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

大桃　諭† 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
E-mail: †{oomomo,chx,furuse,ohbo}@dblab.is.tsukuba.ac.jp

あらまし　近年，時系列データは様々な分野で扱われるようになり，大規模な時系列データベースに対する効率的な類似検索が重要な課題となっている．これまでは時系列データの類似度にユークリッド距離が使用されていたが，時間軸方向のずれに柔軟性を持たせたタイムワーピング距離が近年注目をあびている．しかし，タイムワーピング距離の計算時間は非常に長く，しかも距離公理の三角不等式が成り立たないため，伝統的な索引技法の適用は困難である．そこで本研究では，時系列データに対する新しい次元縮小法と下界関数を提案する．そして，この下界関数をフィルタリングに使用したときの検索効率が，関連研究で提案されている下界関数によるものよりも優れていることを実験によって実証する．

キーワード　時系列，類似検索，タイムワーピング，下界関数

Efficient Search of Similar Time Series under Time Warping with Dimensionality Reduction

Satoshi OOMOMO†, Hanxiong CHEN††, Kazutaka FURUSE††, and Nobuo OHBO††

† Graduate School of Systems and Information Engineering, University of Tsukuba　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
E-mail: †{oomomo,chx,furuse,ohbo}@dblab.is.tsukuba.ac.jp

Abstract　Recently, time series data is produced and treated in many fields, and efficient similarity search in large time series databases has been important subject. Instead of Euclidean distance, similarity between time series data measured by time warping (TW) distance which allows out of phase in the time axis has got a lot of attention in recent years. Efficient indexing is necessary in such search because TW distance is computationally expensive. However, since TW violates the triangle inequality, traditional indexing techniques is powerless. To overcome this problem, in this paper we propose a new lower bound function based on dimensionality reduction. This enables efficient indexing hence filtering in similarity search of time series data. We show by experimental results that our lower bound function outperforms other ones proposed in related works.

Key words　time series, similarity search, time warping, lower bound function

1. はじめに

時系列データとは，時刻 i の時の値を v_i とすると，$V = v_1,\cdots,v_i,\cdots,v_N$ で表される実数のシーケンスである．時系列データの例としては，株価や為替レート，心電図，気温・湿度変化などがあり，ここに挙げた以外にも多く存在する．このように科学，医療，経済，工学などの様々な分野で時系列データが扱われている．さらに近年では，扱うデータ量も非常に多くなってきている．このような現状において，時系列データの効率的な類似検索の必要性が高まっている．例えば，最近の株価の動向と類似したものを過去のデータから見つけ出して，今後の動向を予測するといったことが挙げられる．

類似検索を行うためには，2 つの時系列間の類似度を定義する必要がある．現在，時系列の類似度として最もよく使用されているものはユークリッド距離である [2], [4], [8], [11]．しかしユークリッド距離には，2 つの時系列の長さが等しくなけれ

ばならない，時間軸方向のわずかなずれに対しても大きな影響を受けてしまうなどの問題がある．そこで，ユークリッド距離に代わるものとしてタイムワーピング距離が使用されている [5], [6], [9], [10], [12]〜[14].

類似検索に要する計算コストは，次の 2 つの要因に大きく依存する．1 つ目は時系列の長さである．ユークリッド距離は時系列の長さに比例した CPU コストで済むが，タイムワーピング距離は長さの 2 乗に比例した CPU コストを要する．この問題に対処するために使用される手法の 1 つが次元縮小である．離散フーリエ変換（DFT）[2]，離散ウェーブレット変換（DWT）[3], Piecewise Aggregate Approximation（PAA）[5], [6], [9]，Adaptive Piecewise Constant Approximation（APCA）[8], [11] などが使用されている．2 つ目はデータベースに含まれる時系列の数である．時系列の数が増えることによって，CPU コストだけではなく I/O コストも大きく増加する．そこで，R-tree などのインデックスを使用する手法が提案されている [4], [8]〜[11].

しかしこれらの手法の中には，false dismissal を生じてしまうものがある．false dismissal とは，本来検索結果として返されるべき解が返されないことを意味する．この false dismissal を生じさせないで計算コストを小さくする手法として使用されているものが，下界関数を使用したフィルタリングである [9], [10], [13], [14].

