# データストリームのための部分シーケンスマッチング

豊田真智子† 櫻井 保志†† 市川 俊一†

† 日本電信電話株式会社 NTT 情報流通プラットフォーム研究所
〒 180–8585 東京都武蔵野市緑町 3–9–11
†† 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
〒 619–0237 京都府相楽郡精華町光台 2-4
E-mail: †{toyoda.machiko,ichikawa.toshikazu}@lab.ntt.co.jp, ††yasushi.sakurai@acm.org

**あらまし** 本論文では，データストリームにおける cross-similarity の問題を定義する．本論文の目的は，データストリームから類似する部分シーケンスペアとそれらの周期性を検出することである．シーケンス間の類似度を測定する距離尺度には，時間軸上でのスケーリングを考慮することができるダイナミックタイムワーピング (DTW: Dynamic Time Warping) 距離を利用する．我々の提案する CrossMatch は，厳密に DTW に基づいた手法であり，データストリーム処理に適したワンパスアルゴリズムである．DTW を用いた純粋なアルゴリズムと比べて，CrossMatch は計算コストとメモリ使用量の大幅な低減化を実現する．理論的な分析を行い，提案アルゴリズムが精度を犠牲にすることなく最適解を検出することを示す．また，実データと人工データを用いた実験から，CrossMatch がインクリメンタルに cross-similarity を検出し，時間変化する周期性を効率的に発見することが確認された．

**キーワード** データストリーム，部分シーケンスマッチング，DTW

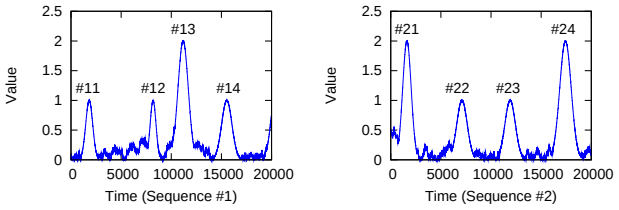
## 1. はじめに

データストリームは，金融やセンサネットワーク管理，移動体追跡，ネットワークトラフィックモニタリング，Web クリックストリーム分析など，様々な分野で重要となっている．データストリームは，無限長時系列データであり，高速かつ連続的にオンラインで到着するため，すべてのデータを保持することは困難である．そのため，データストリームを効率的に処理するためのマイニング技術に注目が集まっている．

データストリーム処理が必要となるアプリケーションにおいてしばしば求められるのは，連続的にデータストリームをモニタリングする技術であり，その重要な要素技術の 1 つが部分シーケンスマッチングである．部分シーケンスマッチングに関する先行研究の多くは，問合せシーケンスに類似する部分シーケンスをデータストリームから検出する問題に焦点を当てている [4] [7]．本論文では，データストリームにおける類似部分シーケンスペアを検出する問題に焦点を当てる．この問題は，頻出パターン検出やルール抽出，異常検出などに適用される重要な問題である．データストリームは，サンプリングレートが異なる場合や，時間周期が変化する場合があるため，これらに対応する距離尺度が必要となる．この問題を解決するために，データストリームにおける cross-similarity の問題を定義し，距離尺度としてダイナミックタイムワーピング (DTW: Dynamic Time Warping) 距離を利用する．DTW は様々な分野で幅広く利用されており，時間軸上でのスケーリングを考慮することが可能なため，データストリームにおける部分シーケンスマッチングに適した距離尺度である．

我々の目標は，データストリームをモニタリングし，すべての類似する部分シーケンスのペアを自動的に検出することである．本論文で解決したい問題は，次のようなものである．

2 つのデータストリームが与えられた時，DTW に基づき，類似する部分シーケンスペアやそれらの周期性を検出する．




(a) シーケンス #1　(b) シーケンス #2

図 1 部分シーケンスマッチングの例．これらのシーケンスには，大小のスパイクが存在し，それぞれの周期は異なっている．

図 1 を用いてこの問題を説明する．問題定義の詳細については，3 章で述べる．シーケンス# 1 は 3 つの小さなスパイクが存在し (# 11, # 12, # 14)，シーケンス# 2 は 2 つの小さなスパイクが存在する (# 22, # 23)．これらのスパイクの振幅はほぼ同じであるが，周期はそれぞれ異なっている．また，これらのシーケンスには 3 つの大きなスパイクも存在し (# 13, # 21, # 24)，これらの周期も異なっている．本論文で解決したい問題は，2 つのシーケンス間の部分的な類似を見つけることである．具体的には，部分シーケンスペア# 11 と# 22，# 11 と# 23，# 13 と# 21，# 13 と# 24 など，大小それぞれのシーケンスペアを意味し，これらはシーケンス# 1 と# 2 における類似部分シーケンスペアである．

従来，DTW は蓄積されたデータセットに対して利用されてきた．問合せシーケンスの長さが固定された従来の設定と異なり，データストリームは頻繁にデータが生成されるため，シーケンス長は時間の経過と共に発達する．そのため，類似度を求めるための計算量は大幅に増加する．この条件の下で高速処理を達成するため，従来のアルゴリズムはマッチングの精度を犠牲にしてきた．しかしながら，アルゴリズムは高い性能を維持しつつも，すべての正しい解を出力することが望ましい．

本論文では，データストリームにおける部分シーケンスマッチングのための手法である CrossMatch を提案する．2 つの伸張する長さ $n$ のシーケンス $X$ と長さ $m$ のシーケンス $Y$ において，ナイーブな手法は $O(m^2n + mn^2)$ の計算時間とメモリ空間を必要とする．一方 CrossMatch は，$O(m + n)$ の計算時間とメモリ空間で類似する部分シーケンスペアを検出することができるワンパスアルゴリズムである．また，CrossMatch は正しい解を検出漏れすることがない．理論的な分析と，実データ，人工データを用いた実験により，CrossMatch が精度を犠牲にすることなく，高速化を達成していることを示す．

本論文は次のように構成される．まず，2 章でストリームマイニングの関連研究を挙げる．3 章で問題定義を行い，4 章で提案手法である CrossMatch について述べる．5 章において，CrossMatch の有効性を検証するための実験について述べ，その結果を考察する．最後に 6 章でまとめを述べる．

## 2. 関連研究

データストリームにおけるパターン検出の研究としては，$L^p$ 距離に基づくストリームの相関検出や予測など，様々な手法が提案されている．Gao らが連続クエリを処理する方法を導入し，データストリームにおける類似パターンを見つける問題を扱った [1]．AWSOM はデータストリームのための最初の予測手法であり，時間シーケンスにおける任意の周期性を発見するものである [2]．Zhu らはリアルタイムで複数ストリームをモニタリングする問題に焦点を当て，すべてのストリーム間の相関を計算する StatStream を提案した [8]．SPRIT は，相関とトレンドに相当する隠れ値を検出する問題に取り組んだものである [3]．Braid はデータストリーム間の遅延相関を検出するための手法である [5]．しかしながら，これらの手法のどれも，我々が序章で述べた問題に取り組んでいるものではない．

