時系列データ類似検索方式 DIASTRA の提案と評価

藤野 友也† 今村 誠† 菅野 幹人† 撫中 達司†

†三菱電機株式会社 情報技術総合研究所 〒247-8501 神奈川県鎌倉市大船 5-1-1

E-mail: †{Fujino.Tomoya@cj, Imamura.Makoto@bx, Kanno.Mikihito@bc, Munaka.Tatsuji@ct}.MitsubishiElectric.co.jp

あらまし 時系列データの検索は，指定された値の完全一致により検索目的が達成されることは少なく，連続した値の系列･パターンに対して，類似性を評価して検索を行う必要がある．本論文では，大量の時系列データの局所波形に対する二次近似により特徴を抽出して類似検索を行うことで，比較する時系列の類似判定を高速化し，波形の概形に基づく類似度評価を行う DIASTRA 方式を提案する．また従来の類似評価の基本手法である Dynamic Time Warping (DTW) との比較評価を実施し, DTW を使用した類似検索よりも 1400 倍以上高速であり, 95%の確率で DTW と同様の類似結果を提示する実験結果を得たことを報告する．

キーワード 類似検索，要約，時系列データ

DIASTRA: A Method for the Similarity Traversals of Time-Series Data

Tomoya FUJINO† Makoto IMAMURA† Mikihito KANNO† and Tatsuji MUNAKA‡

† Information Technology R&D Center, Mitsubishi Electric Corporation

5-1-1 Ofuna, Kamakura-shi, Kanagawa, 247-8501 Japan

E-mail: †{Fujino.Tomoya@cj, Imamura.Makoto@bx, Kanno.Mikihito@bc, Munaka.Tatsuji@ct}.MitsubishiElectric.co.jp

1. はじめに

近年，異常の早期発見を目的として，工場やビルなどで動作中の設備機器の状態を，常時観測する取組みが数多くなされている．観測されたそれらのアナログ値の変化を示す時系列データは一般に膨大であるが，後の活用を想定して蓄積されることが多い．時系列データの活用の際には，過去に蓄積された膨大な時系列データの中から，目的のパターンを抽出する検索を行うことが必須である. しかし, 時系列データの検索は，リレーショナルデータベースにおけるレコード検索とは異なり，指定された値の完全一致により検索目的が達成されることは少なく, 連続した値の系列･パターンに対して，ある程度の揺らぎを許容して，完全一致ではなく類似性を評価して検索を行う必要がある．本論文では，大量の時系列データの局所波形に対する二次近似により特徴を抽出して類似検索を行うことで，比較する時系列の類似判定を高速化し，波形の概形に基づく類似度評価を行う DIASTRA 方式を提案する．

2. 従来研究

センサ情報のうち，高周波成分を含まない波形を確認する際の人の判断基準は，時系列方向の伸縮に捕らわれずに概形で判断がされる．

そのため，時系列データの検索は，リレーショナルデータベースにおけるレコード検索とは異なり，指定された値の完全一致により検索目的が達成されることは少ない. 連続した値の系列･パターンに対して，ある程度の揺らぎを許容して，完全一致ではなく類似性を評価して検索を行う必要がある．

時系列類似検索に関して注目されている技術として，時間方向の歪みを吸収して類似判定を行い検索するダイナミックタイムワーピング (Dynamic Time Warping: DTW) 方式 [1] が知られている．しかし，DTW は一般にメモリと処理時間を多く要する動的計画法を用いており，実用上も処理時間を長く要すると共に，外乱などによる局所的かつ微小な変動(ノイズ)に対しても，類似判定の際に考慮され，誤判定につながるという課題があった．このような DTW の課題を改善するために，様々な近似処理が提案されている．処理時間に対して標準的に用いられるアイデアとして，動的計画法において探索に使用する二次元配列の使用範囲を適切に限定することで，探索範囲 (ワーピングパス) を絞り込む方式がある．

Vlachos らは，LCSS (Longest Common SubSequence) [2] において，通常算出時よりも距離が短くなるように，探索対象を限定する MBE (最小境界) を設定する

ことで，探索範囲を削減して高速化することを提案している．また，類似度の指標として，2 つの時系列中でマッチングする部分の割合を採用している.ただし，データにパルス状にノイズが存在する場合には MBE が適正に設定出来ないという課題がある.

