社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

長大な時系列データの類似検索の研究

吉川　昂伯†　　石川　雅弘††　　陳　　漢雄†††　古瀬　一隆†††　大保　信夫†††

† 筑波大学大学院システム情報工学研究科，〒 305-8573 茨城県つくば市天王台 1-1-1
††† 筑波大学電子・情報工学系，〒 305-8573 茨城県つくば市天王台 1-1-1
†† つくば国際大学産業情報学科，〒 300-0051 茨城県土浦市真鍋 6-20-1
E-mail: †{yosikawa,chx,furuse,ohbo}@dblab.is.tsukuba.ac.jp, ††mi@tius.ac.jp

あらまし　株価などの経済データ、気温などの観測データ、あるいは音声データなど、様々な分野で時系列データが大量に蓄積されるようになってきた。蓄積されたデータを活用するための基本技術の一つとして、時系列データに対する類似検索の研究が行なわれている。時系列データに対する類似検索の研究では、同程度の長さの時系列データを多数集めたデータベースを検索対象とし、問合わせとして与えられた時系列データに類似したデータを検索するのが一般的である。しかし、例えば過去数年間の株価の動きの中から、最近数週間の値動きに類似した期間を検索するような、一本の長大な時系列データから問合せ時系列に類似した部分区間を検索する研究は少ない。本稿では、そのような検索を効率的に行うための手法を提案し、実験によりその有効性を示す。また、現在のところ解決されていないいくつかの問題について議論する。

キーワード　時系列データ，類似検索，部分区間検索，動的タイムワープ距離

A Study on Similarity Search for Very Long Time-Series Data

Takanori YOSHIKAWA†, Masahiro ISHIKAWA††, Hanxiong CHEN†††, Kazutaka FURUSE†††, and Nobuo OHBO†††

† Graduate School of Systems and Information Engineering, Tennodai 1-1-1, Tsukuba, Ibaraki, 305-8573, Japan
††† Institute of Information Sciences and Electronics, University of Tsukuba, Tennodai 1-1-1, Tsukuba, Ibaraki, 305-8573, Japan
†† Department of Social Informatics, Tsukuba International University, Tsuchiura, Ibaraki, 300-0051, Japan
E-mail: †{yosikawa,chx,furuse,ohbo}@dblab.is.tsukuba.ac.jp, ††mi@tius.ac.jp

Abstract　In recent years, huge amount of time-series data is accumulated in various domains, including economics, voice recognition, earth science and so forth. To utilize them, similarity search techniques are heavily studied. However, in the most studies, target database consists of large amount of data whose length are almost the same and a query is a time-series data of almost the same length. We focus on the case where the target database is a single very long time-series, and the length of a query is much shorter than that of the database. In this paper, we propose a new method to execute similarity searches in such situations, and show its efficiency by experimental results. Finally, some deficiency to be overcomed are disccused.

Key words　time-series data, similarity search, subsequence search, dynamic time warping distance

1. はじめに

近年、音声データ、株価、気温などの自然観測データ等、様々な分野において時系列データが大量に蓄積されるようになってきた。ここで時系列データとは、気温など経時変化する実数値を観測時間順に並べて得られる実数値のシーケンスである。蓄積されたデータを有効に活用するため、時系列データの分類やクラスタリングなどの研究が行なわれているが [1] [2]、そのための基本技術の一つとして重要となるのが類似検索である。類似検索においては二つの時系列データ間の類似度の定義が必要となるが、従来は (非類似度として) ユークリッド距離が用いられる事が多かった。長さ n の時系列データは n 次元ベクトルと

捉える事ができるからである。しかしユークリッド距離には、同じ長さ (次元) のデータ間でしか定義されない、時間軸上のずれに対応しきれないため (非) 類似度を適切に捉えられないなどの問題がある。そのため、時間軸上のずれを許容し、長さの異なるデータ間でも距離を与える事ができる動的タイムワーピング (DTW: Dynamic Time Warping) 距離が用いられるようになってきた [4] [5] [6] [7] 。DTW 距離の問題としては、計算コストが高い点があげられる。長さ m と n の二つの時系列データ間の DTW 距離の計算コストは $O(mn)$ である。そのため、DTW 距離を用いた類似検索においては距離計算コストの削減が重要となる。

時系列データに対する類似検索の研究では、同程度の長さの時系列データを多数集めたデータベースを検索対象とし、問合せ時系列データに類似したデータを検索するものが一般的である [8] [6] [9] 。本研究では、それらとは異なり、一本の長大な時系列データを問い合わせ対象とする。すなわちデータベースには一本の長大な時系列データのみが格納されているものとし、問合わせ時系列データの長さはデータベースに比べて大変短かいものとする。検索結果として与えられるのは、問合せに類似した、データベース時系列データの部分区間である。

データベースに格納されているデータから類似した部分区間を検索する研究としては [10] などがある。[10] ではデータベースには同程度の長さの時系列データが多数格納されており、各データから切り出した一定長の部分区間が空間索引を利用して索引付けされる。問合せが与えらえると、その各部分区間を問合せとして索引を利用する事で、効率的に類似した部分区間を検索している。しかし、索引付けのためにデータベース時系列を一定の長さ (sliding window 長) の部分区間に分解したり、後述する DTW 距離計算における時間軸のずれ幅を制限していること (global constraint) などの制約がある。また、実験で用いたデータセットも、データ数は 500 を越えるが平均長が 231 と、比較的短かい。そのため本研究の対象とする問題にはそのままでは適応できない。本研究では、時間軸上のずれにも制限は設けない。また、対象となるデータは長大である。

以上のように、既存研究の枠組みを本研究にそのまま適用することはできない。また、解となりうる部分区間の数が膨大であるため、データベース時系列が長くなると虱潰しの方法では現実的な時間で解を得ることが出来ないなどの問題もある。

