Vol.2012-DBS-154 No.9
Vol.2012-IFAT-107 No.9
2012/8/2

データストリームにおける効率的なパターン検出

豊田 真智子[1]　櫻井 保志[1]　石川 佳治[2]

概要：部分シーケンスマッチングは，問合せパターン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であり，古くは時系列データベースに対する問合せとして検討されてきた．一方，問合せシーケンスのない部分シーケンスマッチングとして，2 つのストリーム間に共通するパターンを検出する問題が挙げられる．この問題はトレンド検出やクラスタリング，異常検出などにつながる問題である．本論文では，ダイナミックタイムワーピング距離 (DTW) に基づき，精度を犠牲にすることなく，ストリーム間の共通パターンを高速に検出する CrossMatch を示す．また，センサネットワークなどのリソースが制限された環境を想定し，近似的なアプローチを利用することにより，より効果的に動作する手法を提案する．実データを用いた実験により，少ない検出エラーで，大幅に性能を向上でき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，近似，ダイナミックタイムワーピング距離

An efficient method for pattern discovery in data streams

Machiko Toyoda[1]　Yasushj Sakurai[1]　Yoshiharu Ishikawa[2]

Abstract: Subsequence matching is the problem of finding subsequences similar to a query sequence in data streams. There has been significant research effort as the problem of queries to time series database. Another challenging issue in relation to subsequence matching is how we identify common local patterns when both sequences are evolbing. This problem arises in trend detection, clustering, and outlier detection. In this paper, we present a one-pass algorithm, CrossMatch, that does not sacrifice accuracy and detect common local patterns between data streams based on dynamic time warping (DTW). Moreover, we propose effective algorithms using approximations for highly likely environments that the resources are limited. Our experimental evaluation with real datasets shows that CrossMatch can discover common local patterns within constant time (per update) and space.

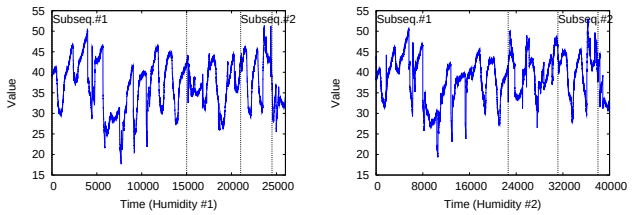
Keywords: Data streams, Subsequence matching, Approximation, Dynamic Time Warping

1. はじめに

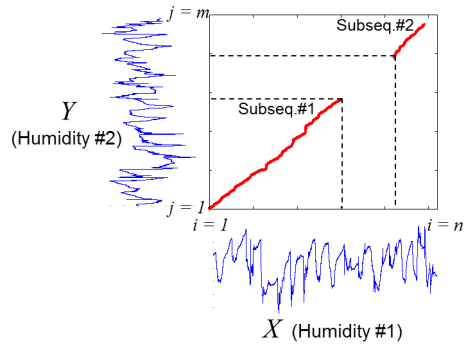
オンラインで連続的に到着するデータはデータストリームと呼ばれ，金融やセンサネットワーク，移動体追跡など様々な分野で発生するデータである．無限長のデータを高速にメモリ上で処理するためのマイニング技術が重要とされ，ストリームマイニングとして研究が進められている．データストリーム処理が必要となるアプリケーションにおいてしばしば求められるのは，連続的にデータストリームをモニタリングする技術であり，その重要な要素技術の 1 つが部分シーケンスマッチングである．部分シーケンスマッチングに関する先行研究の多くは，問合せシーケンスに類似する部分シーケンスをデータストリームから検出する問題に焦点を当てている [12][17]．この問題は，あらかじめ検出したいパターンが決まっている場合に有効である．一方，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．すなわち，問合せシーケンスなしに，データストリーム間に共通するパターンを自動的に検出したい．この問題は，頻出パターン検出やルール抽出，クラスタリング，異常検出などに適用される重要な問題である．図 1 を用いて，本論文で解決したい問題を説明する．

図 1(a) は，2 つの異なる湿度センサから取得された時系列データである．2 つのシーケンスは全体的に似ているが，一部に異なる変化を含んでいる．本論文の目的は，シーケンス間に存在する類似部分シーケンスを検出することである．すなわち，縦の線で示した Subseq. #1 と #2 が類似する部分シーケンスペアとして検出される．図 1(b) は，検出

[1] 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
NTT Communication Science Laboratories
[2] 名古屋大学情報基盤センター
Information Technology Center, Nagoya University



(a) データシーケンス



(b) 検出された共通パターン

図 1 類似部分シーケンスペアの検出例

された 2 つのシーケンスペアの適合を詳細化したものであり，どの要素と要素が適合して共通パターンとなったかが示されている．もし 2 つ類似部分シーケンスペアが完全に一致した場合，直線の対角線として図示される．この例では，2 つの部分シーケンスが時間軸方向に伸縮されて適合したことがわかる．このように，シーケンス間に存在する共通パターンの開始点と終了点，その類似度を，ストリーム処理で高速に検出したい．

本論文では，データストリーム間の共通パターンを検出する CrossMatch について述べる [15][16]．CrossMatch は，距離尺度としてダイナミックタイムワーピング距離 (DTW)[1][14] を利用し，精度を犠牲にすることなく高速に動作するワンパスアルゴリズムである．また，リソースが制限された環境を想定し，CrossMatch の更なる効率化を目指すサンプリングアプローチを提案する．サンプリングアプローチは，シーケンスをサンプリングし，サンプリングされたシーケンスを用いて近似的に共通パターンを検出する．実データを用いた実験により，提案するアプローチが，少ない検出エラーにもかかわらず，大幅に性能を改善することを示す．