中村らは，AMSS (Angular Metrics for Shape Similarity) [3] において，類似評価を多次元時系列データに拡張し，2 系列の各点の組み合わせに対するコサイン類似度を評価し，全体での類似度を算出することで,DTW の短所であるノイズによる類似度の誤判定を抑制している．しかし，変化の激しいデータに対してはノイズが存在しない場合にも誤判定が多いという課題がある.

Salvador らは，FastDTW [4] において，粒度の粗い時間間隔に集約した時系列データを評価し，その結果を元にして，より粒度の細かい時間間隔の時系列データに対してワーピングパスを制限することを，再帰的に反復することで，全体の処理を削減している．この手法は，等間隔で粒度の粗い時間間隔へ要約する場合の区切りの位置や，ワーピングパスの選択方式に，精度が大きく依存する点が課題である.

櫻井らは，FTW (Fast search method for dynamic Time Warping) [5] において,k 近傍距離を基準としたとき，DTW による類似距離が基準より小さい場合には必ず近似算出した類似距離が基準より小さいことを保証することで検索精度を維持し，通常の DTW に比べ処理時間を 222 分の 1 とする性能を実現している.

Morse らは，FTSA (Fast Time Series Evaluation method) [6] において,動的計画法において通常使用するような大きな二次元配列を使用せず，後の計算に必要な最小限の領域のみを用いることで，使用メモリ量をデータサイズの線形に削減し，同時に処理時間を削減している．FTSA の貢献は，主にメモリ使用量の削減である.

これまで紹介した既存技術は，2 つの時系列データに対して，双方の全体としての類似度を評価する，類似距離算出の手法である．一方，長大な時系列データ(データシーケンス) に対して，短い時系列データ (問合せシーケンス) を与え，問合せシーケンスに対して類似距離が極小になるようなデータシーケンスの部分シーケンスを検出する類似検索が考えられる.

櫻井らは SPRING [7] において，一つの行列を更新しながら類似距離を評価することによって，データシーケンスに対する処理時間を短縮し，長さ 256 の問い合わせシーケンスに対し，DTW の 65 万倍の処理性能を示す類似検索を提案している．この方式は，単位時間の入力当たり線形時間である.

上記に紹介した従来技術に関する,DTW との性能評価結果を表 2 に示す.

表 2 既存技術の性能評価 (DTW との比較)

技術名	精度比	時間性能	実測時間比
DTW[1]	---	二乗時間	---
LCSS[2]	誤差率 2-10%	二乗時間	5～7 倍高速
AMSS[3]	20 データセット中 13 件で誤差改善	二乗時間	評価無し
Fast DTW[4]	誤差率 0.8%～19.2%	二乗時間	117 倍高速
FTW[5]	DTW と同等	二乗時間	222 倍高速
FTSA[6]	DTW と同等	二乗時間	7～8 倍高速
SPRING [7]	DTW と同等	単位時間入力当たり線形時間	650000 倍高速

本論文では，時系列データの類似検索方式として，唯一回答を必須とせず，正解候補の複数回答を示すことを許容する用途に対する，新しい類似検索手法 DIASTRA (DIAgnostic Similarity TRAversal method) を提案する．DIASTRA では，二次近似による近似精度の確保，要約要素に基づく時系列データの区間分割，類似距離算出時の個別計算の定数時間化により，近似類似検索の高速化を実現する.

[従来技術と DIASTRA との比較]

本論文の提案手法 DIASTRA のアプローチは，前述の従来技術の課題を考慮し，以下の特徴を持つ.