本稿では、このような検索を効率的に行うための手法を提案し、実験によりその有効性を示す。また、解の取り零しの問題など、解決すべき問題についても論じる。

2. 時系列データ間の距離

2.1 ユークリッド距離

ユークリッド距離は、ベクトル間の距離として一般的に用いられている。また、時系列データは高次元ベクトルと捉えることもできるため、時系列データ間の距離としても広く用いられてきた。n 次元ベクトル $\boldsymbol{x}$,$\boldsymbol{y}$ 間のユークリッド距離 $D_E(\boldsymbol{x}$,$\boldsymbol{y}$) は下式で与えられる。ここで x_i は $\boldsymbol{x}$ の第 i 次元要素である。

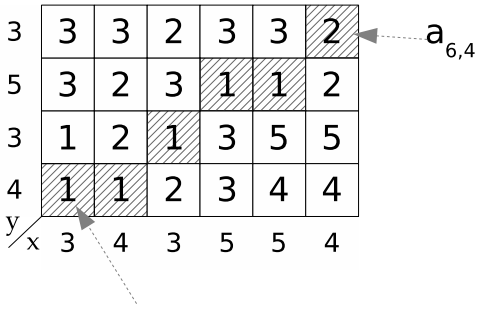

図 1　累積距離テーブル
Fig. 1　Accumulated Distance Table

$$D_E(\boldsymbol{x}\ ,\boldsymbol{y}\ ) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

式 (1) からも分かる通り、ユークリッド距離は同じ次元の要素同士の差を合算する事で距離を求めている。時系列データに当て嵌めれば、同一時間の値の差を合算する事で距離を求める事になる。しかし、例えば音声では「同じ」音が若干異なるピッチで発声されたり、あるいは観測のタイミングが 1 秒置きのデータと 2 秒置きのデータを比較したい場合など、必ずしも「同一時刻」の値同士を比較すれば良い訳ではない。また、同じ長さ (次元) のデータ間にしか距離が定義されないという問題もある。実世界を観測する事で得られる時系列データでは、観測機器の不具合などによりある時刻の観測データが欠損する事も考えられるが、そのような場合には対応できない事になる。このように時系列データ間の距離としてユークリッド距離を用いるのは不適切である場合が多い。

2.2 動的タイムワーピング距離 (DTW 距離)

ユークリッド距離の問題を緩和するものとして、DTW 距離 [3] が用いられるようになってきた。長さ m と n の時系列データ $\boldsymbol{x}$,$\boldsymbol{y}$ 間の DTW 距離 $D(\boldsymbol{x}$,$\boldsymbol{y}$) は下式で与えられる。

$$D(\boldsymbol{x}\ ,\boldsymbol{y}\ ) = D_0(x_1, y_1) + \min\left\{\begin{array}{l} D(\boldsymbol{x}_{2:}, \boldsymbol{y}\ ) \\ D(\boldsymbol{x}\ , \boldsymbol{y}_{2:}) \\ D(\boldsymbol{x}_{2:}, \boldsymbol{y}_{2:}) \end{array}\right\} \tag{2}$$

$$D_0(x, y) = |x - y| \tag{3}$$

ただし、$D_0(x, y)$ は引数の差の絶対値の単調増加関数ならば何でも良く、例えば $|x - y|^2$ が用いられる事もある。本稿では式 (3) に示す通り差の絶対値を用いる。また、$\boldsymbol{x}_{i:}$ は $\boldsymbol{x}$ の第 i 要素から始まる接尾辞となる部分時系列データであり、$\boldsymbol{x}_{i:j}$ は第 i 要素から始まり第 j 要素で終わる部分時系列データである。DTW 距離は異なる長さのデータ間でも定義されており、また、時間軸上のずれも許容している。

2.2.1 DTW 距離の計算

DTW 距離は、動的計画法を用いて計算される。ここでは二つの時系列データ $\boldsymbol{x}\ = \langle x_1, x_2, x_3, x_4\rangle = \langle 4, 3, 5, 3\rangle$ と $\boldsymbol{y}\ = \langle y_1, y_2, y_3, y_4, y_5, y_6\rangle = \langle 3, 4, 3, 5, 5, 4\rangle$ を例に説明する。

まず、二つの時系列データの長さに合わせて 4×6 のテーブルを用意する (図 1 参照)。行方向は$\boldsymbol{x}$ に、列方向は$\boldsymbol{y}$ に対応

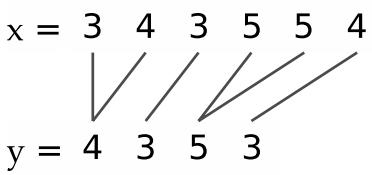

図 2　ワーピングパスが与える要素の対応
Fig. 2　Element Correspondence by the Warping Path

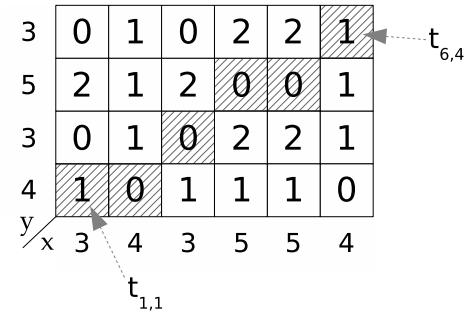

図 3　差分テーブル
Fig. 3　Difference Table

しており、最下行最左要素を $a_{1,1}$、最上行最右要素を $a_{6,4}$ とする。テーブルが空の状態から計算を開始し、全ての要素に値を埋めた時点で計算が終了する。始めに $a_{1,1}$ に $D_0(x_1, y_1)$ を格納し、$a_{1,2}$〜$a_{1,4}$ には $a_{i,j} = a_{1,j-1} + D_0(x_1, y_j)$ を、$a_{2,1}$〜$a_{6,1}$ には $a_{i,j} = a_{i-1,1} + D_0(x_i, y_1)$ を格納する。左、下、左下の三つの接する要素がすでに埋まっている他の各要素には、下式に従って求めた値を格納する。

$$a_{i,j} = D_0(x_i, y_j) + \min \left\{ \begin{array}{l} a_{i-1,j} \\ a_{i,j-1} \\ a_{i-1,j-1} \end{array} \right\} \qquad (4)$$

