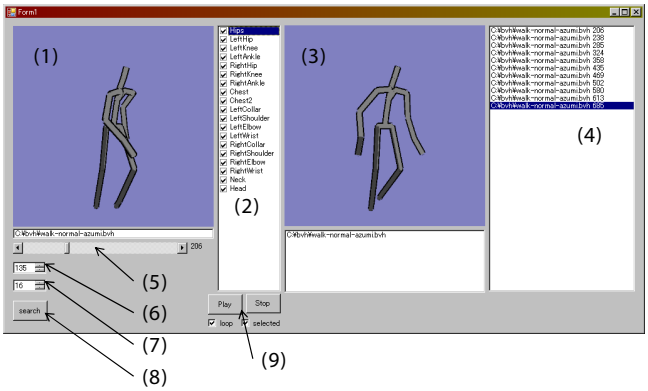

図 28: BPS の GUI 概観

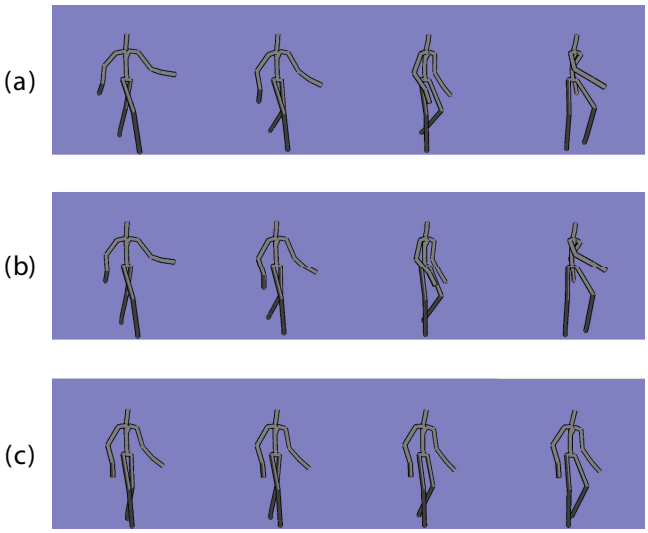

図 29: (a) パターン π (b) 照合の成功例 (c) 照合の失敗例

$h = 0 \sim 0.8$ の間では，$A_1 : \pi$ そのものしか得られなかった．$h = 0.85$ のとき，π に似た動き A_5(図 29-(b)) が検索結果として得られた．$h = 1.35$ まで引き上げた段階で 9 箇所すべてが結果として得られた．しかし，同時に 2 箇所の似ていない動き (図 29-(c)) が結果として得られた．

また計算時間は，ファイルからの読み込み時間を含めないで $n = 747$ のモーションデータで 150msec 程度だった．これは十分実用的な速度であると考えられる．

表 3: パラメータ h に対する照合結果

h	照合結果
$0 \sim 0.8$	A_1
$0.85 \sim 1.05$	A_1, A_5
$1.10 \sim 1.15$	A_1, A_5, A_7
1.20	A_1, A_2, A_5, A_6, A_7
1.25	$A_1, A_2, A_4, A_5, A_6, A_7, B_2$
1.30	$A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_9, B_2$
1.35	$A_1, A_2, B_1, A_3, A_4, A_5, A_6, A_7, A_8, A_9, B_2$

10 おわりに

本論文では，文字列アルゴリズムにおけるビット並列手法を，d 次元実数ストリーム上のパターン照合問題に適用し，幾何的制約による複雑な時系列パターンを照合する問題を考察した．また，この照合問題のために，複数の幾何制約問合せを高速に計算するヒット表計算の手法を示し，それらとビット並列手法を組み合わせる機構を提案した．特に制約条件が軸並行の場合に対して，変数分離法というビット並列による高速な問合せ解決手法を提案し，これを用いて照合問題を実際に高速に解くアルゴリズムを導いた．提案手法は今後のビット幅の拡大に比例した高速化が期待できる．

主結果として，1 次元整数ストリームに対して，$O(mn)$ 時間 の NAIVE 手法より高速な $O(\frac{1}{w}mn)$ 時間アルゴリズムを与えた．また，1 次元実数ストリームに対して，$O(\frac{1}{w}mn \log m)$ 時間アルゴリズムを与えた．ここに，m, n, w はそれぞれ，パターン長と，ストリーム長，マシンレジスタ長である．とくに $m \leq w$ のときは前者は $O(n)$ 時間アルゴリズムとなり，現行の $w = 32$ or 64 ビットのハードウェア上で実用的である．