また，データストリームにおける DTW を用いたシーケンスマッチングの研究としては，Zhou らが DTW 距離に基づいてオンラインで部分シーケンスを検出する手法を提案している [7]．櫻井らは DTW に基づいた複数数値ストリームを効率的にモニタリングする SPRING を提案している [4]．これらの手法は問合せシーケンスに類似する部分シーケンスを検出するためには有用であるが，本論文で扱う問題に適用した場合には計算量とメモリ使用量が増加する．

## 3. 問題定義

本章では，シーケンス間の類似度を測定するダイナミックタイムワーピング (DTW: Dynamic Time Warping) について説明し，本論文で解決したい問題について述べる．

### 3.1 ダイナミックタイムワーピング (DTW)

DTW は時間スケールを考慮した有用な距離尺度の 1 つである．シーケンス間の距離を最小化するように時間軸方向にシーケンス長を調整する性質を持っているため，シーケンス長やサンプリングレートが異なるシーケンス間の類似度を計算することができる．

2 つのシーケンス間の DTW 距離は動的計画法によって計算され，2 つのシーケンスを最適に調節した後の距離の合計となる．図 2 は DTW の計算方法を例示したものである．左図は，距離計算のための DTW による最適なシーケンスの対応付けを表している．距離計算には右図で示されるようなタイムワーピング行列が用いられる．距離値を計算するために対応付けられた要素同士の並びはワーピングパスと呼ばれ，右図では色付けされたセルの集合として示されている．

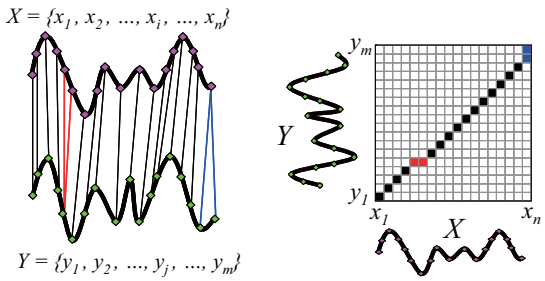



図 2　DTW による全体シーケンスマッチング．左図は DTW による シーケンス間の対応付けを，右図はタイムワーピング行列において，対応付けられた要素同士の組で表されるワーピングパスを示している．

長さ $n$ のシーケンス $X = (x_1, x_2, ..., x_n)$ と長さ $m$ のシーケンス $Y = (y_1, y_2, ..., y_m)$ を考える．これらの DTW 距離 $D(X, Y)$ は次のように定義される．

$$
\begin{aligned}
&D(X,Y) = d(n,m) \\
&d(i,j) = ||x_i - y_j|| + min \begin{cases} d(i,j-1) \\ d(i-1,j) \\ d(i-1,j-1) \end{cases} \qquad (1) \\
&d(0,0) = 0, \quad d(i,0) = d(0,j) = \infty \\
&(i = 1, ..., n;\ j = 1, ..., m)
\end{aligned}
$$

ここで，$||x_i - y_j|| = (x_i - y_j)^2$ は 2 つの数値の距離を表す．L1 距離 ($||x_i - y_j|| = |x_i - y_j|$) など，他の選択でもかまわない．本論文で提案するアルゴリズムは，これらの選択とは完全に独立したものである．

### 3.2 Cross-similarity

データストリーム $X$ は，$x_1, x_2, ..., x_n, ...$ の値からなる半無限長のシーケンスである．$x_n$ は最新の値であり，時間が進むごとに $n$ は増加する．$X[i_s : i_e]$ は時刻 $i_s$ から始まり $i_e$ で終わる $X$ の部分シーケンスであり，$X[i_s : i_e]$ の長さは $l_x = i_e - i_s + 1$ となる．同様に，$Y[j_s : j_e]$ は時刻 $j_s$ から始まり $j_e$ で終わる $Y$ の部分シーケンスであり，その長さは $l_y = j_e - j_s + 1$ となる．本論文の目的は，DTW に基づき，データストリーム処理でシーケンス間の部分的な類似を見つけることである．より具体的には，次の条件を満たす部分シーケンスペアを検出する．

$$D(X[i_s : i_e], Y[j_s : j_e]) \leqq \varepsilon L(l_x, l_y) \qquad (2)$$

ここで，$D(X[i_s : i_e], Y[j_s : j_e])$ は部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ 間の DTW 距離であり，$L$ はこれら部分シーケンスペアの長さを表す関数である．本論文では，2 つの部分シーケンスの平均長である $L(l_x, l_y) = (l_x + l_y)/2$ を用いるが，他の選択でもかまわない (すなわち，$L(l_x, l_y) = max(l_x, l_y)$ や $L(l_x, l_y) = min(l_x, l_y)$)．DTW 距離は，対応付けられた要素間の距離の合計で表されるため，部分シーケンスの長さが長くなるにつれて、その値は増加する．そのため，類似判定の閾値も部分シーケンス長に比例することが望ましいと言え，$\varepsilon L(l_x, l_y)$ を用いて判定を行なう．

ここで，$X$ と $Y$ 間の部分的な類似についてフォーマルに定義する．

［定義 1］（Cross-similarity） 2 つのシーケンス $X$ と $Y$, 距離の閾値 $\varepsilon$, 適合する部分シーケンスの長さの閾値 $l_{min}$ が与えら

れた時，部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ は次の条件を満たす.

$$D(X[i_s : i_e], Y[j_s : j_e]) \leqq \varepsilon(L(l_x, l_y) - l_{min}) \tag{3}$$

適合する部分シーケンスの長さの閾値 $l_{min}$ は、ユーザによって与えられる値である．定義 1 は，オリジナルの DTW に，適合するシーケンスの長さという概念を加えたものである．すなわち，$l_{min}$ 以上の長さの部分シーケンスペアを検出することを意味する．オリジナルの DTW は，ノイズなどによってシーケンス長の短い，意味のないシーケンスペアを適合ペアとして検出する可能性がある．そのため，これらを排除することにより，ユーザの真の要求を満たすシーケンスペアを検出する.

部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ が適合する時，極小値をとる最適解と区間が重複 (overlap) する他の多くの部分シーケンスペアも適合してしまう．ここで重複とは，部分シーケンスペア間のワーピングパスが交わることである．ワーピングパスの重複には，開始点が異なる部分シーケンスペアのワーピングパスが途中で交差する場合と，交差したワーピングパスが途中で分離する場合が該当する．本論文では，重複する部分シーケンスペアの中から距離値が極小値をとる部分シーケンスペアを検出する．すなわち，本論文において解決したい問題は，cross-similarity の最適解を見つけることである.