全ての要素が埋められた時、最上最右要素である $a_{6,4}$ の値が DTW 距離となる。また、最上行の各要素 $a_{i,4}$ は、$\boldsymbol{y}$ と $\boldsymbol{x}$ の各接頭辞 $\boldsymbol{x}_{:i}$ との DTW 距離 $D(\boldsymbol{y}, \boldsymbol{x}_{:i})$ を与える。$a_{1,1}$ から $a_{6.4}$ までを、右上に向かって辿るパス (ワーピングパス) は多数存在するが、図中網掛けで示したのは式 (4) 第二項で選択された要素からなるパスであり、DTW 距離を与える時の$\boldsymbol{x}$と$\boldsymbol{y}$の要素の対応を示している。この時の対応を図 2 に示す。$a_{1,1}$ から $a_{6,4}$ まで辿るワーピングパスは他にも存在するが、DTW 距離とは、全ての可能なパスのうち (すなわち全ての可能な対応付けのうち)、$a_{6,4}$ に最小の値を与えるパスから得られる距離のことである。

2.2.2　DTW 距離と差分テーブル

ここで差分テーブルと呼ぶテーブルを導入し、差分テーブル上での DTW 距離の求め方を説明する。差分テーブルとは、累積距離テーブルと同サイズのテーブルであり、その (i, j) 要素 $t_{i,j}$ の値は $D_0(x_i, y_j)$ である。図 3 に例を示す。解を与えるワーピングパスが与えられた時、パス上の要素の値を合計する事で DTW 距離を求める事ができる。なぜなら、差分テーブルの各要素は式 (2) の第一項に対応しているからである。また、これ以外のパス上の値を合計すると、必ず解よりも大きな値となる。

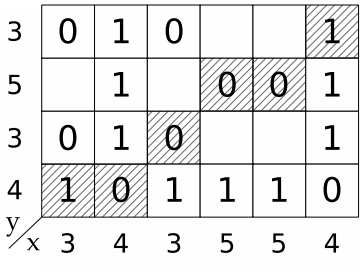

図 4　小さな値だけを残した差分テーブル (1 以下の要素のみ)
Fig. 4　Difference Table with Smaller Values (less than two

図から、解を与えるパスは、差分テーブル上では値の小さな要素上を通る傾向にある事が分かる。なぜなら、全ての可能なパスの中で、値の合計が最小となるものが解を与えるからである。

3. 提案手法

3.1 問題定義

本研究が対象とする k-最近傍部分区間検索を定義する。あらかじめ長さ m のデータベース時系列データ $\boldsymbol{d} = \langle d_1, d_2, \cdots, d_m \rangle$ は与えられているものとする。また、問合せとして与えられるのは、長さ n の時系列データ $\boldsymbol{q} = \langle q_1, q_2, \cdots, q_n \rangle$ であり、$n \ll m$ とする。

[定義 1]（k-最近傍部分区間検索）
長さ m の時系列データ$\boldsymbol{d}$ に対し、長さ $n(\ll m)$ の問合せ時系列データ$\boldsymbol{q}$ と整数 $k(>0)$ が与えられた時、$\boldsymbol{q}$ との DTW 距離 $D(\boldsymbol{q}, \boldsymbol{d}_{i:j})$ が最も小さい k 個の$\boldsymbol{d}$ の部分区間 $\boldsymbol{d}_{i:j}$ を求める検索を、k-最近傍部分区間検索と呼ぶ。

以下では、k-最近傍部分区間検索を単に k-NN 検索 (k-Nearest Neighbour Search) と呼ぶ。また、時系列データを対象とした k-NN 検索では、解となる部分区間に関していくつか現実的な問題が発生するが、それらについては後述する。

3.2 動機付け

2.2.2 で述べたように、DTW 距離の計算は差分テーブル上で最小の合計を与えるワーピングパスを求める事と等しい。ここで、差分テーブルの要素のうち、値の小さなものだけを残した状態を考える。値が 1 以下の要素のみを残した例を図 4 に示す。この時、$t_{1,1}$ と $t_{6,4}$ を結ぶパスは二つしか残っていないが、その一つが解を与えている事が分かる。「解を与えるワーピングパスは、差分テーブル上では値の小さな要素上を通る傾向にある」からである。常にこの例のように解となるパスが残る保証はないが、値の小さな要素だけに注目する事で可能なパスの数を削減し、差分テーブルに基づいて低コストで距離を求められる可能性があるのである。

3.3 提案手法の概要

提案手法の概要を、図 5 に示す差分テーブルを例に説明する。

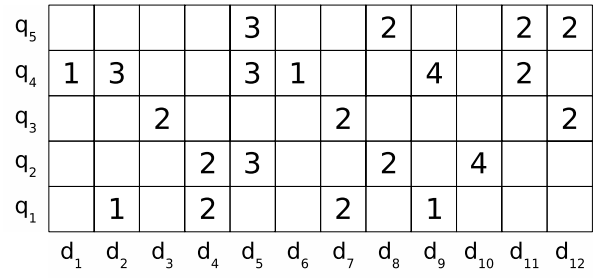

図 5　k-NN 検索における差分テーブル ($m = 12, n = 5$ の例)
Fig. 5　Difference Table for k-NN

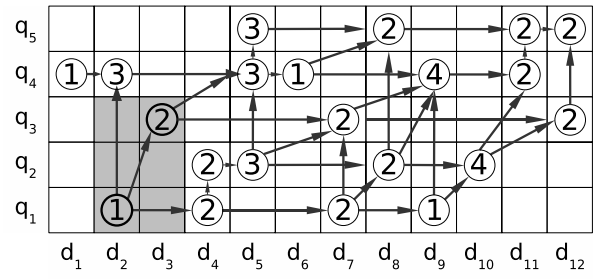

図 6　マークされた要素を結ぶ事で得られる検索グラフ
Fig. 6　Search Graph built from Marked Elements

3.3.1　検索グラフによるパスの削減

作図の都合上ここでは$\boldsymbol{d}$ の長さは $m = 12$、$\boldsymbol{q}$ の長さは $n = 5$ としたが、本研究では $n \ll m$ の場合を想定しているため、実際には非常に横長のテーブルとなる。図 5 では、値が 4 以下の要素にのみ値を記入してある。差分テーブルに具体的な値を記入する事を、要素をマークすると呼ぶ。