本論文は，以下のように構成される．まず 2 章で，データストリーム間のパターン検出についての問題を定義し，この問題を解決する CrossMatch を示す．3 章で CrossMatch の効率化を目的とした 2 つのサンプリングアプローチを導入し，4 章において，実験によりサンプリングアプローチの効果を検証する．5 章で関連研究を示し，最後に 6 章でまとめを述べる．

2. データストリーム間のパターン検出

2.1 ダイナミックタイムワーピング距離 (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 [1][14]．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．距離値を計算するための行列はタイムワーピング行列と呼ばれ，タイムワーピング行列において，DTW 距離を計算するために対応付けられた要素の並びが最適なワーピングパスとなる．

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．

$$D(X,Y) = d(n,m)$$

$$d(i,j) = ||x_i - y_j|| + min \begin{cases} d(i,j-1) \\ d(i-1,j) \\ d(i-1,j-1) \end{cases} \quad (1)$$

$$d(0,0) = 0, \quad d(i,0) = d(0,j) = \infty$$

$$(i = 1, ..., n;\ j = 1, ..., m)$$

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．

一方のシーケンスの小さな範囲と，もう一方のシーケンスの大きな範囲が対応することによるマッチングの精度低下を避けるため，ワーピング制約が導入される．これは，不要なワーピングパスを取り除き，ワーピングの範囲を制限するものであり，Sakoe-Chiba band[11] が代表的なワーピング制約の 1 つである．Sakoe-Chiba band におけるワーピングの範囲を w とすると，式 (1) は $|i - j| \leq w$ という制限の基で計算される．

タイムワーピング行列は nm 個のセルから構成されるため，DTW の計算時間は $O(nm)$ となる．メモリ使用量については，タイムワーピング行列の 2 列 (現在の列と直前の列) だけ必要となるため，各時刻あたり $O(n)$ となる．ワーピング制約を導入した場合，タイムワーピング行列のセルが nw または mw 個に制限されるため，計算時間は $O(nw)$ または $O(mw)$ に削減される．また，この時のメモリ使用量は $2w$ 個のセルのみ保持するため，$O(w)$ となる．

2.2 問題設定

データーストリーム 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．

定義 1 (Cross-similarity) 2 つのシーケンス X と Y，距離の閾値 ε，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 $Y[j_s : j_e]$ は cross-similarity の性質を持つ．

$$D(X[i_s : i_e], Y[j_s : j_e]) \leq \varepsilon(L(l_x, l_y) - l_{min}) \quad (2)$$

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，$L(l_x,l_y)=max(l_x,l_y)$ や $L(l_x,l_y)=min(l_x,l_y)$ を用いてもかまわない．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，部分シーケンスペアの長さ L が l_{min} 以上となる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，距離値が極小値をとる部分シーケンスペアに重複 (overlap) する多くの部分シーケンスペアも適合してしまう．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り，次のように定義される．

定義 2 (Overlap) X と Y の部分シーケンスペアのワーピングパス 2 つが与えられた時，重複とはそれらのワーピングパスの少なくとも 1 つの要素が共有されることである．

重複する部分シーケンスペアは冗長な情報であり，不必要な結果についても報告することでアルゴリズムの処理速度が低下する可能性もある．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 ε，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) 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．

通常，データストリームの最新の要素は過去に到着した要素より重要であるとされている [3]．そこで，DTW にワーピング制約である Sakoe-Chiba band[11] を導入し，新たに到着した要素に注目する．すなわち，ワーピング制約によりタイムワーピング行列の中のセルを制限することで，新しい要素に存在する cross-similarity を検出する．

2.3 CrossMatch

CrossMatch[15][16] は DTW 距離に基づいて効率的に cross-similarity を検出する手法であり，スコアリング関数，位置行列，ストリーム処理アルゴリズムにより構成される．

スコアリング関数は，2 つのシーケンス間の類似度を求める関数であり，DTW と同様に動的計画法に基づいて類似度の計算を行うが，結果をスコアとして出力する点が異なる．スコアとは類似度を計算する行列（スコア行列と呼ぶ）の各要素の累積値として表され，DTW では距離値が小さいほど類似度が高いのに対し，スコアリング関数ではスコアが大きいほど類似度が高いという逆の性質を持っている．スコアリング関数は，以下のように定義される．

6	13			21	0	0	0
5	9		16	22	25	20	0
4	2	11	0	0	27	49	0
3	4	13	0	0	36	42	0
2	9	0	11	26	24	0	
1	11	0	13	19	1		
j	y_j / x_i	5	12	10	6	3	18
	i	1	2	3	4	5	6

(a) スコア行列

6	13			(1,3)	(4,6)	(5,6)	(6,6)
5	9		(1,3)	(1,3)	(2,1)	(2,1)	(6,5)
4	2	(1,3)	(2,4)	(3,4)	(2,1)	(2,1)	(6,4)
3	4	(1,3)	(2,3)	(3,3)	(2,1)	(2,1)	(6,3)
2	9	(1,2)	(2,1)	(2,1)	(2,1)	(5,2)	
1	11	(1,1)	(2,1)	(2,1)	(2,1)		
j	y_j / x_i	5	12	10	6	3	18
	i	1	2	3	4	5	6

(b) 位置行列

図 2 CrossMatch による cross-similarity の検出

定義 3 (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 $V(X[i_s:i_e],Y[j_s:j_e])$ は次のように計算される．