［問題 1］ 2 つのシーケンス $X$ と $Y$，閾値 $\varepsilon$，適合する部分シーケンスの長さの閾値 $l_{min}$ が与えられた時，次の条件を満たす部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ を検出する.

（1） $X[i_s : i_e]$ と $Y[j_s : j_e]$ は cross-similarity の性質を持つ.

（2） ワーピングパスが交差する部分シーケンスペアのグループの中で，$D(X[i_s : i_e], Y[j_s : j_e]) - \varepsilon(L(l_x, l_y) - l_{min})$ が最小値をとる.

これ以降，問題 1 の条件 (1) を満たすものを適合する部分シーケンスペア，2 つの条件を満たすものを最適な部分シーケンスペアと呼ぶ.

本論文で示した問題は従来の時系列分析の問題とは異なり，シーケンス間の部分的な類似を DTW に基づいて検出することに焦点を当てている．理想的には，アルゴリズムはサンプリングレートが異なるシーケンスや時間周期が変化するシーケンスについてもロバストに対応し，データストリーム処理の中でも正しい解を出力することを保証するべきである．データストリーム処理は低メモリと高スピード処理が要求されるため，解決したい問題はより困難なものとなる.

## 4. 提案手法

本章では，データストリームにおける cross-similarity の最適解を見つけるためのワンパスアルゴリズム，CrossMatch について述べる.

### 4.1 ナイーブな手法

cross-similarity を見つけるための最も素直な方法は，データストリーム $X$ と $Y$ の考えられるすべての部分シーケンスの組み合わせについて，それらの距離を DTW によって計算することである．本論文ではこの手法をナイーブな手法と呼ぶ.

$X$ の $i$ 番目の要素と $Y$ の $j$ 番目の要素から始まる $X$ と $Y$ のタイムワーピング行列におけるセル $(p, q)$ が示す距離を $d_{i,j}(p, q)$ とする．$X$ と $Y$ 間の部分シーケンスマッチングの距離は次のように計算される.

$$\begin{aligned}
&D(X[i_s : i_e], Y[j_s : j_e]) = d_{i_s, j_s}(l_x, l_y) \\
&d_{i,j}(p, q) = ||x_{i+p-1} - y_{j+q-1}|| + d_{best} \\
&d_{best} = \begin{cases} d_{i,j}(p, q-1) \\ d_{i,j}(p-1, q) \\ d_{i,j}(p-1, q-1) \end{cases} \\
&d_{i,j}(0, 0) = 0, \quad d_{i,j}(p, 0) = d_{i,j}(0, q) = \infty \\
&(i = 1, ..., n;\ p = 1, ..., n-i+1; \\
&\qquad j = 1, ..., m;\ q = 1, ..., m-j+1)
\end{aligned} \tag{4}$$

ナイーブな手法では，すべての考えられる部分シーケンスについて，時刻 $i$ で流入する $x_i$ と時刻 $j$ で流入する $y_j$ のタイムワーピング行列の距離を更新し，重複する部分シーケンスペアのグループの中で $D(X[i_s : i_e], Y[j_s : j_e]) - \varepsilon(L(l_x, l_y) - l_{min})$ が最小値となる部分シーケンスペア 1 つを選択する.

タイムワーピング行列については毎時刻，2 列 (現在の列と直前の列) の距離値，すなわち $O(m + n)$ の距離値が更新される．さらに，ナイーブな手法は新たな行列を毎時刻作成するため，$O(mn)$ 個の行列を扱うことになる．そのため合計すると，$O(m^2n + mn^2)$ 個もの値を更新しなければならない．ナイーブな手法は $O(m^2n + mn^2)$ の計算時間とメモリ使用量を必要とするが，本論文では，この膨大なコストを精度を犠牲にすることなく劇的に削減できることを示す.

### 4.2 基本的なアイデア

本節では Cross-similarity の最適解を見つけるための我々の手法，CrossMatch について述べる．CrossMatch は，cross-similarity を求めるための新しいスコアリング関数と位置行列，そしてそれらを用いたストリーム処理アルゴリズムの 3 つのアイデアに基づいている.

［知見 1］ 累積スコア値を計算するスコアリング関数により，1 つの行列のみで cross-similarity を発見することができる.

第 1 のアイデアとして，累積スコア値を計算する関数をシーケンスマッチングに導入する．ナイーブな手法は，毎時刻，新しいタイムワーピング行列を作成する．$O(mn)$ の行列を必要とするナイーブな手法を用いる代わりに，我々はスコアリング関数を用い，$X$ と $Y$ の部分シーケンスペアの DTW 距離を 1 つの行列のみで計算する．我々のスコアリング関数は以下の特徴を持つ.

（1） スコアリング関数は非負の累積スコア値を出力する.

（2） スコアリング関数の操作は DTW 距離に関して可逆である.

スコア計算は DTW の距離計算と同様に動的計画法に基づいて計算を行うが，その違いは DTW が最小の累積距離を計算するのに対し，我々が提案するスコアリング関数は最大の累積スコアを計算することにある．すなわち，類似したシーケンスペアは高いスコア値を示す．累積スコア値は非負であり，行列 (すなわち，スコア行列) のスコア値がもし負であれば 0 に置き換えられる．つまり，定義 1 を満たす可能性のない部分シーケンスペアについては，スコア行列の中で初期化される．この性質によって，適合する部分シーケンスペアを効率的に追い求めることが可能となる.

さらに重要なことに，非負の累積スコアを出力するスコアリング関数の操作は，DTW 距離に関して可逆である．すなわち，部分シーケンスペア間の類似度は，スコアリング関数によりスコア値として表現されるが，このスコア値を容易に DTW 距離

に変換することが可能である．

スコアリング関数により，部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ 間の cross-similarity の存在を知ることが可能となった．しかし，このアイデアのみでは問題を解決することはできない．単にスコアリング関数のみを用いた場合，部分シーケンスペアの開始点に関する情報が失われてしまう．すなわち，1 つの行列を用いることによって適合する部分シーケンスペアのスコアを得ることができるが，どの部分シーケンスペアが最大スコアを出力したのかを判断することができなくなる．そこで，我々は第 2 のアイデアとして，部分シーケンスペアの開始点を保持するための位置行列を導入する．

［知見 2］ 位置行列には部分シーケンスペアの開始点が保持される．スコア行列と位置行列を用いることによって，ワンパスの処理で適合する部分シーケンスペアを特定することができる．

検出すべき部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ については，スコア値 $v(i_e, j_e)$ がスコア行列のセル $(i_e, j_e)$ に，開始点 $s(i_e, j_e)$ が位置行列のセル $(i_e, j_e)$ に保存される．$s(i_e, j_e)$ は，$X[i_s : i_e]$ と $Y[j_s : j_e]$ のマッチングの開始点 $(i_s, j_s)$ を指し示す．スコア行列のスコア値だけでなく，位置行列に格納されている部分シーケンスペアの開始点も更新する．このことによって，どの部分シーケンスペアが最大スコアを出力したのかをストリーム処理の間も認識することができる．