3.2 で述べたように、解となるワーピングパスは値の小さな要素を通る傾向にあるため、なるべく多くのマークを通過するパスほど解に近い値となる可能性が高いと期待できる。しかし図 4 とは異なり、マークされた要素だけでは完全なパスを構成できない。

ここで、検索グラフと呼ぶ、マークされた要素 $t_{i,j}$ をノードとする有向グラフを考える。図 6 に検索グラフの例を示す。検索グラフは、マークされた二つの要素 $t_{i,j}, t_{k,l}$ 間に以下の条件が成り立つ場合に有向辺 $\langle t_{i,j},\, t_{k,l} \rangle$ を張る事で得られる。

（1）　$i \leqq k$ かつ $j \leqq l$ (ただし $t_{i,j} \neq t_{k,l}$)。

（2）　$i \leqq u$ かつ $j \leqq v$ かつ $u \leqq k$ かつ $v \leqq l$ なる $t_{u,v}$ が存在しない (ただし $t_{i,j} \neq t_{u,v}$ かつ $t_{u,v} \neq t_{k,l}$)。

図 6 に図 5 から得られる検索グラフを示す。

提案手法の基本方針は、差分テーブル上のワーピングパスを全て考慮する代わりに、検索グラフ上のパスのみを考慮する事で考慮すべきパスの数を削減し、また次に述べる下限距離を用いた枝刈りを行なう事で、効率的に距離の小さなパスを発見する事である。

3.3.2　検索グラフ上の距離の下限値

検索グラフにおいて、q_1 から q_n、すなわち問合せ全体に渡るパスを考える。例えば、$t_{2,1} \rightarrow t_{3,3} \rightarrow t_{5,4} \rightarrow t_{5,5}$ は、q_1 から q_5 に渡るパスであり、$\boldsymbol{q}$ 全体と$\boldsymbol{d}$ の部分区間 $\boldsymbol{d}_{2:5}$ の対応を表す。このパス上の距離 (マーク要素の値の和) は 9 であるが、

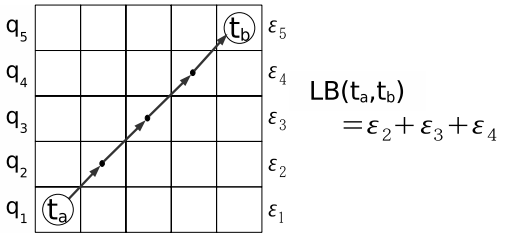

図 7　正方形の場合の下限距離計算
Fig. 7　Lower Bound (square case)

これが$\boldsymbol{q}$と $\boldsymbol{d}_{2:5}$ の DTW 距離となる保証はない。第一に、検索グラフ上のパスにはマークされた要素しか含まれておらず、途中の非マーク要素を無視しているからである。第二に、差分テーブル上で非マーク要素上のパスが確定していないためである。しかし、間に非マーク要素を狭んでいても、マーク間のパスが取り得る値の下限値は求める事ができる。従って$\boldsymbol{q}$と $\boldsymbol{d}_{2:5}$ の DTW 距離の下限値も求める事ができる。

差分テーブル上の $t_{2,1}$ から $t_{3,3}$ に至るパスを考える。両要素により形成される矩形区間 (図 6 参照) でマークされているのは $t_{2,1}$ と $t_{3,,3}$ のみであり、それぞれの値は 1 と 2 であるため、パスがこの二つの要素を通る以上、最低でも値の合計は 3 である。問題は $t_{2,1}$ から $t_{3,3}$ の間でどの非マーク要素を通るかであるが、可能なパスは

（1）　$t_{2,1} \rightarrow t_{3,1} \rightarrow t_{3,2} \rightarrow t_{3,3}$

（2）　$t_{2,1} \rightarrow t_{3,2} \rightarrow t_{3,3}$

（3）　$t_{2,1} \rightarrow t_{2,2} \rightarrow t_{3,3}$

（4）　$t_{2,1} \rightarrow t_{2,2} \rightarrow t_{2,3} \rightarrow t_{3,3}$

の 4 つである。どのパスでも q_2 に対応する要素を必ず一つは通過する必要があり、かつそれだけで $t_{3,3}$ に到達することが可能であるため、$t_{2,1}$ から $t_{3,3}$ に達するためのパスの与える下限値は 7(= 3 + 4) である。なぜなら、マークされている要素の上限値は 4 であるため、マークされていない要素の値の下限値は 4 だからである。

このように、閾値以下の要素のみをマークする事で、マーク要素間のパスの与える下限値を求める事ができる。検索グラフ上の探索と下限値を用いた枝刈りにより、効率的に解の候補となる部分区間を求める事ができる。

3.3.3　有向辺の下限値の計算方法

ここで、検索グラフ上の一つの有向辺の下限距離の求め方について、矩形の形状ごとに説明する。

a）　正方形の場合

二つのマーク要素をそれぞれ t_a, t_b とし、形成される矩形が正方形の場合の例を図 7 に示す。ここで、要素 t_a, t_b の値をそれぞれ a, b とする。また、閾値は各 q_i 毎に異なる値とし、それぞれを ε_i で表す。

t_a と t_b を結ぶパスは、少なくとも q_2〜q_4 のそれぞれに対応する要素を一つずつ通過する必要があり、かつそれだけで t_b に達する事もできる。したがって、この有向辺の距離の下限は $LB(t_a, t_b) = \varepsilon_2 + \varepsilon_3 + \varepsilon_4$ である。