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

$$v(i,j)=\max\begin{cases}0\\ \varepsilon b_v-||x_i-y_j||+v(i,j-1)\\ \varepsilon b_h-||x_i-y_j||+v(i-1,j)\\ \varepsilon b_d-||x_i-y_j||+v(i-1,j-1)\end{cases} \quad (3)$$

$$v(0,0)=v(i,0)=v(0,j)=0$$
$$(i=1,...,n;\ j=1,...,m;$$
$$n-w\le i\le n;\ m-w\le j\le m;).$$

ここで，b_v,b_h,b_d は L によって決定される重みであり，$L(l_x,l_y)=(l_x+l_y)/2$ では，$b_v=b_h=1/2$，$b_d=1$ となる．

スコアは距離の閾値 ε と各要素の距離値の差を累積することにより決定されるため，部分シーケンスペアが条件を満たさない場合には，スコアは負の値を示す．また，負の値となった要素のスコアをゼロでリセットすることで，その要素から新たにスコア計算を開始する．この性質を利用して，不適合な部分シーケンスペアを枝刈りし，適合する可能性の高い部分シーケンスペアのみを 1 つの行列で効率的に計算する．一方，スコアリング関数により出力されるスコア値は，DTW 距離に変換可能であることが保証されており，部分シーケンスペアの DTW 距離は，次の式により計算される．

$$D(X[i_s:i_e],Y[j_s:j_e])=\varepsilon L(l_x,l_y)-V(X[i_s:i_e],Y[j_s:j_e]) \quad (4)$$

また，スコアを用いた場合，問題 1 の 2 つの条件は次の条件と等価となる (詳細な証明は文献 [16] を参照)．

(1) $V(X[i_s:i_e],Y[j_s:j_e])\ge\varepsilon l_{min}$
(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．

図 2 は $X=(5,12,6,10,3,18)$, $Y=(11,9,4,2,9,13)$ のシーケンスに対して，$\varepsilon=14$, $l_{min}=2$, $w=3$ を設定し

た場合の類似部分シーケンスペア検出例を示したものである．図 2(a) はスコア行列の例である．濃く色付けしたセルが最適な部分シーケンスペアであり，スコアが 49，終了点が $(5,4)$ であることが確認される．一方，薄く色付けしたセルが適合する部分シーケンスペアを表す．0 を示すセルは，条件を満たさない部分シーケンスペアを特定している．

スコアリング関数は，類似する部分シーケンスペアの終了点，スコアを保持するものであり，開始点を特定するためにはワーピングパスを遡らなければならない．位置行列はこの問題を解決するためのものであり，行列のセルの中に，スコア行列の同じセルに保持されている部分シーケンスペアの開始点の情報を保持する．

定義 4 (位置行列) スコア $v(i,j)$ に対応する位置行列の開始点 $s(i,j)$ は以下のように求められる．

$$s(i,j)=\begin{cases} s(i,j-1) & (v(i,j-1)\neq 0 \ \wedge \ v(i,j) \\ & =\varepsilon b_v-||x_i-y_j||+v(i,j-1)) \\ s(i-1,j) & (v(i-1,j)\neq 0 \ \wedge \ v(i,j) \\ & =\varepsilon b_h-||x_i-y_j||+v(i-1,j)) \\ s(i-1,j-1) & (v(i-1,j-1)\neq 0 \ \wedge \ v(i,j) \\ & =\varepsilon b_d-||x_i-y_j||+v(i-1,j-1)) \\ (i,j) & (otherwise). \end{cases} \quad (5)$$

図 2(b) は，図 2(a) のスコア行列に対応する位置行列を示している．位置行列より，最適な部分シーケンスペアの開始点は $(2,1)$ であることが確認される．このように，2 つの行列を組合せることで，類似する部分シーケンスペアがストリーム処理において特定可能となる．

CrossMatch の最後のアイディアは，ストリーム処理アルゴリズムである．最新のデータが到着する度に，スコアと開始点を式 (3) と (5) を用いてインクリメンタルに計算する．そして定義 1 を満たす部分シーケンスペアを検出すると，そのスコア，開始点，終了点を候補集合の配列に格納する．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 $v(n,j)-\varepsilon l_{min}$ が最大値となるペアのみが格納される．これらのペアは，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が確認されたのちに報告される．

純粋に DTW を用いて cross-similarity の検出を行った場合，計算コストやメモリ使用量は $O(nw^2+mw^2)$ である (これをナイーブな手法と呼ぶ)．一方，CrossMatch を用いることで，$O(w)$ にまで削減される．

3. 近似手法

CrossMatch は，高速かつ一定コストで類似する部分シーケンスペアを特定する．次の課題は，ユーザがより限られたリソースで効率的な検出を必要とした場合にどうすべきかである．すなわち，cross-similarity の検出に関して高い精度を要求するが，理論的な保証までは必要としない場合である．この問題を解決するために，CrossMatch に近似のアプローチを導入する．具体的には，シーケンスに対するデータ削減を行う．DTW の最適なアライメントは時間軸に対する要素の適合により求められる．類似部分シーケンスペアの検出における近似の誤差を少なくするためには，時間軸方向に関するデータ削減が望ましい．CrossMatch への近似導入を検討するにあたり，次の定理に注目する．

定理 1 (サンプリング定理) 連続信号の最大周波数成分が f_{high} である時，$2f_{high}$ より高い周波数 f_{Nq} でサンプリングすることによりその信号を完全に復元することができる．

証明 1 文献 [6] 参照.