［知見 3］ スコアリング関数とスコア行列，位置行列はデータストリーム処理に適用することができる．検出漏れがないことを保証しながらも $O(m+n)$ の計算時間とメモリ使用量で処理を行う．

最後に，上記で述べたスコアリング関数とスコア行列，位置行列をデータストリーム処理に適用する．我々の提案するワンパスアルゴリズムは 2 つの行列しか使わないため，単位時間あたり $O(m+n)$ のスコア値と開始点を更新するだけで最適な部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ を検出することができる．また，アルゴリズムはデータストリーム処理の中でも検出漏れを起こさないように注意深く設計されている．スコアリング関数が出力するスコア値は DTW 距離に対して可逆である．出力結果の厳密性を保証するため，我々はこの可逆性を利用する．具体的には，まずアルゴリズムは $X[i_s : i_e]$ と $Y[j_s : j_e]$ の開始点 $(i_s, j_s)$ と終了点 $(i_e, j_e)$，累積スコア値 $V(X[i_s : i_e], Y[j_s : j_e])$ を求める．そして，累積スコア値と部分シーケンス長から以下のように $X[i_s : i_e]$ と $Y[j_s : j_e]$ の DTW 距離 $D(X[i_s : i_e], Y[j_s : j_e])$ を計算し，cross-similarity の最適解として出力する．

$$D(X[i_s : i_e], Y[j_s : j_e]) = \varepsilon L(l_x, l_y) - V(X[i_s : i_e], Y[j_s : j_e]) \tag{5}$$

ここで，図 3 を用いて CrossMatch を説明する．シーケンスの要素が到着する度，スコア行列の 1 列が計算され，位置行列の 1 列が更新される．条件を満たす部分シーケンスペアが検出された場合，そのペアを最適な部分シーケンスペア (cross-similarity の最適解) として報告する．図の色付けされたセルは，スコア行列における最適な部分シーケンスペアのワーピングパスを表している．例えば図 3 では，実線で囲まれた 6 つのワーピングパスが，$X$ と $Y$ に存在する小さなスパイクの cross-similarity を，点線で囲まれた 2 つのワーピングパスが，大きなスパイクの cross-similarity を表している．これらのワーピングパスは規則正しく現れており，周期性を捉えている．

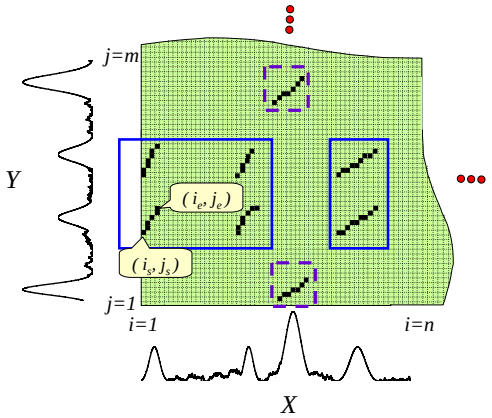



図 3 CrossMatch による部分シーケンスマッチング．我々の手法は，1 つの行列のみで cross-similarity の最適解をワンパスで検出することができる．

### 4.3 CrossMatch

本節では我々のアルゴリズムについて詳細に述べる．

#### 4.3.1 スコアリング関数

2 つのシーケンス $X = (x_1, \ldots, x_i, \ldots, x_n)$ と $Y = (y_1, \ldots, y_j, \ldots, y_m)$ が与えられた時，$X[i_s : i_e]$ と $Y[j_s : j_e]$ のスコア値 $V(X[i_s : i_e], Y[j_s : j_e])$ は次のように計算される．

$$V(X[i_s : i_e], Y[j_s : j_e]) = v(i_e, j_e)$$

$$v(i,j) = max \begin{cases} 0 \\ w_v \varepsilon + v(i, j-1) - ||x_i - y_j|| \\ w_h \varepsilon + v(i-1, j) - ||x_i - y_j|| \\ w_d \varepsilon + v(i-1, j-1) - ||x_i - y_j|| \end{cases} \tag{6}$$

$$v(0,0) = v(i,0) = v(0,j) = 0$$

最適な部分シーケンスペアを効率的に追い求めるため，累積スコアが負の値であるときにはスコアは初期化され，現在のセルからスコアを再計算する．これは，隣接するセルのすべてが負の値である場合，セル $(i, j)$ で終了する $X$ と $Y$ の部分シーケンスペアがもはや定義 1 を満たす可能性がないことを意味する．そのため，これまでのスコアを引き継がず，$(i, j)$ から始まる新たなマッチング処理を開始する．ナイーブな手法が単位時間あたり $O(mn^2 + m^2 n)$ 個の距離値を更新するのに対し，スコアリング関数は $O(m+n)$ のスコア値しか更新しないため，計算時間とメモリ量の大幅な低減化につながる．

またスコアリング関数の操作は DTW 距離に関して可逆である．式 (6) において，$w_v$，$w_h$，$w_d$ は $L$ によって決定される重みである．定義 1 において示したように，DTW 距離の閾値は部分シーケンス長 $L(l_x, l_y)$ に比例する．例えば，$L(l_x, l_y) = (l_x + l_y)/2$ において，$w_v = w_h = 1/2$，$w_d = 1$ となる(注1)．これは，垂直方向または水平方向の要素が引き継がれた場合，シーケンス長は 1/2 増加し，対角方向の要素が引き継がれた場合は 1 増加するためである．スコア行列の中での各ワーピングパス上の重み (すなわち $w_v$，$w_h$，$w_d$) の合計は，$L(l_x, l_y)$ と等しくなるように設計されており，DTW とスコアリング関数の可逆の関係が保証される．

(注1)：$L(l_x, l_y) = max(l_x, l_y)$ において，$l_x > l_y$ の場合，$w_d = w_h = 1$，$w_v = 0$，$l_x < l_y$ の場合，$w_d = w_v = 1$，$w_h = 0$，$l_x = l_y$ の場合，$w_d = 1$，$w_v = w_h = 0$ となる．$L(l_x, l_y) = min(l_x, l_y)$ についても同様に重みを決定することができる．

**Algorithm**
**Input:** a new value $x_i$ at time-tick $i$
**Output:** optimal subsequence pairs and DTW distances

```
//Detect optimal subsequence pairs.
for j := 1 to m do
  C'_v := v(n,j);   //Score value derived by Equation (6)
  C'_s := s(n,j);   //Starting position derived by Equation (7)
  C'_e := (n,j);    //End position
  if C'_v ≧ εl_min then
    //Add the subsequence pair as a new candidate.
    if C'_s ∉ S then
      add C' to S;
    else
      for each candidate C ∈ S do
        //Overwrite the maximum score.
        if C'_s = C_s ∧ C'_v ≧ C_v then
          C_v := C'_v;
          C_e := C'_e;
        endif
      endfor
    endif
  endif
endfor
//Report the optimal subsequence pairs.
for each candidate C ∈ S do
  if (∀_i, s(i,m) ≠ C_s) ∧ (∀_j, s(n,j) ≠ C_s) then
    d_min := εL(l_x, l_y) − C_v;
    Report d_min, C_s and C_e;
    Remove C from S;
  endif
endfor
```

図 4　ストリーム処理アルゴリズム. 重複したものの中から最適な部分シーケンスペアを出力する.