本研究では，タイムワーピング距離を使用して類似検索を行う場合における効率的な検索手法を提案する．提案手法では，与えられた時系列に対してその時系列を包含する領域ボックス列というものを作成する．そして，この領域ボックス列を使用して下界関数を定義する．この下界関数をフィルタリングに使用することで，効率的な検索を可能にする．

本稿のこれからの構成は以下のようになっている．第 2. 章では，タイムワーピング距離の計算方法について述べる．第 3. 章では，下界関数をフィルタリングに使用する方法について述べる．また，関連研究で提案されている下界関数の概要と利点，欠点についても述べる．第 4. 章では，本研究で提案する領域ボックス列の作成方法と下界関数定義，それらを使用した検索方法について述べる．第 5. 章では，本研究で提案した手法と関連研究で提案されている手法の検索効率の比較実験を行った結果について述べる．第 6. 章では，まとめと今後の課題について述べる．

2. タイムワーピング

この章ではタイムワーピング距離の計算方法について簡単に述べる [1].

長さが N，M の 2 つの時系列 $Q = q_1, \cdots, q_N$，$C = c_1, \cdots, c_M$ に対して，2 つの点 q_i，c_j 間の距離 $d(q_i, c_j)$ を (i, j) 要素の値とする $N \times M$ 行列を作成する．距離 $d(q_i, c_j)$ は以下の式で表される．

$$d(q_i, c_j) = (q_i - c_j)^2 \tag{1}$$

次にワーピングパス $W = w_1, \cdots, w_k, \cdots, w_K$ を求める．ワーピングパスは以下の 3 つの条件を満たす行列の要素の順列で表

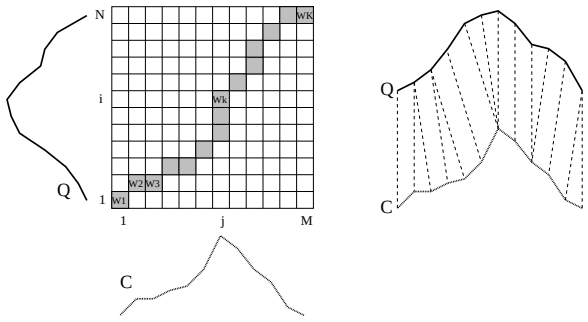

図 1　ワーピングパスの例
Fig. 1　An example of warping path.

現される．ワーピングパスの例を図 1 に示す.

- 境界条件：　$w_1 = (1, 1), w_K = (N, M)$ とする.
- 連続性：　$w_k = (a, b), w_{k-1} = (a', b')$ とすると，$a - a' \leqq 1$ かつ $b - b' \leqq 1$ となる.
- 単調性：　$w_k = (a, b), w_{k-1} = (a', b')$ とすると，$a - a' \geqq 0$ かつ $b - b' \geqq 0$ となる.

上記の 3 つの条件の下で得られるワーピングパスは非常に多く存在するが，それぞれのワーピングパスに対してワーピングコストと呼ばれるものが存在する．ワーピングコストとはワーピングパス上の要素の値の和の平方根である.

$$C(W) = \sqrt{\sum_{k=1}^{K} w_k} \tag{2}$$

この式から得られるワーピングコストの中で最小のものがタイムワーピング距離になる.

$$D_{tw}(Q, C) = \min\{C(W)\} \tag{3}$$

ワーピングパスは非常に多く存在するので，全てのワーピングパスに対してワーピングコストを計算して最小値を求めるのは現実的ではない．そこで，以下の再帰関数を使用してタイムワーピング距離を計算する.

$$\gamma(i, j) = d(q_i, c_j) + \min \begin{cases} \gamma(i-1, j-1) \\ \gamma(i-1, j) \\ \gamma(i, j-1) \end{cases} \tag{4}$$

$$\gamma(0, 0) = 0, \quad \gamma(i, 0) = \gamma(0, j) = \infty \tag{5}$$

この再帰関数を使用すると，式 (3) は以下のように表される.

$$D_{tw}(Q, C) = \sqrt{\gamma(N, M)} \tag{6}$$

これは動的プログラミングによって $O(NM)$ で計算可能である.