サンプリング定理において，最小サンプリング周波数 f_{Nq} はナイキスト周波数と呼ばれている．我々はこの定理を利用し，シーケンスのデータ削減を行う．すなわち，cross-similarity の検出には，オリジナルのシーケンスではなく，サンプリング定理に基づいてサンプリングされた粗いシーケンスを使用する．本章では，サンプリングを用いた 2 つの近似アプローチを示す．オリジナルのシーケンスは $T=1/f_{Nq}$ 毎にサンプルされるため，検出に使用する行列の大幅な削減が可能となる．

3.1 サンプリングアプローチ

サンプリングされたシーケンス間のスコアをどのように計算するのか．直感的なアイディアは，スコアの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を選択する場合，オリジナルのシーケンスから削除された要素によって与えられる距離値を補完することである．サンプリングされたシーケンス間のスコアを計算するとき，現在のセルと隣接するセルとの間には $T-1$ 個の隠れたセルが存在する．この隠れたセルにおける距離値を，現在のセルの距離値を用いて近似する．この近似に基づく類似部分シーケンスペアの検出を，サンプリングアプローチと呼ぶ．

長さ n のシーケンス X と長さ m のシーケンス Y のサンプリング周期を T_x と T_y，X と Y のサンプリングされたシーケンスを $\mathcal{X}=(\mathrm{x}_1,...,\mathrm{x}_i,...,\mathrm{x}_{\lfloor n/T_x \rfloor})$ と $\mathcal{Y}=(\mathrm{y}_1,...,\mathrm{y}_j,...,\mathrm{y}_{\lfloor m/T_y \rfloor})$ とする時，$\mathcal{X}$ と $\mathcal{Y}$ の部分シーケンスのスコアは次のように計算される．

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

$$v(i,j)=max\begin{cases} 0 \\ \varepsilon b_v-T_y\cdot||\mathrm{x}_i-\mathrm{y}_j||+v(i,j-1) \\ \varepsilon b_h-T_x\cdot||\mathrm{x}_i-\mathrm{y}_j||+v(i-1,j) \\ \varepsilon b_d-max(T_x,T_y)\cdot||\mathrm{x}_i-\mathrm{y}_j||+v(i-1,j-1) \end{cases}$$

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

$$(i=1,...,\lfloor n/T_x \rfloor;\ j=1,...,\lfloor m/T_y \rfloor;$$
$$\lfloor (n-w)/T_x \rfloor \leq i \leq \lfloor n/Tx \rfloor;\ \lfloor (m-w)/T_y \rfloor \leq j \leq \lfloor m/Ty \rfloor;) \quad (6)$$

検出エラーを最小化するため，各方向のスコアを選択した場合に T_x または T_y 個の距離値を補完する．サンプリングアプローチのために，b_v，b_h，b_d の重みも更新される．$L(l_x,l_y)=(l_x+l_y)/2$ において，垂直方向または水平方向のセルのスコアが選択された場合，L の値は $T_x/2$ または $T_y/2$ 増加する．同様に，対角方向のセルのスコアが選択された場合，L の値は $(T_x+T_y)/2$ 増加する．そのため，$b_v=T_y/2$，$b_h=T_x/2$，$b_d=(T_x+T_y)/2$ となる．

サンプリングアプローチにおける位置行列は，オリジナ

ルの式 (5) とほぼ同様である．スコアの計算において隣接する 3 つのセルのいずれかが選択された場合，そのセルの開始点を引継ぎ，それ以外の場合には，サンプリング周期 T_x，T_y に基づく開始点が設定される．具体的には，開始点 $s(i,j)$ は以下のように計算される．

$$s(i,j)=\begin{cases}(i\cdot T_x, j\cdot T_y) & (v(i,j)\leq 0)\\ s(i,j-1) & (v(i,j-1)\neq 0 \wedge v(i,j)\\ & \quad =\varepsilon b_v - T_y||\mathrm{x}_i-\mathrm{y}_j||+v(i,j-1)\\ s(i-1,j) & (v(i-1,j)\neq 0 \wedge v(i,j)\\ & \quad =\varepsilon b_h - T_x||\mathrm{x}_i-\mathrm{y}_j||+v(i-1,j)\\ s(i-1,j-1) & (v(i-1,j-1)\neq 0 \wedge v(i,j)=\varepsilon b_d\\ & \quad -max(T_x,T_y)||\mathrm{x}_i-\mathrm{y}_j||+v(i-1,j-1)\\ ((i-1)\cdot T_x+1,(j-1)\cdot T_y+1) & (otherwise).\end{cases} \tag{7}$$

アルゴリズム 1 は，サンプリングアプローチにおける CrossMatch の動作を示している．新たに要素 x_n が到着した時，n がサンプリング周期 T_x に合致するかどうかのチェックが行われる．もしサンプリング周期に一致しなければ，その要素の計算をスキップし，サンプリング周期に一致した場合，CrossMatch はその要素を $\mathrm{x}_{\lfloor n/T_x \rfloor}$ として，スコア $v(n/T_x,j)$ と開始点 $s(n/T_x,j)$ を計算する．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すると，そのスコア $C'_v := v(n/T_x,j)$，開始点 $C'_s := s(n/T_x,j)$，終了点 $C'_e := (n, j*T_y)$ を候補集合の配列 $\mathcal{S}$ に格納する．重複する部分シーケンスペアが存在する場合，$v(n/T_x,j)-\varepsilon l_{min}$ が最大値となるペアのみが配列 $\mathcal{S}$ に格納される．部分シーケンスペアの重複は，配列 $\mathcal{S}$ に含まれるすべての開始点 (この集合を S_{c_s} とする) と現在の開始点 C'_s が一致するかどうかによって判断される．

候補配列には，開始点が異なる複数の候補部分シーケンスペアの情報 C が保持されている (すなわち，スコア C_v，開始点 C_s，終了点 C_e)．CrossMatch は以下の 2 つの条件を満たす時，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．

$$(\forall_i, s(i,\lfloor m/T_y \rfloor) \neq C_s) \neq C_s) \ \wedge \ (\forall_j, s(n/T_x,j) \neq C_s)$$