#### 4.3.2　位置行列

我々は，部分シーケンスペアのスコア値と開始点それぞれについて，$m \times n$ 行列を一つずつ用いる．スコア値 $v(i,j)$ がスコア行列のセル $(i,j)$ に，開始点 $s(i,j)$ が位置行列のセル $(i,j)$ に格納される $(i = 1,\ldots,n; j = 1,\ldots,m)$．位置行列の開始点 $s(i,j)$ は以下のように求められる．

$$s(i,j) = \begin{cases} s(i,j-1) & (v(i,j-1) > 0 \ \wedge \ v(i,j) \\ & \quad = w_v \varepsilon + v(i,j-1) - ||x_i - y_j||) \\ s(i-1,j) & (v(i-1,j) > 0 \ \wedge \ v(i,j) \\ & \quad = w_h \varepsilon + v(i-1,j) - ||x_i - y_j||) \\ s(i-1,j-1) & (v(i-1,j-1) > 0 \ \wedge \ v(i,j) \\ & \quad = w_d \varepsilon + v(i-1,j-1) - ||x_i - y_j||) \\ (i,j) & (otherwise) \end{cases} \tag{7}$$

スコア計算において，垂直方向，水平方向，対角方向のいずれかの要素が選択された場合 (すなわち，$v(i,j-1)$, $v(i-1,j)$, $v(i-1,j-1)$ のいずれかが $v(i,j)$ に加算された場合)，同じ方向が位置行列においても選択され，選択された要素が保持している開始点が引き継がれる．スコア行列においていずれの方向の要素も 0 である場合，位置行列では開始点として $(i,j)$ が選択される．スコア計算と開始点の更新は，各々の行列において完全に同じワーピングパス上を伝搬し，最大スコアを示す最適な部分シーケンスペアの開始点を得ることができる．

#### 4.3.3　ストリーム処理アルゴリズム

ストリーム処理アルゴリズムを図 4 に示す．時刻 $i$ において $x_i$ が到着する度に，スコア $v(n,j)$ と開始点 $s(n,j)$ を式 (6) と (7) を用いてインクリメンタルに計算する．そして cross-similarity の条件 (定義 1) を満たす部分シーケンスペアを検出

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $y_7$ = 13 | 0 | 22 | 0 | 30 | 0 | 0 | 0 |
| $y_6$ = 9 | 2 | 16 | 17 | 32 | 39 | 46 | 0 |
| $y_5$ = 2 | 11 | 0 | 19 | 0 | 41 | 55 | 0 |
| $y_4$ = 4 | 13 | 0 | 28 | 11 | 50 | 56 | 0 |
| $y_3$ = 9 | 4 | 18 | 25 | 40 | 38 | 29 | 0 |
| $y_2$ = 6 | 13 | 0 | 27 | 18 | 27 | 33 | 0 |
| $y_1$ = 11 | 0 | 13 | 0 | 13 | 0 | 0 | 0 |
| $X_i$ | 5 | 12 | 6 | 10 | 6 | 5 | 21 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

(a) スコア行列

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $y_7$ = 13 | (1,7) | (1,4) | (3,7) | (1,2) | (5,7) | (6,7) | (7,7) |
| $y_6$ = 9 | (1,4) | (1,4) | (1,2) | (1,2) | (2,1) | (2,1) | (7,6) |
| $y_5$ = 2 | (1,4) | (2,5) | (1,2) | (4,5) | (2,1) | (2,1) | (7,5) |
| $y_4$ = 4 | (1,4) | (2,4) | (1,2) | (2,1) | (2,1) | (2,1) | (7,4) |
| $y_3$ = 9 | (1,2) | (1,2) | (2,1) | (2,1) | (2,1) | (2,1) | (7,3) |
| $y_2$ = 6 | (1,2) | (2,2) | (2,1) | (2,1) | (4,1) | (4,1) | (7,2) |
| $y_1$ = 11 | (1,1) | (2,1) | (3,1) | (4,1) | (5,1) | (6,1) | (7,1) |
| $X_i$ | 5 | 12 | 6 | 10 | 6 | 5 | 21 |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

(b) 位置行列

図 5　cross-similarity の検出の例．ストリーム処理アルゴリズムはこれら 2 つの行列を用いて最適な部分シーケンスペアを検出する．

すると，そのスコア $C'_v = v(n,j)$，開始点 $C'_s = s(n,j)$，終了点 $C'_e := (n,j)$ を候補集合の配列 $\mathcal{S}$ に格納する．

区間が重複する部分シーケンスペアが存在する場合，候補配列には重複する候補ペアのグループの中で 1 つのペアのみが格納される．すなわち，$v(n,j) - \varepsilon l_{min}$ が最大値となる候補部分シーケンスペアが選択され，格納される．

そして，以下の 2 つの条件を満たす時，その候補部分シーケンスペアを cross-similarity の最適解として報告する．

（1）　$\forall_i, s(i,m) \neq C_s$
（2）　$\forall_j, s(n,j) \neq C_s$

これは，候補部分シーケンスペアが今後出現する部分シーケンスペアによって置き換わることがないことを意味する．アルゴリズムは部分シーケンスペアの類似度として DTW 距離を報告する．DTW 距離は図 4 に示しているように累積スコア値と部分シーケンス長から簡単に求めることができる．

このアルゴリズムでは，時刻 $i$ で $X$ の要素 $x_i$ を受け取った場合のストリーム処理に焦点を当てているが，時刻 $j$ で受け取った $Y$ の要素 $y_j$ の処理についても，同様に計算することができる．

図 5 は，$X = (5,12,6,10,6,5,18)$, $Y = (11,6,9,4,2,9,13)$ の 2 つのシーケンスについて，距離の閾値 $\varepsilon = 14$，適合する部分シーケンスの最小長 $l_{min} = 3$ を設定した場合に計算されるスコア行列の例である．アルゴリズムはセル $(i,j)$ にスコア $v(i,j)$ と開始点 $s(i,j)$ を保持する．濃く色付けされたセルは最適な類似部分シーケンスペア（すなわち，cross-similarity の最適解) を表しており，薄く色付けされたセルは適合する部分シーケンスペア (すなわち，cross-similarity) を表している．