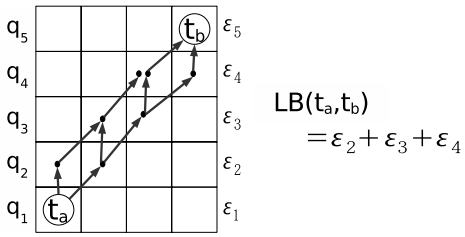

図 8　縦長の長方形の場合の下限距離計算
Fig. 8　Lower Bound (vertical case)

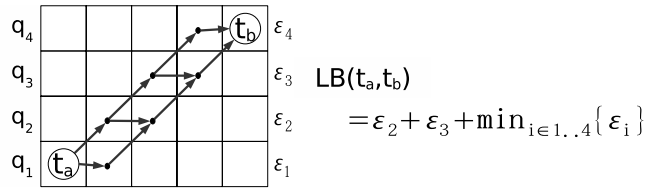

図 9　横長の場合の下限距離計算
Fig. 9　Lower Bound (horizontal case)

b）　縦長の長方形の場合

図 8 に示したのは、縦長の長方形になる例である。この時、t_a から t_b に達するには、少なくとも q_2〜q_4 に対応する要素を一つずつ通過する必要があり、かつそれだけで t_b に達する事もできる。よって下限距離は $LB(t_a, t_b) = \varepsilon_2 + \varepsilon_3 + \varepsilon_4$ である。

c）　横長の長方形の場合

図 9 は横長の長方形の場合である。t_a から t_b に達するには、少なくとも q_2, q_3 に対応する要素を一つずつ通過する必要がある。t_b に到達するにはさらに、横長の分だけ q_1〜q_4 のどこかを余計に通過する必要がある。下限値を与えるのは、その中で最小閾値の要素を通過した時である。よって下限値は $LB(t_a, t_b) = \varepsilon_2 + \varepsilon_3 + min_{k\in 1..4}\{\varepsilon_k\}$ である。

d）　一般の場合

以上から、一つの有向辺の距離の下限値は下式で与えられる。ここで n, m はそれぞれ矩形の縦幅と横幅である。

$$LB(t_a, t_b) = \begin{cases} \sum_{i=2}^{n-1} \varepsilon_i & (n \geqq m) \\ \sum_{i=2}^{n-1} \varepsilon_i + (m-n)\min\{\varepsilon_k\} & (n < m) \end{cases} \tag{5}$$

e）　多段の場合の下限距離

ここまでは、一つの有向辺を辿る場合を考えた。次に、一つ以上の有向辺を既に辿った後、さらに新たな有向辺を辿る場合の下限距離について説明する。図 10 は、既に t_a から t_b に辿った後に、さらに t_c へ辿るところを表している。$t_a \to t_b$ 間の下限距離は既に $L(t_a, t_b)$ として求まっているとする。また、$t_b \to t_c$ 間の下限距離も上述の計算方法を適用して $LB(t_b, t_c)$ として求まっているとする。この時、$t_a \to t_c$ 間の下限距離 $L(t_a, t_c)$ は、下式で与えられる。

$$L(t_a, t_c) = \min \left\{ \begin{array}{l} L(t_a, t_b) + b + LB(t_b, t_c) \\ LB(t_a, t_c) \end{array} \right\} \tag{6}$$

$LB(t_a, t_c)$ は、t_a, t_c 間にマーク要素がないと考えて求めた下限距離である。$LB(t_a, t_b) + b + LB(t_b, t_c)$ をそのまま下限値とできないのは、図示した二つの矩形の外を通ったパスを考慮していないからである。

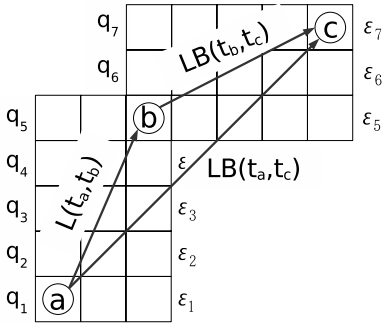

図 10　多段の場合の下限距離
Fig. 10　Lower Bound (Multiple Step)

3.4　提案手法の流れ

提案手法では、検索グラフとその上の下限値を利用し、以下の 3 ステップで類似区間を検索する。

（1）　マーク付け ··· 値の小さな差分要素をマーク付けする。

（2）　初期解候補の選択 ··· 次のステップの枝刈りに用いるため、k 個の部分区間を解候補として選択し、問合せとそれらとの実 DTW 距離を計算する。

（3）　検索グラフの探索 ··· k 個の解候補への実距離と検索グラフ上のパスの下限距離を用いて、枝刈りを行ないながら検索グラフを探索し、解を求める。

それぞれについて以下で解説する。

3.4.1　マーク付け

マーク付けでは、各要素のうち $D_0(d_i, q_j)$ の値の小さなものを見付ける必要がある。しかし全ての $D_0(d_i, q_j)$ を計算する必要はなく、各 q_j について値の近い d_i を見付ければ良い。これを効率的に行なうために、$\boldsymbol{d}$ を整列した上で各 q_j の二分探索を行なう。

マーク付けするための閾値は、各 q_j 毎に異なる値を許し、それぞれを ε_j で表す。この値はパラメータとして与える必要があるが、どの程度の値を指定すれば良いかを決定するのは難しい。そこで、充填率 f と呼ぶパラメータを導入し、データベース長 m の何%をマークするかを指定する事にした。充填率 f が指定されたとき、各 q_j に近い値を持つ f%の要素がマーク付けされ、その中で最も大きな値が閾値 ε_j を与える。