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．最終的な部分シーケンスペアの類似度は式 (4) により DTW 距離 d_{min} として報告される．

スコア行列，位置行列では，$\mathcal{X}$ と $\mathcal{Y}$ それぞれについて 2 列 (現在の列と直前の列) のみ保持する．このアルゴリズムでは，時刻 n で $\mathcal{X}$ の要素 $\mathrm{x}_{\lfloor n/T_x \rfloor}$ を受け取った場合の処理に焦点を当てているが，時刻 m で受け取った $\mathcal{Y}$ の要素 $\mathrm{y}_{\lfloor m/T_y \rfloor}$ の処理についても，同様に計算することができる．

補題 1 サンプリングアプローチは，$O(w/T)$ のメモリ使用量と $O(w/T)$ の計算時間を要する．

証明 2 $T = min(T_x, T_y)$ とする時，サンプリングされたシーケンスは，オリジナルのシーケンスを $1/T$ に圧縮したものである．時刻 i において x_i を受け取った場合には $O(w/T_y)$ 個の値を，時刻 j において y_j を受け取った場合には $O(w/T_x)$ 個の値を更新する．そのため，$O(w/T)$ の計算時間と $O(w/T)$ のメモリ使用量を必要とする． □

Algorithm 1 CrossMatch (sampling)

Input: 時刻 n における新たな要素 x_n
Output: 近似の最適な部分シーケンスペアとその DTW 距離

```
1: if n mod T_x = 0 then
2:   // 最適な部分シーケンスペアの検出
3:   for j := ⌊(m − w)/T_y⌋ to ⌊m/T_y⌋ do
4:     C'_v := v(n/T_x, j); // 式 (6) によるスコア
5:     C'_s := s(n/T_x, j); // 式 (7) による開始点
6:     C'_e := (n, j ∗ T_y); // 終了点
7:     if C'_v ≥ εl_min then
8:       // 新たな候補ペアとして配列に追加.
9:       if C'_s ∉ S_{c_s} then
10:        add C'_v, C'_s, and C'_e to S;
11:      else
12:        for each candidate C ∈ S do
13:          // 最大スコアを更新
14:          if C'_s = C_s ∧ C'_v ≥ C_v then
15:            C_v := C'_v;
16:            C_e := C'_e;
17:          end if
18:        end for
19:      end if
20:    end if
21:  end for
22:  // 最適な部分シーケンスペアの報告
23:  for each candidate C ∈ S do
24:    if (∀_i, s(i, ⌊m/T_y⌋) ≠ C_s) ∧ (∀_j, s(n/T_x, j) ≠ C_s)
       then
25:      d_min = εL(l_x, l_y) − C_v;
26:      Report d_min, C_s and C_e;
27:      Remove C from S;
28:    end if
29:  end for
30: end if
```

3.2 動的サンプリングアプローチ

前節では，固定のサンプリング周期が与えられた場合について述べた．データストリームは，時間と共に変化するシーケンスであるため，最新の要素を反映したシーケンスに含まれる高周波と低周波の割合も，時間と共に変わる可能性がある．これは，ナイキスト周波数が動的に変化することを意味する．そこで，この変化を CrossMatch に組込み，新たな要素が到着した時にナイキスト周波数を計算し，サンプリング周期を決定する．この近似に基づく類似シーケンスペアの検出を，動的サンプリングアプローチと呼ぶ．

X と Y のサンプリングされたシーケンスを $\mathcal{X} = (\mathrm{x}_1,...,\mathrm{x}_i,...,\mathrm{x}_{n'})$ と $\mathcal{Y} = (\mathrm{y}_1,...,\mathrm{y}_j,...,\mathrm{y}_{m'})$，$\mathcal{X}$ と $\mathcal{Y}$ の各要素までのシーケンスを用いて決定されたサンプリング周期を $\mathcal{T}_x = (t_{\mathrm{x}_1},...,t_{\mathrm{x}_i},...,t_{\mathrm{x}_{n'}})$ と $\mathcal{T}_y = (t_{\mathrm{y}_1},...,t_{\mathrm{y}_j},...,t_{\mathrm{y}_{m'}})$ とする．動的サンプリングアプローチにおけるスコア $V(\mathcal{X}[i_s:i_e],\mathcal{Y}[j_s:j_e])$ は，次のように計算される[*1]．

[*1] 動的サンプリングアプローチにおけるその他の設定は以下のとおりである．$v(0,0)=v(i,0)=v(0,j)=0$. $i=1,...,n'$, $j=1,...,m'$.

$$V(\mathcal{X}[i_s:i_e],\mathcal{Y}[j_s:j_e]) = v(i_e,j_e)$$
$$v(i,j) = max\begin{cases} 0 \\ \varepsilon B_v(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) - t_{\mathrm{y}_j}\cdot||\mathrm{x}_i - \mathrm{y}_j|| + v(i,j-1) \\ \varepsilon B_h(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) - t_{\mathrm{x}_i}\cdot||\mathrm{x}_i - \mathrm{y}_j|| + v(i-1,j) \\ \varepsilon B_d(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) - max(t_{\mathrm{x}_i},t_{\mathrm{y}_j})\cdot||\mathrm{x}_i - \mathrm{y}_j|| \\ \qquad\qquad + v(i-1,j-1) \end{cases} \tag{8}$$