(A) 近似精度の確保

LCSS のように観測値に最小境界を設定し，長方形の領域で粗く要約するのではなく，区間内を二次近似した二次式の係数とする新たな方式を用いて，近似精度を確保する．この手法は，パルス状のノイズに対して頑強である．また，二次曲線近似を用いることにより，時間方向の波形の伸縮を二次曲線の横軸方向の伸縮に置き換えることができ，類似評価が簡潔となる.

(B)要約のための適応的な区間分割

FastDTW のように等間隔で時間を区切るのではなく，要約の精度を確保できるように不定長で区切る新たな方式を用いて，近似精度を確保する．DIAPSTRA では，区間の区切り幅を，二次近似による誤差が許容範囲内となる区切り幅に設定する．また，AMSS で課題となる変化の激しいデータに対しても，変化の激しい箇所を細かい区間とすることで，精度を確保する.

(C) 類似検索にかかる計算量の抑制

櫻井らの FTW, SPRINT のように，時系列データを元の値系列のまま動的計画法による探索対象とするのではなく，区間を単位として動的計画法による探索対象

とすることで，探索総数を大幅に削減し，高速化を行う．マッチングされた区間同士の距離は，2 つの二次曲線の間の距離として，定数時間で算出可能である．

3. 提案手法

3.1. 概要

我々が提案する DIASTRA 方式では，時系列データを，要約精度が許容範囲内となる不定長の区間に分割し，各区間に対応する要約情報 (以下，要約要素) を基準として,類似検索に用いる距離の近似算出を行う．

DIASTRA 処理の流れは，以下に示す通りである．

(1) 検索対象の時系列データ (データシーケンス) について，区間の設定及び区間毎の二次近似による要約を行い，要約要素の配列を得る．(前処理)
(2) 検索のキーとする時系列データ (問合せシーケンス) について，区間の設定および区間毎の要約を行い，要約要素の配列を得る．
(3) データシーケンスの要約要素配列の部分配列と，問合せシーケンスの要約要素配列とのマッチングを，動的計画法を用いて行い，類似距離を算出する．検索では，問合せシーケンスより大きい幅のウィンドウを，データシーケンス上でスライドさせ，ウィンドウの範囲を部分配列とする．
(4) 類似距離が極小となるインデックスを，類似箇所として出力する．

動作概要を疑似コードで示したものを，付録に記載する．以下の節では，それぞれの工程の中から特に，(1)にあたる二次近似による要約方式と(3)にあたる類似距離算出の方式について述べる．

3.2. 二次近似による要約方式

本節では，データシーケンスを，二次近似が可能な不定長の区間へ分割し要約する方式について述べる．(付録の関数 Summarize(DATA)を参照)

[二次近似による要約手順]

1) 区間の始点をデータシーケンスの始点とする．
2) 区間の終点を段階的に伸張しながら，区間内データの二次近似曲線の係数を，伸張した増分に対する差分演算により算出し，その誤差を評価する．
3) 二次近似誤差が区切り条件を満たす最小のインデックスを区間区切りとし，区間を確定する．
4) 確定区間の二次近似曲線の係数を，始点・終点情報と共に要約要素として保持する．
5)確定区間の終点の次を新たな始点とし，2)に戻る．

上記 2) では，時刻を x 値，データシーケンス値を y 値として二次近似を行う．要約要素としてデータシーケンスのインデックス i に対する時刻インデックスを x_i，時系列データの振幅値を y_i，区間の始点のインデックスを x_S，終点のインデックスを x_G とおいたとき，式(1)の解として二次近似式の係数 a, b, c を得る．要約要素は x_S，x_G，a, b, c である．

$$\text{minimize} \sum_i (y_i - a(x_i - x_S)^2 + b(x_i - x_S) + c)^2 \cdots (1)$$

差分演算により，二次近似による要約手順は全体で線形時間で処理される．

上記 3) の区切り条件は，次の様に設定する．図 1 上の観測値の推移に対して，区間の始点 x_S を基点とし，終点 x_G を順次増加させた場合の，二次近似式における x_G に対応する二次近似誤差率 (二乗誤差/区間幅) は図 1 下のようになる．区切り条件として，二次近似誤差率を終端インデックス毎に保持し，過去の誤差率の平均値との差分が，過去の誤差率の標準偏差を超えた箇所(立ち上がり点)とする．本論文では，近似誤差評価のインデックス幅の下限 MIN_WIDTH を 45 とした．