ここで，図 5 を用いてアルゴリズムの動作を説明する．説明を単純にするために，$x_i$ と $y_j$ が交互に到着すると想定する．$i = 5$ において，$v(5,4) = 50 \geqq \varepsilon l_{min}$ となる $(2,1)$ から始まる適合する部分シーケンスペア $X[2:5]$ と $Y[1:4]$ を検出する．$j = 5$ において，条件を満たす部分シーケンスペアは検出されないが，これから出現する部分シーケンスペアが最適な部分シーケンスペアになる可能性があるため，$X[2:5]$ と $Y[1:4]$ を報告しない．$i = 6$ において，これらのグループの最適な部分シーケンスペア $X[2:6]$ と $Y[1:4]$ を検出する．最終的に $j = 7$ において，今後出現する部分シーケンスペアが最適な部分シーケンスペアとなることはないことが確認され，部分シーケンスペア $X[2:6]$ と $Y[1:4]$ を cross-similarity の最適解として報告する．

### 4.4　理論的な分析

#### 4.4.1　精　　度

［補題 1］　2 つのシーケンス $X$ と $Y$ が与えられた時，問題 1 は次の条件と等価である．

（1）　$V(X[i_s : i_e], Y[j_s : j_e]) \geqq \varepsilon l_{min}$
（2）　ワーピングパスが交差する部分シーケンスペアのグ

ループの中で，$V(X[i_s : i_e], Y[j_s : j_e]) - \varepsilon l_{min}$ が最大値をとる．

［証明 1］ タイムワーピング行列とスコア行列の双方で，$(i_s, j_s)$ を開始点，$(i_e, j_e)$ を終了点とするワーピングパスが $(i_e, j_e - 1)$ を通るとするならば，式 (4) より

$$||x_{i_e} - y_{j_e}|| = d_{i_s,j_s}(l_x, l_y) - d_{i_s,j_s}(l_x, l_y - 1).$$

また，$w_v = L(l_x, l_y) - L(l_x, l_y - 1)$ であるため，式 (6) より，

$$\begin{aligned} ||x_{i_e} - y_{j_e}|| = \ & \varepsilon L(l_x, l_y) - v(i_e, j_e) \\ & -\varepsilon L(l_x, l_y - 1) + v(i_e, j_e - 1) \end{aligned}$$

が成り立つ．同様に，$(i_e - 1, j_e)$ と $(i_e - 1, j_e - 1)$ を通るときは各々

$$\begin{aligned} ||x_{i_e} - y_{j_e}|| &= d_{i_s,j_s}(l_x, l_y) - d_{i_s,j_s}(l_x - 1, l_y) \\ &= \varepsilon L(l_x, l_y) - v(i_e, j_e) \\ &\quad -\varepsilon L(l_x - 1, l_y) + v(i_e - 1, j_e) \\ ||x_{i_e} - y_{j_e}|| &= d_{i_s,j_s}(l_x, l_y) - d_{i_s,j_s}(l_x - 1, l_y - 1) \\ &= \varepsilon L(l_x, l_y) - v(i_e, j_e) \\ &\quad -\varepsilon L(l_x - 1, l_y - 1) + v(i_e - 1, j_e - 1) \end{aligned}$$

が成り立つ．

$$||x_{i_s} - y_{j_s}|| = d_{i_s,j_s}(1, 1) = \varepsilon - v(i_s, j_s)$$

であるため

$$d_{i_s,j_s}(l_x, l_y) = \varepsilon L(l_x, l_y) - v(i_e, j_e) \tag{8}$$

となり，問題 1 の条件 (1) より，

$$v(i_e, j_e) \geqq \varepsilon l_{min}$$

が得られる．

タイムワーピング行列では，問題 1 条件 (2) より，$(i_s, j_s)$ から $(i_e, j_e)$ までの最適ワーピングパスはワーピングパスが交差する部分シーケンスペアのグループの中で最小距離を示すことが明らかである．また式 (6) と式 (8) より，スコア行列でも同じワーピングパスが選択され，最大スコアを出力することがわかる．よって，問題 1 の 2 つの条件と等価となる．□

［補題 2］ CrossMatch は，探索漏れを発生させずに cross-similarity を満たす 最適な部分シーケンスペアを検出する．

［証明 2］ 最適な部分シーケンスペア $X[i_s : i_e]$ と $Y[j_s : j_e]$ のワーピングパ スの開始点を $C_s = (i_s, j_s)$ とする．$s(n, m) \neq C_s$ であるとき，部分シーケンスペア $X[i : n]$ と $Y[j : m]$ のワーピングパスは最適な ワーピングパスと重複しない $(1 \leqq i \leqq n, 1 \leqq j \leqq m)$．同様に，以下の条件を満たすとき，今後出現する部分シーケンスペアは 候補配列の中の部分シーケンスペアと重複することはない．

$$(\forall_i, s(i, m) \neq C_s) \ \wedge \ (\forall_j, s(n, j) \neq C_s).$$

CrossMatch は上記の条件を満たすときのみ $X[i_s : i_e]$ と $Y[j_s : j_e]$ を最適な部分シーケンスペアとして報告する．よって，CrossMatch は最適な部分シーケンスペアを見落とすことはない． □

表 1 データセットの詳細と実験パラメータ設定

| Data sets | Seq#1 length | Seq#2 length | $\varepsilon$ | $l_{min}$ |
|---|---|---|---|---|
| *Sines* | 25000 | 25000 | 5.5e-5 | 500 |
| *Spikes* | 28000 | 28000 | 1.0e-6 | 1100 |
| *Automobile traffic* | 16000 | 16000 | 8.5e+4 | 2000 |
| *Web* | 32000 | 32000 | 4.5e+4 | 8640 |
| *Sunspots* | 18000 | 18000 | 2.5e+2 | 2000 |

#### 4.4.2 計 算 量

$X$ を伸張している長さ $n$ のシーケンス，$Y$ を伸張している長さ $m$ のシーケンスとする．

［補題 3］ ナイーブな手法は，$O(m^2n + mn^2)$ のメモリ量と $O(m^2n + mn^2)$ の計算時間を要する．

［証明 3］ 最適な部分シーケンスペアを特定するために，ナイーブな手法は $O(mn)$ 個のタイムワーピング行列を用いる．時刻 $i$ において $x_i$ を受け取った場合には $O(m^2n)$ 個の値を，時刻 $j$ において $y_j$ を受け取った場合には $O(mn^2)$ 個の値を更新する必要がある．そのため，$O(m^2n + mn^2)$ の計算時間を要する．また，各行列毎に $m$ 個の配列 2 列と $n$ 個の配列 2 列を保持するので，全体として $O(m^2n + mn^2)$ のメモリ使用量が必要となる．□