充填率が高ければ高いほど検索グラフのノードと有向辺の数も増加するため、検索の効率に影響すると考えられる。

3.4.2　初期解候補の選択

ステップ 3 の検索グラフの探索では枝刈りを行なうが、そこで用いるのは、各パスの下限距離と、既に得られている k 個の候補解の中の最大距離である。そのため、それに先立って k 個の解候補とそれらと $\boldsymbol{q}$ との実 DTW 距離を求める必要がある。

効率的に枝刈りを行なうためにはできるだけ実解に近い解候補を得るのが望ましいが、一方で解候補の選択に大きなコストを掛けるのは避けたい。ここでは、ヒューリスティックに解候補を選択する **RF0** 法を用いた。

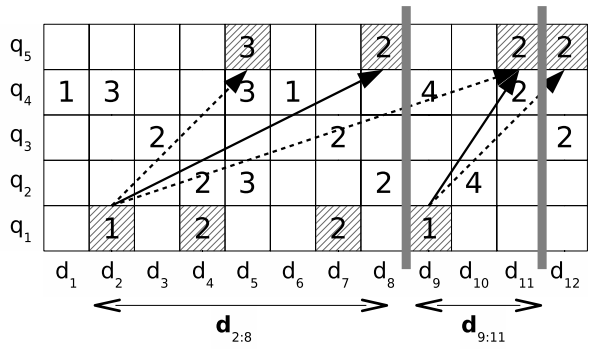

図 11　RF0 の動作イメージ
Fig. 11　Sketch of RF0 Execution

なお、検索グラフの探索に先立って別途解候補を求める論理的必然性はない。例えば探索の初期に発見した k 個の部分区間を解候補とすれば足りるからである。ここで前もって **RF0** 法を適用するのは、できるだけデータベースの全域から解候補を選択することで、データの偏りなどの影響を排除したいからである。

3.4.3　RF0 アルゴリズム

RF0 では、q_1 に対応するマーク要素 $t_{s,1}$ と q_n に対応するマーク要素 $t_{e,n}$ の組み合わせで決定される部分区間 $\boldsymbol{d}_{s:e}$ との距離 $D(\boldsymbol{q}, \boldsymbol{d}_{s:e})$ を順次計算し、その中で距離の小さい k 個を選択する。ただし、$\boldsymbol{d}_{s:e}$ が候補として選択された場合には、それと重ならないように、次の候補は d_{e+1} 以降から始まる区間を選択する。図 11 に **RF0** の動作イメージを示す。d_2 を始点とする部分区間としては $\boldsymbol{d}_{2:5}$, $\boldsymbol{d}_{2:8}$, $\boldsymbol{d}_{2:11}$ などがあるが、ここでは $D(\boldsymbol{q}, \boldsymbol{d}_{2:5}) < D(\boldsymbol{q}, \boldsymbol{d}_{2:8}) > D(\boldsymbol{q}, \boldsymbol{d}_{2:11})$ であるとする。この時、d_2 を始点とする部分区間としては、$\boldsymbol{d}_{2:8}$ のみを選択し、既に得られている k 個の候補よりも距離が小さければ解候補とする。続いて順次 d_9 以降のマーク要素を始点とする組み合わせに対して同様の処理を行なう。最終的には k 個の重ならない部分区間が解候補として選択される。

3.4.4　検索グラフの構築と探索

k 個の初期候補と実距離が与えられてから、検索グラフの探索を行なう。基本的な手法は k-NN 検索一般と同じであり、k 個の解候補をサイズ k のヒープに積み、下限距離がヒープ内の最大距離よりも小さな区間が見付かれば実 DTW 距離を計算する。もしも実距離もヒープ内最大距離よりも小さければ、それと新たな候補を入れ換える。

検索グラフの構築は、3.3.1 で示した二条件を満たすマーク要素間に有向辺を張る事で行なわれる。また、グラフ構築後に q_1 に対応する各マーク要素を始点として探索を行なうことで解候補を抽出する事ができる。

3.4.5　検索グラフの構築

検索グラフの構築は、全マーク要素を順番にチェックし、その要素から出る全ての有向辺を作成する事で行なう。要素をチェックする順序は $t_{1,1}, t_{1,2}, t_{1,3}, \cdots, t_{2,1}, t_{2,2}, t_{2,3}, \cdots$ である。

3.5　提案手法の問題点

以上のように、本手法ではマーク要素のみをノードとするグラフを探索する事で解候補を選択する。そのため、始点と終点がマークされていない部分区間は解候補としても選択されない。例えば $t_{10,1}$ がマークされていなければ、$\boldsymbol{d}_{10:k}$ なる部分区間は選択されようがない。すなわち、充填率が低ければ低いほど、取りこぼされる解が増える事になる。

また、例えば実解が $\boldsymbol{d}_{10:100}$ であった場合に、それと完全には一致しないが部分的に重なる $\boldsymbol{d}_{30:120}$ などが検索結果として得られる事も考えられ、検索結果の評価には注意が必要である。そのため、本稿で示す実験では再現率と適合率という指標を用いて精度を評価している。再現率 $Recall$ と適合率 $Precision$ をそれぞれを以下のように定義する。ここで$\boldsymbol{A}$は正解区間、$\boldsymbol{C}$は検索結果の部分区間である。

$$Recall = \frac{\boldsymbol{A}\text{と}\boldsymbol{C}\text{の重なりの長さ}}{\boldsymbol{A}\text{の長さ}} \tag{7}$$

$$Precision = \frac{\boldsymbol{A}\text{と}\boldsymbol{C}\text{の重なりの長さ}}{\boldsymbol{C}\text{の長さ}} \tag{8}$$