3. 下界関数

3.1 下界関数の概要

2 つの時系列 Q，C に対して本来の距離を返す関数を $D(Q, C)$ とすると，以下の式を満たす $D_{lb}(Q, C)$ が下界関数である.

$$D_{lb}(Q, C) \leqq D(Q, C) \tag{7}$$

この下界関数をフィルタリングに使用することで，効率的な

検索を行うことができる．例えば，問い合わせ Q との距離が ϵ 以下の時系列をデータベースから検索する場合を考える．データベースから取り出した時系列 C に対して，本来の距離を計算する前に下界距離を計算する．そして結果が $D_{lb}(Q,C) > \epsilon$ であったとする．すると，式 (7) から以下の式が成り立つ．

$$D(Q,C) \geqq D_{lb}(Q,C) > \epsilon \tag{8}$$

よって，$D(Q,C)$ を計算しなくとも Q と C の距離は ϵ より大きいことがわかる．$D_{lb}(Q,C) \leqq \epsilon$ の場合にのみ，$D(Q,C)$ を計算すればよい．

また，下界関数には性能の善し悪しがある．1 つは下界距離の計算コストである．最低でも本来の距離の計算コストよりも小さくなければ，下界関数をフィルタリングに使用することは意味を成さない．計算コストが小さくなるほどフィルタリングに要する時間が短くなるので，結果として検索時間も短くなる．もう 1 つは下界距離と本来の距離との差である．例えば，$D_{lb}(Q,C) = 0$ とすれば式 (7) は必ず成り立つ．しかし，ϵ がどんなに小さな正の値であっても $D_{lb}(Q,C) \leqq \epsilon$ となってしまうので，結局は $D(Q,C)$ を計算することになってしまう．$D_{lb}(Q,C)$ がより $D(Q,C)$ に近いほど $D(Q,C)$ の計算を避けることができるので検索時間が短くなる．

3.2 下界関数の関連研究

この章ではタイムワーピング距離の下界関数を使用して類似検索を行う手法を提案している関連研究について簡単に述べる．

Yi らは [14] において以下のような下界関数を定義している．第 2. 章で述べたように，タイムワーピング距離の計算では一方の時系列の点は他方の時系列のいずれかの点と必ず対応させなくてはならない．このことを利用して，一方の時系列の点の中で他方の時系列の最大の点より大きいもの，もしくは最小の点より小さいものを見つけて，それらの点の値の差を用いて下界関数を定義している．図で表すと，図 2 の点線で表される距離の和が下界距離になる．この手法は CPU コストが $O(N+M)$ で済むが，データベースから全ての時系列を取り出す必要があるので I/O コストは減少しない．

Kim らも [10] において以下のような下界関数を定義している．この手法ではそれぞれの時系列から 4 次元の特徴ベクトルを抽出する．特徴ベクトルは時系列の最初の値 $First(C)$，最後の値 $Last(C)$，最大値 $Max(C)$，最小値 $Min(C)$ から成る．そして 2 つの特徴ベクトルが与えられたとき，それぞれの要素の値の差の中で最大のものが下界距離になる．図 3 では最後の値の差 $Last(Q) - Last(C)$ が下界距離となる．この特徴ベクトルを 4 次元空間上に配置して R-tree などのインデックスを構築することで，計算コストを減少させている．この手法はデータベースが大きい場合には有効であるが，データベースがそれほど大きくない場合には性能が悪くなる．

[9], [13] では時間のシフト量に制限を設けた場合のタイムワーピング距離の下界関数を定義している．そのため，制限を設けない場合のタイムワーピング距離に対しての下界距離にはならない．

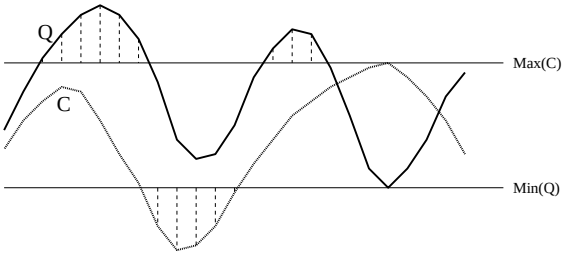

図 2　Yi による下界関数
Fig. 2　Lower bound function defined by Yi.