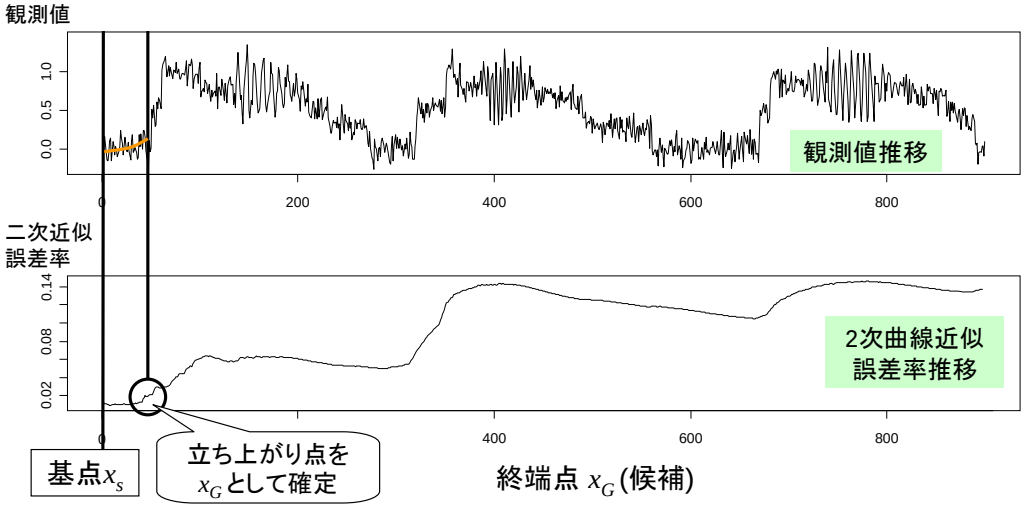

図 1 観測値に対する二次近似誤差率の評価

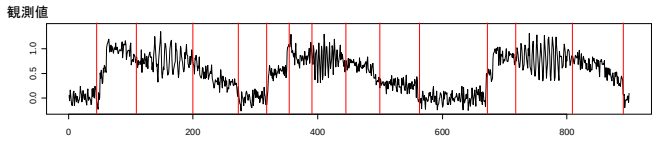

図 2 区間分割結果例

図 2 に，データシーケンスに対する区間分割の結果例を示す．

DIASTRA では，要約要素数が多いほど近似精度は高まるが，処理時間は要約要素数の二乗に比例するオーダーで増加するトレードオフが発生する．そのため，元データに対してどの程度の細かさで要約要素に分割されるように，誤差率立ち上がりの判定基準を設定するかは，問題に応じて調整する運用が求められる．

3.3. 類似距離算出

本節では，要約要素の配列同士からの類似距離の算出方式について述べる．(付録の関数 GetDistance(ONE, ANOTHER)を参照)

距離算出には，最初に要約要素の配列同士でマッチングを行い，マッチングされた要約要素同士の距離を算出し，それらの総和を全体の類似距離とする．

要約要素のマッチングは，個数の異なる要素のマッチングとなるため，動的計画法を用いて行う．マッチングは多対一の対応付けを許し，系列の順番が逆転する対応付けは行わない．

要約要素同士の距離算出では，要約要素に対応する 2 つの二次曲線の定義域を 0 以上 1 以下に正規化し，差の二乗値の積分値の平方根を距離とする．これは定数時間で算出可能である．それぞれの要約要素の定義域正規化後の係数の差を a_Δ, b_Δ, c_Δ とおくと，距離 D は以下の式 (2) で定数時間にて求められる．

$$D = \sqrt{\int_0^1 \left(a_\Delta x^2 + b_\Delta x + c_\Delta\right)^2 dx}$$

$$= \sqrt{\frac{a_\Delta^2}{5} + \frac{a_\Delta b_\Delta}{2} + \frac{b_\Delta^2}{3} + \frac{2a_\Delta c_\Delta}{3} + b_\Delta c_\Delta + c_\Delta^2} \quad \cdots(2)$$