［補題 4］ 提案手法は，$O(m + n)$ のメモリ使用量と $O(m + n)$ の計算時間を要する．

［証明 4］ 最適な部分シーケンスペアを特定するために，提案手法は 2 個の行列 (スコア行列と位置行列) を用いる．時刻 $i$ において $x_i$ を受け取った場合には $O(m)$ 個の値を，時刻 $j$ において $y_j$ を受け取った場合には $O(n)$ 個の値を更新する．そのため，$O(m + n)$ の計算時間と $O(m + n)$ のメモリ使用量を必要とする． □

## 5. 評 価 実 験

CrossMatch の有効性を検証するため，実データと人工データを用いた実験を行った．CrossMatch に対する比較手法には，ナイーブな手法と SPRING を用いた．SPRING は文献[4]で提案され，データストリームから固定長の問合せシーケンスに類似する部分シーケンスを検出するための手法である．cross-similarity を検出するための SPRING のアルゴリズムは，$O(m^2 + mn)$ のメモリ使用量と $O(m^2 + mn)$ の時間が必要となる．実験には，4GB のメモリと Intel Xeon E5335 2 GHz の CPU を搭載した Linux マシンを用いた．

### 5.1 cross-similarity の検出

データストリーム間の最適な部分シーケンスペア (すなわち，cross-similarity の最適解) の検出について，実データと人工データを用いたケーススタディを示す．各データセットの詳細と実験の設定は表 1 の通りである．なお，実験結果を示す図 6 は，左と中央の図がデータセットを，右の図がこれらのデータセットから検出された cross-similarity の最適なワーピングパスを示している．

#### 5.1.1 *Sines*

*Sines* はホワイトノイズを持つ不連続のサイン波から構成された人工データである (図 6 (a))．各シーケンスにおけるサイン波の周期は一致しておらず，サイン波が現れる間隔も異なっている．図 6 (a) の右図に示す通り，CrossMatch はすべてのサイン波と時間変化している周期性を完全に特定していることが分かる．各サイン波の周期の違いは，描画の傾きの違いとして表れている．

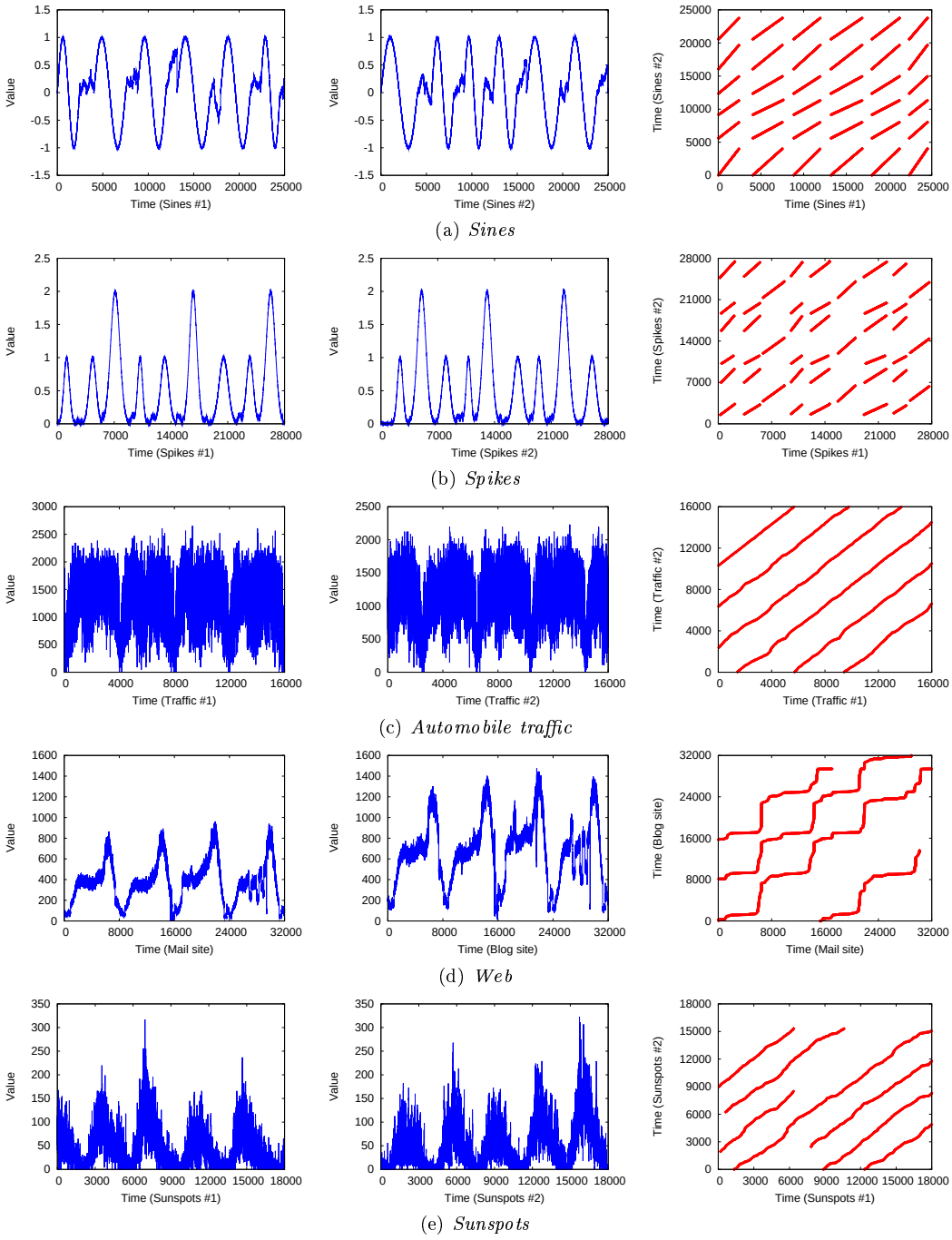



図 6　*Sines*，*Spikes*，*Automobile traffic*，*Web*，*Sunspots* を用いた cross-similarity の発見．左と中央の図がデータシーケンスであり，右の図が検出された cross-similarity の最適なワーピングパスを示している．

**5.1.2**　*Spikes*

*Spikes* は図 6 (b) で示される人工データセットであり，大小のスパイクから構成される．スパイクとスパイクの間のデータは，ランダムウォーク関数を用いて異なる長さで生成し，各スパイクの周期も異なっている．*Spikes* を用いた実験結果は，図 6 (b) の右図のようになっており，実験結果は CrossMatch が大小のスパイクを完全に検出している様子を示している．各スパイクの周期の違いは，描画の長さの違いとして現れており，幅の広いスパイクは描画長が長く，幅の狭いスパイクは描画長が短くなっている．

**5.1.3**　*Automobile traffic*

図 6 (c) は自動車交通量の時系列データであり，1 日の周期と朝夕のラッシュアワーを示す半日周期が存在する．時間単位の交通量はバースト的であり，ホワイトノイズとみなすことができる．