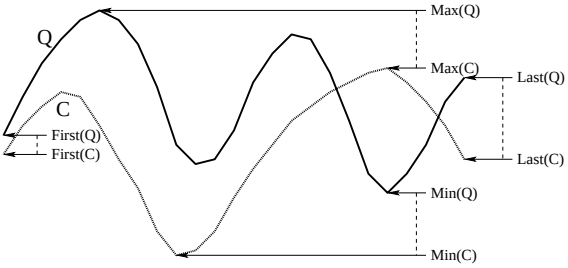

図 3　Kim による下界関数
Fig. 3　Lower bound function defined by Kim.

4. 提 案 手 法

4.1 領域ボックス列の作成

ある時系列 $Q = q_1, \cdots, q_N$ が与えられたときに，その時系列を包含する領域ボックス列 $R^Q = r_1^Q, \cdots, r_{N'}^Q$ を作成する方法を述べる．

まずは，時系列 Q を重なりのない複数の部分時系列 $\langle q_1, \cdots, q_h \rangle, \langle q_{h+1}, \cdots, q_k \rangle, \cdots, \langle q_{l+1}, \cdots, q_N \rangle$ に分割する．そして，それぞれの部分時系列に対して 1 つの領域ボックスを作成する．1 つの領域ボックスは上限 U，下限 L，サイズ S の 3 つの要素から成る．部分時系列 $\langle q_{h+1}, \cdots, q_k \rangle$ に対して作成された領域ボックスを r_i^Q とすると，3 つの要素は以下のように定義される．

$$r_i^Q.U = \max\{q_{h+1}, \cdots, q_k\} \tag{9}$$

$$r_i^Q.L = \min\{q_{h+1}, \cdots, q_k\} \tag{10}$$

$$r_i^Q.S = k - h \tag{11}$$

領域ボックス列の作成において中心となる処理は部分時系列への分割である．そして，この分割方法が検索効率に大きな影響を与える．影響を与える 1 つの要因は領域ボックスの数 N' であり，もう 1 つは領域ボックスの面積 $\sum_{i=1}^{N'}(r_i^Q.U - r_i^Q.L) \cdot r_i^Q.S$ である．下界距離の計算方法については次の章で詳しく説明するが，数が少なくなるほど下界距離の計算コストは小さくなり，面積が小さくなるほど下界距離によるフィルタリング効果は上がる．しかし数と面積はトレードオフの関係にあり，数を少なくすると面積が大きく，面積を小さくすると数が多くなってしまう．さらに，時系列データは時間の経過と共に絶えず生成されるので，リアルタイムで領域ボックスを作成できることも重要である [7]．そこで，本研究では表 1 のアルゴリズムを用いて領域ボックス列を生成する．このアルゴリズムに現れる τ は

表 1　領域ボックス列作成アルゴリズム
Table 1　Region box series creation algorithm.

1. $i=1, n=1$
2. find minimum $j(i \leqq j < N)$ such that
   $\max\{q_i,\cdots,q_j\} - \min\{q_i,\cdots,q_j\} \leqq \tau$
   $\max\{q_i,\cdots,q_{j+1}\} - \min\{q_i,\cdots,q_{j+1}\} > \tau$
3. **if** j dose not exist
4. 　　$r_n^Q.U = \max\{q_i,\cdots,q_N\}$
5. 　　$r_n^Q.L = \min\{q_i,\cdots,q_N\}$
6. 　　$r_n^Q.S = N - i + 1$
7. 　　**return** R^Q
8. $r_n^Q.U = \max\{q_i,\cdots,q_j\}$
9. $r_n^Q.L = \min\{q_i,\cdots,q_j\}$
10. $r_n^Q.S = j - i + 1$
11. $i = j+1, n = n+1$
12. go to 2.

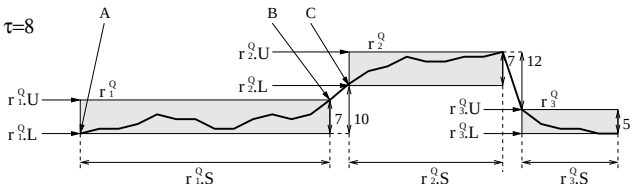

図 4　領域ボックス列の例
Fig. 4　An example of region box series.