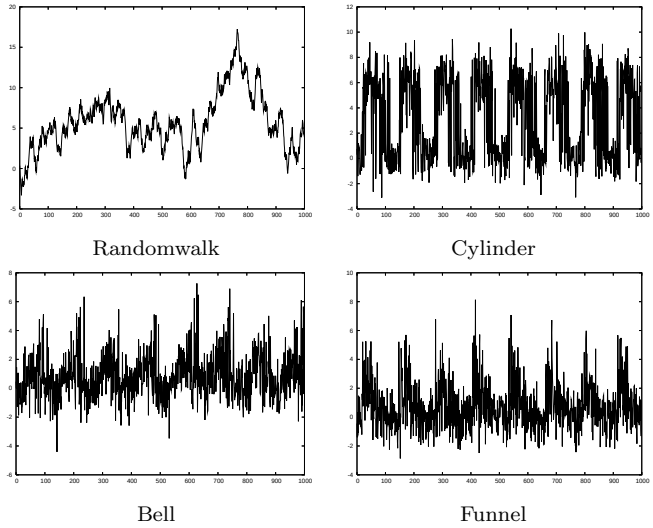

図 12　合成データの例
Fig. 12　Example of Synthetic Dataset

4.　実　　験

4.1　実験データと問合せ

提案手法の効果を検証するため、実験を行なった。用いた時系列データは、非周期性のランダムウォークデータと、周期性の Cylinder, Bell, Funnel の二種類の合成データである。Cylinder, Bell, Funnel データは [11] の方法で生成した。図 12 に各データの例を示す。問合せは各データから切り取った部分時系列とし、各データに対して 10-NN 検索を 20 回行ない、その平均で評価した。

4.2　比 較 手 法

比較対象としたのは、問合せ$\boldsymbol{q}$とデータベースの接尾辞 $\boldsymbol{d}_{1:}, \boldsymbol{d}_{2:}, \boldsymbol{d}_{3:}$ との DTW 距離を順次累積距離テーブルを用いて順次計算する方法である。$D(\boldsymbol{q}, \boldsymbol{d}_{i:})$ を計算する際には $\boldsymbol{d}_{i:k}$ なる部分区間との距離も求まるため、全ての部分区間をチェックする事ができる。ただし、$\boldsymbol{d}_{i:}$ の末尾まで計算するのは無駄が大きいため、累積距離テーブルを埋める際に k 番目の候補解

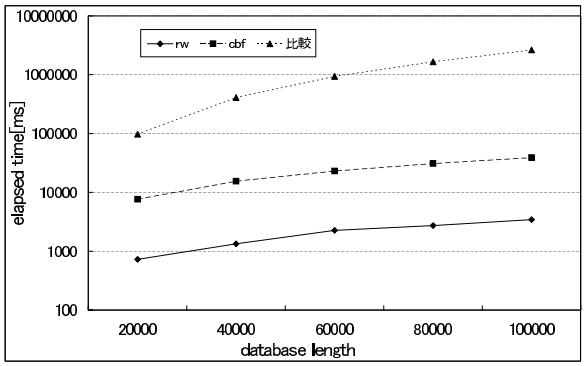

図 13　データベース長に対する検索時間の変化
Fig. 13　Database Length vs. Elapsed Time

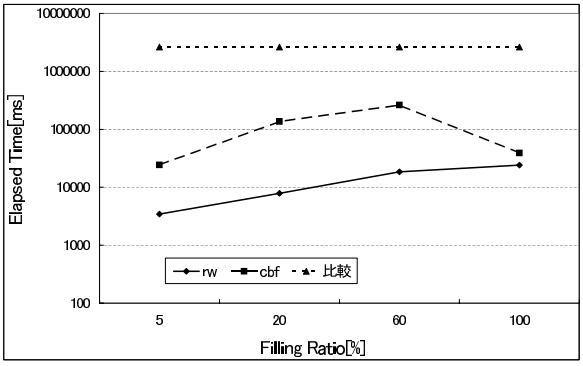

図 14　充填率に対する検索時間の変化
Fig. 14　Filling Ratio vs. Elapsed Time

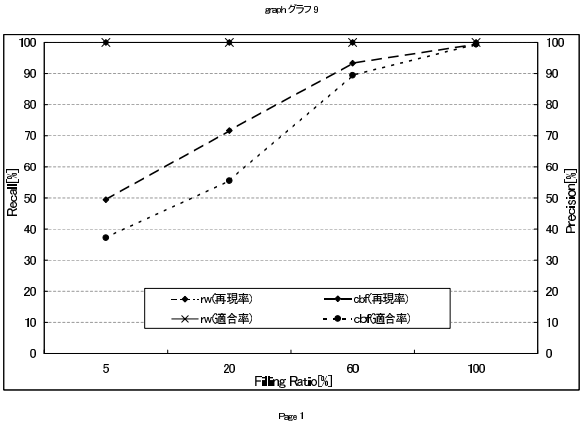

図 15　充填率に対する検索精度の変化
Fig. 15　Filling Ratio vs. Recall & Precision

の最大距離よりも大きな値しか求まらなくなった時点で計算を打ち切る。

4.3 実験結果

図 13 は、データベース時系列データの長さに対する実行時間の変化を示したものである。時間軸は対数目盛である。Cylinder, Bell, Funnel の三つのデータセットについては、それらの平均を CBF として示した。rw はランダムウォークである。充填率は、ランダムウォークデータで 5%、CBF データで 100%である。この時いずれも再現率と適合率は 100%である。

グラフからデータベース長に関わらず提案手法の速度が比較手法を大幅に上回っている事がわかる。データベース長が 100,000 の時には、ランダムウォークデータでは約 800 倍の、CBF データでは約 70 倍の速度向上がみられた。両データ共、データベースが長くなるに従い速度向上の比率が高まる事も確認できる。

速度向上比率のデータセットによる差は、一次的には充填率の差によるものと考えられるが、根源的にはデータセットの特徴に由来すると考えられる。その生成方法に由来し、ランダムウォークデータは、時刻 T の値と $T+1$ の値の差は最大でも 1.0 である。それに対し CBF データの値変動の幅は大きい。そのため、低い充填率でも高い再現率や適合率が得られるランダムウォークデータに対して、CBF データでは高い充填率を必要とされると考えられる。これが検索グラフのノード数と有向辺の数を増加させ、速度向上の足枷となったと考えられる。