加えて，変数分離形の d 次元実数ストリームに対して，$O(\frac{1}{w}dmn \log m)$ 時間の照合アルゴリズムを与えた．また，$d = 2$ 次元実数ストリームに対して $O(\frac{1}{w}mn \log v)$ 時間の照合アルゴリズムを与えた．ここに，v はパターンサイズである．

今後の課題として，より一般的な制約への拡張や，より柔軟な正規表現パターンへの拡張が挙げられる．実数データストリームからのオンラインマイニングへの拡張も興味深い課題である [2]．また，モーションデータの検索実験では，正しい照合結果と共に，意図しない照合結果も得られた．より尤もらしい照合結果を得るための手法が必要と考えられる．また，動きの大きく違うモーションデータに対する照合実

験など，より深い観察が課題として挙げられる．モーションデータの他にも，センサーデータストリームの監視問題や蛋白質の三次元構造に対する検索などの実データに対する応用が今後の課題として挙げられる．

謝辞

今回，この研究をする機会を与えてくださり，夜遅くまで議論に付き合っていただいた有村博紀先生に深く感謝いたします．修士課程で執筆した論文はすべて先生の厚い助力があってこそでした．

喜田拓也先生には，研究のきっかけとなった講義をしていただいたことをはじめ，論文指導や研究議論の面でも大変お世話になりました．ここに厚く感謝いたします．

研究補助員の南野香子さんには日頃から数多くのサポートをしていただき，本当にお世話になりました．ありがとうございます．

トーマス・ツォイクマン先生と湊真一先生には本研究に関して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．

また，大学院修士課程に進学させてくれた両親に感謝します．最後に，執筆中に逝去した祖父に，この論文を捧げます．

参考文献

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] T. Asai, H. Arimura, K. Abe, S. Kawasoe and S. Arikawa, Online Algorithms for Mining Semi-structured Data Stream, Proc. *ICDM'02*, 27–34, 2002.

[3] R. S. Boyer and J. S. Moore, A Fast String Searching Algorithm, *CACM*, 762–772, 1977.

[4] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul and S. Zdonik, Monitoring Streams: A New Class of Data Management Applications, Proc. *VLDB'02*, 215–226, 2002.

[5] M. Crochemore and W. Rytter, *Text Algorithms*, Oxford University Press, 1994.

[6] L. Harada, Complex Temporal Patterns Detection over Continuous Data Streams, Proc. *ADBIS'02*, 401–414, 2002.

[7] L. Harada, Pattern Matching over Multi-attribute Data Streams, Proc. *SPIRE'02*, LNCS 2476, 187–193, 2002.

[8] D. E. Knuth, J. H. Morris Jr. and V. R. Pratt, Fast Pattern Matching in Strings, *SIAM J. Comput.*, 323–350, 1977.

[9] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, 1981.

[10] 森川，浅井，有村, データストリーム処理のための効率良い XPath 問合せ機構, Proc. *DBWS'03*, 211–218, 2003.

[11] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein and R. Varma, Query Processing, Approximation, and Resource Management in a Data Stream Management System, Proc. *CIDR'03*, 2003.

[12] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings*, Cambridge University Press, 2002.

[13] R. Sadri, C. Zaniolo, A. M. Zarkesh and J. Adibi, Optimization of Sequence Queries in Database Systems, Proc. *PODS'01*, ACM, 71–81, 2001.

[14] T. Saito, T. Kida and H. Arimura, An Efficient Algorithm for Complex Pattern Matching over Continuous Data Streams Based on Bit-parallel Method, Proc. *SWOD'07*, IEEE, 13–18, 2007.

[15] 斉藤，喜田，有村, ビット並列手法に基づく大規模連続ストリームパターン照合, 第 6 回情報科学技術フォーラム講演論文集 (FIT2007), 情報処理学会・電子情報通信学会, D-019, 2007.

[16] 斉藤，喜田，有村, ビット並列手法に基づく効率良い時系列パターン照合機構と多次元データストリーム処理への応用, 第 19 回データ工学ワークショップ (DEWS2008), 電子情報通信学会・日本データベース学会, E1-1, 2008.

[17] 斉藤，喜田，有村， 連続データストリームに対するビット並列手法を用いた高度な時系列パターン照合, 第 18 回データ工学ワークショップ (DEWS2007), 電子情報通信学会・日本データベース学会, C1-9, 2007.

[18] G. A. Stephen, *String Searching Algorithms*, World Scientific, 1994.

[19] M. Stonebraker and U. Cetintemel, "One Size Fits All": An Idea Whose Time Has Come and Gone, Proc. *ICDE'05*, 2–11, 2005.

[20] Y. Watanabe and H. Kitagawa, A Multiple Continuous Query Optimization Method Based on Query Execution Pattern Analysis, Proc. *DASFAA'04*, 443–456, 2004.