閾値であり，これについては後で説明する．

このアルゴリズムに従って閾値 $\tau = 8$ として領域ボックス列を作成する様子を図 4 に示す．点 A から点 B までの部分時系列の最大値と最小値の差は 7 で τ 以下の値となっているが，点 B の次の点である点 C まで進むと差は 10 となって τ より大きくなる．よって，点 A から点 B までの部分時系列から領域ボックス r_1^Q を作成する．次に点 C を開始点として同様の処理を行う．これを時系列の最後まで繰り返すことで，領域ボックス列を作成する．この図では 3 つの領域ボックスから成る領域ボックス列 $R^Q = r_1^Q, r_2^Q, r_3^Q$ が作成されている．

ここで閾値 τ の指定方法について説明する．この値によって領域ボックスの数や面積は決まるが，時系列の取り得る値の範囲が事前にわかっていない限り，閾値 τ を直接指定することは困難である．そこで，許容振幅率 ρ というものを定義して，この ρ から閾値 τ を以下の式で計算する．

$$\tau = \rho \cdot (\max\{q_1,\cdots,q_N\} - \min\{q_1,\cdots,q_N\}) \tag{12}$$

これによって，時系列の取り得る値の範囲に関係なく，0 から 1 の値で指定することができる．ρ が小さいほど領域ボックスの面積は小さくなるが数は多くなる．ρ が大きいほど領域ボックスの数は少なくなるが面積は大きくなる．先程も述べた通り，面積と数はトレードオフの関係にあるので，より効率的な検索を行うためにはユーザが ρ に適切な値を指定する必要がある．

4.2　下界関数の定義

2 つの時系列 $Q = q_1,\cdots,q_N$，$C = c_1,\cdots,c_M$ に対して，それぞれの領域ボックス列 $R^Q = r_1^Q,\cdots,r_{N'}^Q$，$R^C = r_1^C,\cdots,r_{M'}^C$ が作成されたとする．この 2 つの領域ボックス列 R^Q，R^C

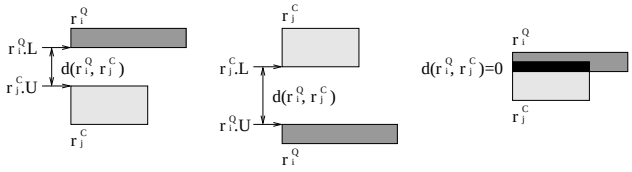

図 5　領域ボックス間の距離
Fig. 5　Distance between region boxes.

を使用して，タイムワーピング距離 $D_{tw}(Q,C)$ の下界関数 $D_{tw\_lb}(R^Q,R^C)$ を定義する．基本的には第 2. 章で述べたタイムワーピング距離の計算方法をもとに下界関数を定義する．

まずは式 (1) で表される 2 点間の距離の代わりとなる，2 つの領域ボックス r_i^Q，r_j^C 間の距離 $d'(r_i^Q, r_j^C)$ を定義する．図で表すと図 5 のようになる．

$$d'(r_i^Q, r_j^C) = \begin{cases} (r_i^Q.L - r_j^C.U)^2 & if\ r_i^Q.L > r_j^C.U \\ (r_j^C.L - r_i^Q.U)^2 & if\ r_j^C.L > r_i^Q.U \\ 0 & otherwise \end{cases} \tag{13}$$

次に再帰関数 (4)，(5) をもとに，下界関数のための新たな再帰関数を定義する．

$$\gamma'_r(i,j) = \min \begin{cases} \gamma'_d(i-1,j-1) + d'(r_i^Q, r_j^C) \cdot r_j^C.S \\ \gamma'_c(i-1,j) + d'(r_i^Q, r_j^C) \\ \gamma'_r(i,j-1) + d'(r_i^Q, r_j^C) \cdot r_j^C.S \end{cases} \tag{14}$$