4. 評価

本章での評価は，基本となる DTW と比較した精度及び時間性能の評価を行う．

検索対象のデータとして， UCI KDD Archive [8] の海水温度推移データ tao-all2.dat を使用．s.s.temp. (sea surface temperature) 項目の値で，欠損値を除外した先頭 85000 件を評価用のデータシーケンスとした．評価用の問合せシーケンスは，データシーケンスの部分データに時間方向の変調をかけたものとした．

問合せシーケンスとするためのデータシーケンスの切り出し箇所は，データシーケンスの後半を等間隔に区切った 10 箇所とし，抽出長は 5000 とする．

また評価では，複数回答を許容する用途での使用を前提とし，複数回答の件数に対する精度と，高速化の効果について示す．応用上，状態監視において，類似検索の精度は用途に応じて 95%～99%の範囲で求められることが多いため，評価も上記の範囲の精度を達成するか否かを基本として評価を行う．

4.1. 精度評価

10 種類の問合せシーケンスに対して，変調 (時間方向の伸縮) およびノイズの付与を行い，抽出箇所が類似箇所と判定される精度を評価する．

時間方向に問合せシーケンス系列を伸縮する変調方式は，以下の 4 種類とする．

(1) 変調方式 1：変調を行わない．
(2) 変調方式 2：時間方向に全体を 1.1 倍する．
(3) 変調方式 3：問合せシーケンスを等分し，前半を 0.9 倍，後半を 1.1 倍に時間方向に伸縮する．
(4) 変調方式 4：時間方向に，問合せシーケンスを 10 箇所に等分割し，それぞれ 0.9 倍～1.1 倍の範囲で，正弦波と同じ変動倍率で時間方向に伸縮する．

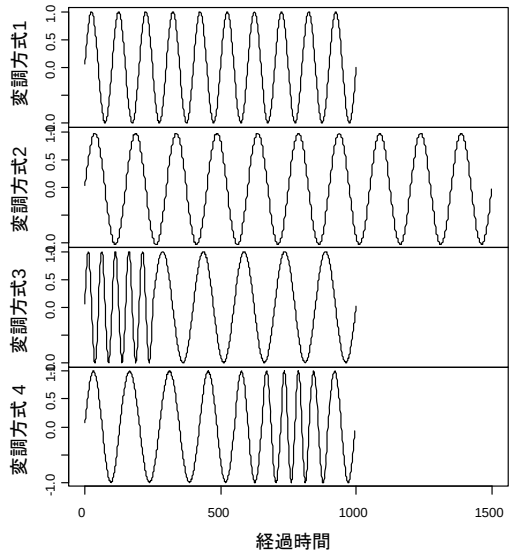

図 3　変調方式

なお，正弦波を各方式で変調した場合のイメージを図 3 に示す．(図 3 では変化率を 5 倍にしている)

一方，各問合せシーケンスに付与するノイズは，問合せシーケンス系列の値が取る範囲幅に対して 0%か

ら 5%未満の 10 段階とし，それぞれの段階の範囲幅比率を最大幅とする一様乱数を，問合せシーケンス系列に加算して評価する．

精度評価は，問合せシーケンス番号，変調方式，ノイズレベルごとの検索数 400 件と，類似検索結果のヒット数の比で評価する．なおここでは，類似判定の特性上，問合せシーケンスの検索位置にずれがあることを考慮し，問合せシーケンスを抽出したインデックスの前後 2500 の範囲に,類似検索結果のインデックスが存在する場合をヒットとした．

ヒットと見なす上位件数を 1 から 15 まで変更した場合のヒット率の推移を表 3 および図 4 に示す．DTW による検索では，いずれのケースでもヒット率は 1.00 であった．DTW で検出した事例の取り漏らしは，特に変調方式 4 において発生しており，二次曲線近似の区間内での時間伸縮率の変化によるものと推定される．