図 14 は、充填率と実行時間の関係を示すグラフである。データベース長は 100,000 である。グラフから、どの充填率でも提案手法により大幅な速度向上が得られる事が確認できる。

ランダムウォークデータでは、充填率の上昇に従って実行時間も増加している事が分かる。これは検索グラフのノード数と有向辺の数が増加することでコストが上昇したためだと考えられる。

CBF データでは、充填率 60%までは単調に増加しているが、100%時には 20%時を上回るパフォーマンスを示している。ランダムウォークデータ同様の考察からはこの現象は理解し難いが、値変動が大きいというデータの性質から解釈することができる。

検索グラフ探索時の枝狩りには下限距離が利用されるが、充填率が 100%の時には、厳密な下限距離が求められるため、枝刈りがより効率良く行なえる。そのため実行時間が抑えられたと考えられる。なお、DTW 距離計算回に着目すると、充填率が 100 %時は 60 %時に比べて約 300 分の 1 に抑制されている事が確認できた。CBF データのように値変動が激しいものには、このような傾向が見られると考えられる。

図 15 は、充填率と再現率・適合率の関係を示すグラフである。データベース長は 100,000 である。ランダムウォークデータでは、充填率 5%ですでに再現率・適合率共に 100%となっており、低い充填率でも高い精度を得ることができることを確認できた。上述のとおり、値変動幅の小さい、穏やかに変化するデータであるためと考えられる。

CBF データでは、充填率を上げることによって再現率・適合率が上昇し、充填率 100%でようやく再現率・適合率も 100%となることが確認できた。この理由も上述した通り、CBF データは値の変動が激しいため、高い充填率でないと解となるべき要素をマークできなかったためと考えられる。また、CBF データでは再現率が適合率より高くなる傾向がみられるのも同じ理由であると考えられる。

5. まとめと今後の課題

本稿では、一本の長大な時系列データを検索対象とし、問合せ時系列データに類似する部分区間を効率的に検索する手法を

提案した。類似尺度にはDTW 距離を用いたが、DTW 距離の計算は高コストであり、長大な時系列データを対象にした場合の計算コストは膨大である。計算コストを削減するため、提案手法では部分区間探索の問題をグラフの探索と枝刈りの問題に還元し、下限距離による枝刈りの方法を示した。また、二種の合成データによる k-最近傍部分区間検索実験により、提案手法の有効性を示した。

今後の課題としては、さらに長いデータや実データによる実験を行なう必要がある。関連研究のうち長大な時系列データへの適用が可能なものは実装し、比較実験を行う必要もある。また、パフォーマンスはデータの特徴と充填率に影響されることから、データの特徴を考慮した上で充填率を設定する必要があるが、適切な値を決定するのは困難と考えられる。そこでデータの偏りなどの特徴を判断して充填率を自動設定する方法も考えたい。

また、本手法では始点と終点がマークされた区間しか検索できない。ランダムウォークデータでは低い充填率でも 100%の精度を示すことができたが、CBF データでは充填率を上げる必要があり速度劣化につながっている。効率を落とさずに精度を上げるための方法として、問合せの先頭と末尾要素 (q_1, q_n) に対応する要素の充填率のみを 100%とし、それ以外の充填率を下げる事などが考えられる。

類似尺度そのものについて考えると、ユークリッド距離の不備を補うものとして利用される DTW 距離にも、まだ不備があると言える。例えば、データの中に異常値などのはずれ値があった場合、DTW 距離でもその値を無視することはできないため、そのような値を含んだ区間を検索する事は困難である。このような問題を回避するためには、類似尺度として最長共有部分列 (LCS: Longest Common Subsequence) に基づくものを採用するのが望ましいが、本手法でもそれらへの対応を考えたい。

文　献

[1] Chotirat Ratanamahatana and Eamonn Keogh, “Making Time-Series Classification More Accurate Using Learned”, In proc. of 4th SIAM International Conference on Data Mining, 2004.

[2] Li Junkui, Wang Yuanzhen and Li Xinping, ”LB_HUST: A Symmetrical Boundary Distance for Clustering Time Series”, In proc. of the 9th International Conference of Information Technology, 2006.

[3] Joseph B. Kruskal and Mark Liberman, “The Symmetric Time-Warping Problem: From Continuous to Discrete”, Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, pp. 125-161 Addison-Wesley, 1983

[4] Selina Chu, Eamonn Keogh, David Hart and Michael Pazzani, “Interactive Deepening Dynamic Time Warping for Time Series”, In proc. of 2nd SIAM International Conference on Data Mining, 2002.

[5] Eamonn Keogh and Michael Pazzani, “Scaling up Dynamic Time Warping for Datamining Applications”, In proc. of 6th ACM SIGKDD Knowledge Discovery and Data Mining, 2000.

[6] Eamonn Keogh, “Exact Indexing of Dynamic Time Warping”, In proc. of 28th International Conference on VLDB, 2002.

[7] Sang-Wook Kim, Sanghyun Park, et al., “An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases”, In proc. of 17th ICDE, 2001.

[8] Yasushi Sakurai, Masatoshi Yoshikawa and Christos Faloutsos ”FTW: Fast Similarity Search under the Time Warping Distance”, In proc. of ACM PODS Conference 2005, pp.326-336, 2005.

[9] 大桃諭, 陳漢雄, 古瀬一隆, 大保信夫, 「タイムワーピングに基づく時系列データの類似検索：次元縮小による効率化」, DBSJ Letters Vol.4, No.1, pp.1-4

[10] Teddy Siu Fung Wong and Man Hon Wong, “Efficient Subsequence Matching for Sequences Databases under Time Warping”, In proc. of 7th International Database Engineering and Applications Symposium, 2003.

[11] Hansheng Lei, Venu Govindaraju ”Regression Time Warping for Similarity Measure of Sequence”, In proceedings of the Fourth International Conference on Computer and Information Technology, 2004