サンプリングされたシーケンス間のスコア計算における隠れたセルの数は，各時刻のサンプリング周期に対応している．例えば，新たな要素 $\mathrm{x}_{n'}$ に対して，水平方向のセルからスコアを引継いだ場合，$t_{\mathrm{x}_{n'}}-1$ 個の隠れセルが存在する．そのため，各時刻において，変化するサンプリング周期を反映してスコアを計算する．また，部分シーケンスの長さ L が各時刻のサンプリング周期により決定されるため，各方向の重み B_v, B_h, B_d もこれに対応させる．すなわち，$L = (l_x + l_y)/2$ において，セル (i,j) を計算する場合，$B_v(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) = t_{\mathrm{y}_j}/2$，$B_h(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) = t_{\mathrm{x}_i}/2$，$B_d(t_{\mathrm{x}_i},t_{\mathrm{y}_j}) = (t_{\mathrm{x}_i}+t_{\mathrm{y}_j})/2$ となる．

このように，動的サンプリングアプローチは，計算の度に変化したサンプリング周期を用いるため，最新の要素を反映した近似が可能となる[*2]．

なお，インクリメンタルに周波数を計算するアルゴリズムは文献 [5][8] で提案されている．CrossMatch はこれらの手法を利用し効率的に周波数を計算することが可能である．

4. 評価実験

CrossMatch の 2 つのサンプリングアプローチの有効性を検証するため，評価実験を行った．実験で使用したデータセットは，以下の通りである．

- *Automobile traffic*: 自動車の交通量を測定した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 (図 3(a))．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．
- *Web*: 各 Web サイトにおけるアクセス数を 10 秒毎に記録したデータセットである (図 3(b))．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 が検出されたメールサイトとブログサイトのアクセスパターンを示す．
- *Sunspots*: 1 日毎の太陽の黒点数を記録したものである (図 3(c))．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．
- *Temperature*: 温度センサを使って 1 分間隔に約 16 日間取得された時系列データである (図 3(d))．これのデータセットは多くの時刻で測定値が欠けており，各シーケンスの長さは異なっている．

実験には，4GByte のメモリと Intel Core 2 Duo 2.4GHz の CPU を搭載したマシンを使用した．

4.1 パターン検出精度

データストリーム間の類似パターンの検出について，実データを用いたケーススタディを示す．本実験では，検出精度を検証することが目的であるため，事前にサンプリング周期の計算を行った．まず，サンプリングアプローチにおいて，各データセットをサンプリングするためのパワースペクトルを計算する．パワースペクトルは，正規化されたシーケンスから計算され，$\sum \mathcal{F}^2 = 1$，$\mathcal{F}^2(f_0) = 0$ となる．各データセットには，小さなエネルギーを持つ高い周波数が多く含まれており，このようなデータにおけるナイキスト周波数は極めて高い値になる．そこで，小さなエネルギーを持つ周波数成分を無視し，大きなエネルギーを持つ周波数成分に焦点を当てた．これは，音声処理やネットワーク分析などの分野においても一般的に行われている方法である．この実験においては，パワースペクトルの閾値を 5.0e-4 に設定した．これにより，*Traffic # 1* は $0 \le f \le 1478$ の範囲に，*Traffic # 2* は，$0 \le f \le 1438$ の範囲にエネルギーが分布していることがわかった．このことから，これらのデータセットのナイキスト周波数は $f_{Nq_x} = 2596/n$，$f_{Nq_y} = 2876/m$ と計算されるため，サンプリング周期はそれぞれ $T_x = 5$, $T_y = 6$ となる[*3]．

次に，動的サンプリングアプローチのサンプリング周期を計算した．各シーケンスにおいて，シーケンス長 100 までは，サンプリングアプローチで用いた固定のサンプリング周期に基づいてシーケンスをサンプリングし，その後はサンプリング周期に対応する要素でシーケンスを区切り，次のサンプリング周期を計算するという操作を繰り返した．

図 3 は，各データセットと 3 つのアプローチに関する cross-similarity の検出結果を示している．2 つのサンプリングアプローチは近似的な計算を行うにもかかわらず，オリジナルアプローチと同様に cross-similarity の検出に成功していることが確認される．特徴的なのは，*Web* における 2 つのサンプリングアプローチの実験結果であり，オリジナルアプローチでは検出されなかった類似シーケンスペアの検出に成功している．これは，サンプリングにより高周波成分の影響が減り，正しく DTW のアライメントに反映されたためであると考えられる．

次に，2 つのサンプリングアプローチの検出エラーについて検証する．検出エラーは，DER (Diarization Error Rate)[4] により測定した．DER は，録音された音声データから，話者が会話している部分を特定するという音声認識においてよく利用されている尺度であり，話者が会話した正しい時間を合計し，その時間に対して間違って認識した会話時間の割合を調べることにより測定される．本実験においては，オリジナルアプローチによって検出された正

$\lfloor(\sum_{i=1}^{n'} t_{\mathrm{x}_i} - w)/t_{\mathrm{x}_{n'}}\rfloor \le i \le n'$, $\lfloor(\sum_{j=1}^{m'} t_{\mathrm{y}_j} - w)/t_{\mathrm{y}_{m'}}\rfloor \le j \le m'$.

[*2] 位置行列とアルゴリズムについては省略するが，サンプリングアプローチにおける固定のサンプリング周期を，動的なサンプリング周期に書き換えることにより，同様に処理される．

[*3] その他のデータセットの周波数成分 f とサンプリング周期 T は以下の通りである．
Mail: $0 \le f \le 63$, $T_x = 254$. *Blog*: $0 \le f \le 80$, $T_y = 200$.
Sunspots #1: $0 \le f \le 1431$, $T_x = 6$. *Sunspots #2*: $0 \le f \le 1497$, $T_y = 6$.
Temperature #1: $0 \le f \le 201$, $T_x = 70$. *Temperature #2*: $0 \le f \le 137$, $T_y = 88$.