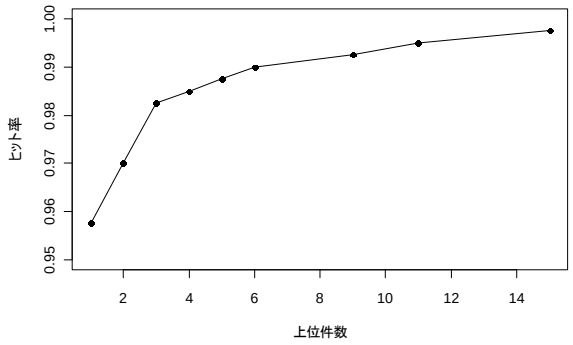

図 4　上位件数に対するヒット率

類似距離最小の箇所のみ採用 (上位件数 1 件) の場合には 95.8%の確率でヒットしており，上位 6 件以上の場合は 99.0%のヒット率が得られている．このことより，実用上必要とされる精度 95%以上を満たす運用が可能であることを確認した．

4.2. 時間性能評価

表 3 に示す実行環境にて,時間性能評価を実施した．

表 3　性能評価環境

CPU	Intel Xeon (3.20GHz×2)
OS	Microsoft Windows Server 2003 SP2
Memory	6GB
HDD	73GB, 10000rpm, Ultra320 ×4

本節では，4.1 節と同じ 10 種類の問い合わせシーケンスに対する類似検索時間を評価する．なお本評価では DTW の検索時の走査ウィンドウを一度に 100 インデックス移動するように間引くよう設定した．

上記条件にて検索処理を行った処理時間を，表 4 に示す．(DIASTRA については，要約処理の時間を含む)

表 4　処理時間評価結果

Key No.	要約要素数	DIASTRA	DTW	処理時間比
1	78	1.44 秒	2022 秒	1406
2	72	1.22 秒	2022 秒	1659
3	65	1.03 秒	2027 秒	1964
4	64	0.99 秒	2203 秒	2237
5	66	0.99 秒	2012 秒	2042
6	70	1.05 秒	2025 秒	1936
7	78	1.44 秒	2015 秒	1401
8	69	1.14 秒	2027 秒	1777
9	68	1.11 秒	2022 秒	1822
10	64	0.97 秒	2022 秒	2086

いずれも，DTW の処理時間に対して 1400 倍～2200 倍の性能を確保している．従来手法である FTW と比較して 5 倍～10 倍の時間処理性能を示した．

このように，DIASTRA により，DTW と比較して応用上必要な精度を維持しつつ，処理速度が大幅に改善されることを確認した．

5. まとめ

本論文では，データシーケンスを二次近似誤差が極小となる不定長の区間へ分割し要約することにより，検索時の類似距離の評価時間を削減し高速化する類似検索方式 DIASTRA を提案した．

DIASTRA による実データの類似検索の精度検証により，ヒット率が単一回答のみの場合に 95.8%，6 件の回答を許容する場合に 99.0% を示すことを確認した.また問合せシーケンス長が 5000 の場合の検索時間比較について,DTW に対する通常の実装にて 100 分の 1 に間引いた検索処理に対しても，要約処理および検索時間に要する時間を合わせて 1400 倍以上の性能を確保していることを確認した．

課題として，本評価結果は SPRING 以外の従来手法に対しては処理時間面での改善が得られているが，SPRING による問合せシーケンス長 256 に対する性能評価 650000 倍という性能には及んでおらず，同一条件での追加評価を実施する必要がある．

提案技術は，単体を類似結果として提示するのではなく，結果の候補を複数表示する補遺的なシステムへの導入を想定しており，精度と処理速度のトレードオフを考慮して運用を行っていく．

参　考　文　献

[1] Myers C., and Rabiner L., “A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition,” Proc. ASSP-29, pp.284-296, Apr. 1981.