$$\gamma'_c(i,j) = \min \begin{cases} \gamma'_d(i-1,j-1) + d'(r_i^Q, r_j^C) \cdot r_i^Q.S \\ \gamma'_c(i-1,j) + d'(r_i^Q, r_j^C) \cdot r_i^Q.S \\ \gamma'_r(i,j-1) + d'(r_i^Q, r_j^C) \end{cases} \tag{15}$$

$$\gamma'_d(i,j) = \max \begin{cases} \gamma'_r(i,j) \\ \gamma'_c(i,j) \end{cases} \tag{16}$$

$$\gamma'_d(0,0) = 0, \quad \gamma'_r(i,0) = \gamma'_c(0,j) = \infty \tag{17}$$

この再帰関数を使用して下界関数 $D_{tw\_lb}(R^Q,R^C)$ を以下のように定義する．

$$D_{tw\_lb}(R^Q,R^C) = \sqrt{\gamma'_d(N',M')} \tag{18}$$

これは動的プログラミングによって $O(N'M')$ で計算可能である．

このように定義された下界関数は以下の式を満たす．

$$D_{tw\_lb}(R^Q,R^C) \leqq D_{tw}(Q,C) \tag{19}$$

■証明■

証明するに当たって数学的帰納法を用いる．

2 つの領域ボックス r_i^Q，r_j^C が以下の部分時系列から作成されているものとする．

$$r_i^Q : \langle q_{s+1},\cdots,q_x,\cdots,q_t\rangle \quad s+1 \leqq x \leqq t \tag{20}$$
$$if\ i=1\ then\ s=0, \quad if\ i=N'\ then\ t=N$$

$$r_j^C : \langle c_{u+1},\cdots,c_y,\cdots,c_v\rangle \quad u+1 \leqq y \leqq v \tag{21}$$
$$if\ j=1\ then\ u=0, \quad if\ j=M'\ then\ v=M$$

そして既に以下のことが成り立っていると仮定する．

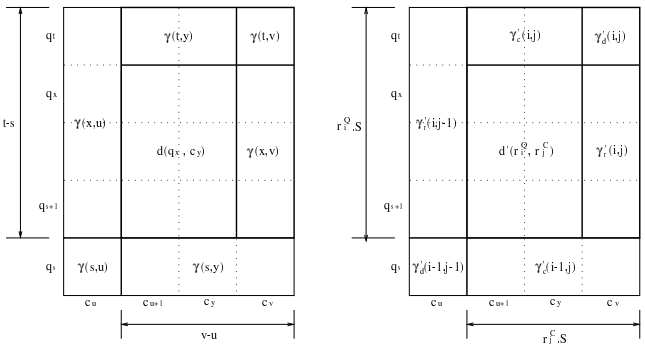

図 6　2 つの再帰関数の関係

Fig. 6　Relation between two recursive functions.

$$\gamma'_r(i,j-1) \leqq \gamma(x,u) \tag{22}$$

$$\gamma'_c(i-1,j) \leqq \gamma(s,y) \tag{23}$$

$$\gamma'_d(i-1,j-1) \leqq \gamma(s,u) \tag{24}$$

詳しい説明は省くが，式 (5) と (17) から $i=1$，$j=1$ のときは上記の式が実際に成り立つ．このときに以下のことが成り立つことを証明する．

$$\gamma'_r(i,j) \leqq \gamma(x,v) \tag{25}$$

$$\gamma'_c(i,j) \leqq \gamma(t,y) \tag{26}$$

$$\gamma'_d(i,j) \leqq \gamma(t,v) \tag{27}$$

2 点間の距離の定義式 (1) と 2 つの領域ボックス間の距離の定義式 (13) から以下の式が成り立つ．

$$d'(r_i^Q, r_j^C) \leqq (q_x - c_y)^2 = d(q_x, c_y) \tag{28}$$

式 (4) と図 6 を見てみると，$\gamma(x,v)$ は $\gamma(s,u)$，$\gamma(x,u)$，$\gamma(s,y)$ のいずれかに $d(q_x,c_y)$ を何回か加えることで求められることがわかる．