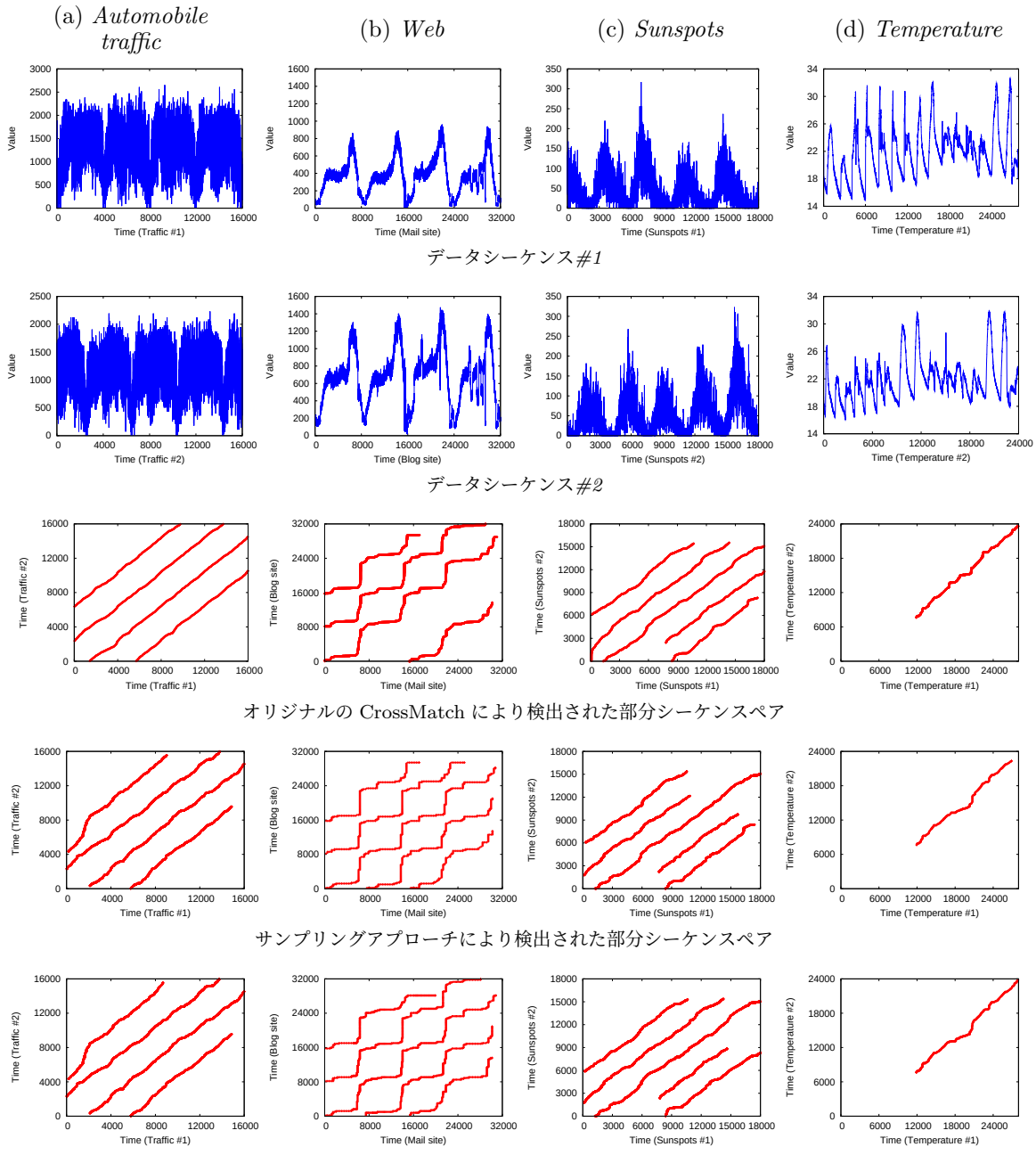


図 3 cross-similarity の発見

しい部分シーケンス長に対する，各サンプリングアプローチによって検出された部分シーケンスペアの不一致区間の長さによって計算され，以下の式で定義される．

$$DER_x = \frac{|i_s - i'_s| + |i_e - i'_e|}{i_e - i_s + 1}, DER_y = \frac{|j_s - j'_s| + |j_e - j'_e|}{j_e - j_s + 1}$$

(i_s, j_s) はオリジナルの CrossMatch における開始点，(i'_s, j'_s) は 2 つのサンプリングアプローチにおける開始点を表し，終了点も同様である．

表 1 は，すべての検出された部分シーケンスペア毎に DER を計算し，その値をデータセット毎に平均値として求めたものである．データセット毎に若干違いはあるものの，どちらのアプローチも少ないエラーで最適な部分シーケンスペアの位置を特定していることが確認される．2 つのアプローチの違いは，*Temperature* に表れている．主に高周波で構成される他のデータセットと異なり，*Temperature* は高周波と低周波の両方を含むデータセットである．そのため，周波数の変化に対応してスコアを近似する動的サンプリングアプローチの方が，より精度よく cross-similarity を特定していると言える．

4.2 性能

CrossMatch の性能に関して，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．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．SPRING は文献 [12] 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り，1 データ処理当り $O(nw + mw)$ のメモリ使用量と

表 1 2 つのサンプリングアプローチにおける検出エラー

	サンプリングアプローチ		動的サンプリングアプローチ	
	DER_x	DER_y	DER_x	DER_y
Automobile traffic	0.066	0.094	0.074	0.095
Web	0.039	0.042	0.039	0.036
Sunspots	0.105	0.121	0.093	0.106
Temperature	0.068	0.084	0.002	0.015