[2] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh. Indexing Multi-dimensional Time-series with Support for Multiple Distance Measures, Proc. of ACM SIGKDD, pp.216-225,2003.

[3] 中村，瀧，野宮，上原．AMSS: 時系列データの効率的な類似度測定手法，電子情報通信学会論文誌，D，Vol. J91-D，No.11, pp.2579-2588, 2008.

[4] S. Salvador and P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. TDM 2004.

[5] Sakurai Y., Yoshikawa M., and Christos F., FTW: fast similarity search under the time warping distance, Proc. PODS, pp.326-337, Jun. 2005.

[6] M. Morse, J.M. Patel. An Efficient and Accurate Method for Evaluating Time Series Similarity. Proc. of ACM SIGMOD 2007, 2007.

[7] 櫻井，Faloutsos，山室，ダイナミックタイムワーピング距離に基づくストリーム処理，DEWS2007 L6-5, 2007.

[8] UCI KDD Archive : http://kdd.ics.uci.edu/databases/el_nino/el_nino.html

付録 (各処理の擬似コード)

[入力]
STREAM: データシーケンス
KEY:問合せシーケンス
NUM: 考慮する類似上位件数

```
# 上位 NUM 件までの類似判定箇所を返す
Algorithm SimilaritySearch(STREAM, KEY, NUM)
  list := SimilarityEvaluation(STREAM, KEY)
  return list[1...NUM] のインデックス情報

# 時系列データの各箇所の類似評価距離を返す
# STREAM, KEY は時系列データ
Algorithm SimilarityEvaluation(STREAM, KEY)
  S_sum := Summarize(STREAM)
  K_sum := Summarize(KEY)
  distance_list = NULL      # 空のリスト
  for idx:=0 to Length(STREAM)-WINDOW_WIDTH do
    end_idx := idx+WINDOW_WIDTH-1
    sub_S_sum := STREAM[idx...end_idx]
    distance := GetDistance(sub_S_sum, K_sum)
    distance_list に{distance, idx}を追加
  endfor
  sorted_distance_list := Sort(distance_list)
  return sorted_distance_list

# 時系列データを要約要素の配列へ変換する
# DATA は時系列データ
Algorithm Summarize(DATA)
  summary := NULL   #要約要素のリスト
  begin_idx = 0
  end_idx := MIN_WIDTH
  prev_fit_info := NULL
  while end_idx < Length(DATA) do
    if prev_fit_info = NULL then
      fit_info := QuadraticFit(Data[begin_idx...end_idx])
    else
      diff_data := Data[end_idx]
      fit_info := DiffQuadFit(diff_data, prev_fit_info)
    endif
    if fit_info による近似誤差が許容範囲外 then
      summary に{fit_info, begin_idx, end_idx}を追加
      prev_fit_info := NULL
      beginning_idx := end_idx+1
      end_idx := begin_idx+MIN_WIDTH
    else
      prev_fit_info := fit_info
      end_idx := end_idx+1
    endif
  endwhile
  return summary

# 要約要素の配列間の距離を返す
# ONE, ANOTHER はいずれも要約要素の配列
Algorithm GetDistance (ONE, ANOTHER)
    min_distance := INFINITY
    foreach m in DPMatching(ONE, ANOTHER) do
      # m は ONE の要素と ANOTHER の要素との
      # マッチングの一つ
      distance := 0
      foreach {o, a} in m do
        # マッチング m において，ONE の要素 o と
        # ANOTHER の要素 a とがマッチ
        distance += GetElementalDistance(o, a)
      endforeach
      if distance < min_distance then
        min_distance := distance
      endif
    endforeach
    return min_distance
```

[定数]
WINDOW_WIDTH: 検索時のウィンドウ幅
MIN_WIDTH: 近似誤差を評価する最小データ数
[関数] (以下の関数の擬似コードは割愛する)
QuadraticFit: 二次関数近似を行う関数
DiffQuadFit: 増分から二次関数近似を更新する関数
DPMatching: 2 つの配列の可能なマッチングのリストを返す関数