CrossMatch は高周波である時間単位の交通量に惑わされることなく，1 日の周期を完璧に検出することに成功している．図 6 (c) の右図において，描画線が連続していることと，それらが一定間隔で現れていることが，1 日の周期が繰り返されている様子を表している．また，描画線同士の間隔が 1 日の周期と対応しており，*Automobile traffic* データの特徴がよく反映されている．

**5.1.4**　*Web*

*Web* は，Web サイトにおけるアクセス数を 10 秒毎に記録し

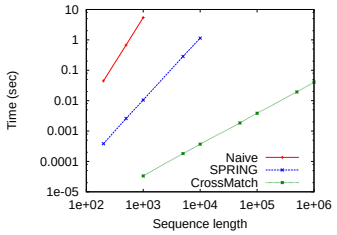
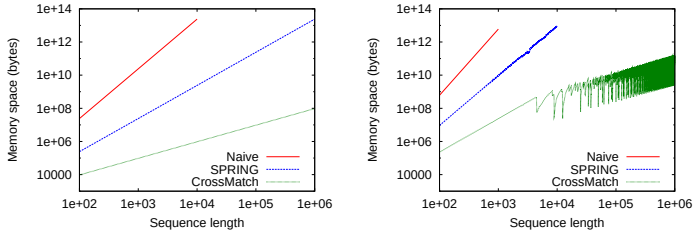



図 7　データシーケンス長に対する計算時間．CrossMatch は大幅な高速化を達成している．




(a) ワーピングパスを保持しない場合　(b) ワーピングパスを保持する場合

図 8　データシーケンス長に対するメモリ使用量．ワーピングパスを保持した場合であっても，CrossMatch は Naive や SPRING より少ないメモリであることが示されている．

たデータセットである．各サイト毎にアクセスパターンが異なる中，CrossMatch により cross-similarity が検出されたのが，メールサイトとブログサイトのアクセスパターンである (図 6 (d))．これらのデータはアクセス数のスケールが若干異なるが，朝からアクセスが上昇し，夜にかけてピークを迎えるという，よく似た 1 日の周期が存在する．

実験結果から，CrossMatch はこれらの周期性の検出にほぼ成功していることが確認される．*Automobile traffic* と異なり，cross-similarity の結果は曲がりくねった線を描いている．これは，CrossMatch がデータシーケンスの要素同士を時間軸方向に伸張させながら最適に対応付けたためであり，CrossMatch の時間スケールを考慮するという特徴によって cross-similarity が検出されていることがわかる．

#### 5.1.5　*Sunspots*

図 6 (e) のデータ集合は，1 日毎の太陽の黒点数を記録したものである．太陽の黒点には周期性があることがよく知られており，太陽の活動とも密接に関連している．太陽活動が活発な時は黒点が多く出現し、逆に太陽活動が不活発な時は黒点が減少する．この変化は約 11 年の周期で増減する．CrossMatch は黒点数の各周期の増減を区別し，類似する変化を示す周期を捉えている．

### 5.2　性　　能

4.4.2 節で議論した計算量を検証するために，シーケンス長を変化させた場合の計算時間，メモリ使用量を測定した．実験には *Sines* データセットを用いた．各実験では，CrossMatch，ナイーブな手法 (Naive)，SPRING の 3 手法についての比較を行っている．

図 7 は計算時間に関する比較結果である．実験結果から，CrossMatch の性能が，ナイーブな手法，SPRING と比べて非常に高いことが分かる．この傾向は 4.4.2 節における理論的な議論と合致しており，計算コストの大幅な低減化を達成している．

図 8 は，各行列のメモリ使用量を比較した結果である．図の横軸は，2 つのデータストリームのシーケンス長を表している．図 8 (a) は，タイムワーピング行列，スコア行列，位置行列のためのメモリ使用量を示している．本論文では，最適な部分シーケンスペアの距離値と位置情報のみを出力すること想定してきたが，CrossMatch は，最適な部分シーケンスペアのワーピングパスに関する情報も出力することができる．図 8 (b) は，最適な部分シーケンスペアのワーピングパスを出力するために必要となるメモリ使用量を示している．ワーピングパスの情報を保持する場合，必要となるメモリ使用量は増加する．しかしながら CrossMatch は，ワーピングパスの情報を保持した場合でも，ナイーブな手法や SPRING よりもメモリ使用量の低減化を達成している．また，これらの結果も，4.4.2 節で議論した通りのメモリ使用量であることが確認される．

## 6.　おわりに

本論文では，データストリームにおける cross-similarity の問題を扱い，その問題を解決するための手法である CrossMatch を提案した．CrossMatch は DTW に基づく手法であり，cross-similarity を検出するための新しいスコアリング関数，位置行列，それらを用いたストリーム処理アルゴリズムの 3 つのアイデアから成り立っている．CrossMatch は探索漏れを発生させることなく正しい解を見つけることができるワンパスアルゴリズムであり，計算時間とメモリ使用量を劇的に低減化する．理論的な分析と，実データ，人工データを用いた実験により，CrossMatch の有効性が確認された．

## 文　　献


[1] L. Gao and X. S. Wang: “Continually Evaluating Similarity-Based Pattern Queries on a Streaming Time Series, ” *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 370-381, Madison, Wisconsin, USA, June 2002.

[2] S. Papadimitriou, A. Brockwell, and C. Faloutsos: “Adaptive, Hands-Off Stream Mining, ” *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003)*, pages 560-571, Berlin, Germany, September 2003.

[3] S. Papadimitriou, J. Sun, and C. Faloutsos: “Streaming Pattern Discovery in Multiple Time-Series, ” *Proceedings of the 31th International Conference on Very Large Data Bases (VLDB 2005)*, pages 697-708, Trondheim, Norway, August-September 2005.

[4] Y. Sakurai, C. Faloutsos, and M. Yamamuro: “Stream Monitoring under the Time Warping Distance, ” *Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE 2007)*, pages 1046-1055, Istanbul, Turkey, April 2007.

[5] Y. Sakurai, S. Papadimitriou, and, C. Faloutsos: “Braid: Stream Mining through Group Lag Correlations,” *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 599-610, Baltimore, Maryland, June 2005.

[6] T. F. Smith and M. S. Waterman: “Identification of Common Molecular Subsequences, ” Journal of Molecular Biology 147, pages 195-197, 1981.

[7] M. Zhou and M. H. Wong: “Efficient Online Subsequence Searching in Data Streams under Dynamic Time Warping Distance,” *Proceedings of IEEE 24th International Conference on Data Engineering (ICDE2008)*, pages 686-695, Cancun, Mexico, April 2008.

[8] Y. Zhu and D. Shasha: “StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time,” *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, pages 358 - 369, Hong Kong, China, August 2002.