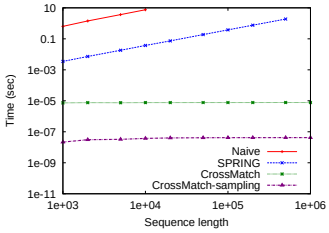


図 4 シーケンス長に対する計算時間

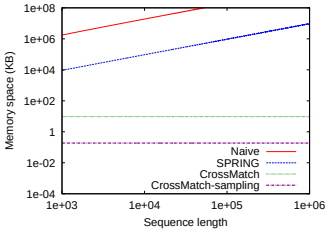


図 5 シーケンス長に対するメモリ使用量

$O(nw + mw)$ の時間が必要となる．実験には *Temperature* データセットを用いた．

図 4 は計算時間に関する比較結果である．結果は，各シーケンス長において，1 データ処理当りに行列を更新する時間のみを測定し，その平均を計算時間として示している．図 5 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．本実験では，理論的な分析と一致するかどうかを検証するため，サンプリング周期の計算については考慮していない．その結果，どちらのアプローチもほぼ同等の結果であったため，図にはサンプリングアプローチの結果を示す．実験結果から，CrossMatch が，ナイーブな手法，SPRING と比べて非常に高い計算時間，メモリ使用量を示すにも関わらず，サンプリングにより，さらにその性能を向上し，メモリの低減化を達成していることが確認される．この効果の度合は，サンプリング周期が大きいほど大きくなるため，使用するアプリケーションによっては，より効果的な手法と成り得る可能性がある．

5. 関連研究

データストリームにおけるシーケンスマッチングの研究としては，時間周期のずれと振幅のスケールの違いを考慮する SpADe を用いた研究 [2] や，DTW を用いた研究 [12][17] が挙げられるが，いずれも固定長の問合せシーケンスが与えられた時の，類似部分シーケンスを検出する問題を扱っている．また，ストリームマイニングの研究としては，相関 [13][18] やトレンド検出 [10]，予測 [9]，モチーフ発見 [7] などが挙げられるが，本論文で示した問題を対象とするものではない．

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 を示した．また，CrossMatch をより効率的にするため，サンプリングによる近似を導入した 2 つのサンプリングアプローチを提案した．どちらのアプローチも，少ない検出エラーにもかかわらず，計算時間とメモリ使用量を劇的に低減化することが確認された．

参考文献

[1] Berndt, D. J. and Clifford, J.: Finding Patterns in Time Series: A Dynamic Programming Approach, *Advances in Knowledge Discovery and Data Mining*, pp. 229–248 (1996).
[2] Chen, Y., Nascimento, M. A., Ooi, B. C. and Tung, A. K. H.: SpADe: On Shape-based Pattern Detection in Streaming Time Series, *ICDE*, pp. 786–795 (2007).
[3] Cormode, G., Korn, F. and Tirthapura, S.: Time-decaying aggregates in out-of-order streams, *PODS*, pp. 89–98 (2008).
[4] Fiscus, J. G., Ajot, J. and Garofolo, J. S.: The Rich Transcription 2007 Meeting Recognition Evaluation, *Multimodal Technologies for Perception of Humans, International Evaluation Workshops CLEAR 2007 and RT 2007*, Lecture Notes in Computer Science, Vol. 4625, Springer, pp. 373–389 (2008).
[5] Gilbert, A. C., Guha, S., Indyk, P., Muthukrishnan, S. and Strauss, M.: Near-optimal sparse fourier representations via sampling, *STOC*, pp. 152–161 (2002).
[6] Lathi, B. P.: *Signal Processing and Linear Systems*, Oxford University Press (1998).
[7] Mueen, A. and Keogh, E. J.: Online discovery and maintenance of time series motifs, *KDD*, pp. 1089–1098 (2010).
[8] Ogras, Ü. Y. and Ferhatosmanogl, H.: Online summarization of dynamic time series data, *VLDB J.* (2006).
[9] Papadimitriou, S., Brockwell, A. and Faloutsos, C.: Adaptive, hands-off stream mining, *VLDB*, pp. 560–571 (2003).
[10] Papadimitriou, S. and Yu, P. S.: Optimal multi-scale patterns in time series streams, *SIGMOD Conference*, pp. 647–658 (2006).
[11] Sakoe, H. and Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 26, No. 1, pp. 43–49 (1978).
[12] Sakurai, Y., Faloutsos, C. and Yamamuro, M.: Stream Monitoring under the Time Warping Distance, *ICDE*, pp. 1046–1055 (2007).
[13] Sakurai, Y., Papadimitriou, S. and Faloutsos, C.: BRAID: stream mining through group lag correlations, *SIGMOD Conference*, pp. 599–610 (2005).
[14] Sakurai, Y., Yoshikawa, M. and Faloutsos, C.: FTW: fast similarity search under the time warping distance, *PODS*, pp. 326–337 (2005).
[15] Toyoda, M. and Sakurai, Y.: Discovery of Cross-similarity in Data Streams, *ICDE*, pp. 101–104 (2010).
[16] Toyoda, M., Sakurai, Y. and Ishikawa, Y.: Pattern discovery in data streams under the dynamic time warping, *VLDB J.*
[17] Zhou, M. and Wong, M. H.: Efficient Online Subsequence Searching in Data Streams under Dynamic Time Warping Distance, *ICDE*, pp. 686–695 (2008).
[18] Zhu, Y. and Shasha, D.: StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time, *VLDB*, pp. 358–369 (2002).