$\gamma(x,v)$ が $\gamma(s,u)$ から求められる場合には最低でも $v-u$ 回以上 $d(q_x,c_y)$ を加える必要がある．このことと式 (24)，(28) から以下の式が成り立つ．

$$\begin{aligned}\gamma(x,v) &\geqq \gamma(s,u) + d(q_x,c_y)\cdot(v-u)\\ &\geqq \gamma'_d(i-1,j-1) + d'(r_i^Q, r_j^C)\cdot r_j^C.S\end{aligned} \tag{29}$$

$\gamma(x,v)$ が $\gamma(s,y)$ から求められる場合には最低でも 1 回 $d(q_x,c_y)$ を加える必要がある．このことと式 (23)，(28) から以下の式が成り立つ．

$$\begin{aligned}\gamma(x,v) &\geqq \gamma(s,y) + d(q_x,c_y)\\ &\geqq \gamma'_c(i-1,j) + d'(r_i^Q, r_j^C)\end{aligned} \tag{30}$$

$\gamma(x,v)$ が $\gamma(x,u)$ から求められる場合には最低でも $v-u$ 回以上 $d(q_x,c_y)$ を加える必要がある．このことと式 (22)，(28) から以下の式が成り立つ．

$$\begin{aligned}\gamma(x,v) &\geqq \gamma(x,u) + d(q_x,c_y)\cdot(v-u)\\ &\geqq \gamma'_r(i,j-1) + d'(r_i^Q, r_j^C)\cdot r_j^C.S\end{aligned} \tag{31}$$

任意の x に対してこれら 3 つの式のいずれかは成り立つので，式 (14) から式 (25) が成り立つことが証明される．

$$\begin{aligned}\gamma'_r(i,j) &= \min\begin{cases}\gamma'_d(i-1,j-1) + d'(r_i^Q, r_j^C)\cdot r_j^C.S\\ \gamma'_c(i-1,j) + d'(r_i^Q, r_j^C)\\ \gamma'_r(i,j-1) + d'(r_i^Q, r_j^C)\cdot r_j^C.S\end{cases}\\ &\leqq \gamma(x,v)\end{aligned} \tag{32}$$

同様の方法で式 (26) が成り立つことも証明できる．

既に成り立つことが証明された式 (25)，(26) において $x=t$，$y=v$ とすると以下のようになる．

$$\gamma'_r(i,j) \leqq \gamma(t,v) \tag{33}$$

$$\gamma'_c(i,j) \leqq \gamma(t,v) \tag{34}$$

これらの式と式 (16) から式 (27) が成り立つことが証明される．

$$\gamma'_d(i,j) = \max\begin{cases}\gamma'_r(i,j)\\ \gamma'_c(i,j)\end{cases} \leqq \gamma(t,v) \tag{35}$$

最終的に $i=N'$，$j=M'$ とすると以下の式が成り立つ．

$$\gamma'_d(N',M') \leqq \gamma(N,M) \tag{36}$$

この式と式 (6)，(18) から，式 (19) が成り立つ．

$$\begin{aligned}D_{tw\_lb}(R^Q,R^C) &= \sqrt{\gamma'_d(N',M')}\\ &\leqq \sqrt{\gamma(N,M)} = D_{tw}(Q,C)\end{aligned} \tag{37}$$

4.3　検索アルゴリズム

類似検索には大きく分けると近傍検索と範囲検索の 2 種類がある．近傍検索とは，問い合わせ Q と数値 k が与えられたときに，Q に最も類似した k 個のデータを検索するものである．範囲検索とは，問い合わせ Q と閾値 ϵ が与えられたときに，Q から距離 ϵ 以内にある全てのデータを検索するものである．ここでは $k=1$ のときの最近傍検索アルゴリズムを説明するが，$k \neq 1$ の場合の近傍検索や範囲検索もアルゴリズムを少し変更することで行うことができる．

実際に検索を行う前に，データベース中の全ての時系列に対して領域ボックス列を作成してファイルに格納しておく．このファイルを圧縮ファイルと呼ぶことにする．そして問い合わせ

表 2　最近傍検索アルゴリズム

Table 2　1 nearest neighbor search algorithm.

1. make R^Q from Q
2. $mindist = \infty$
3. get a region box series R^C from compression file
4. **if** R^C dose not exist
5. 　　**return** $result$
6. calculate $D_{tw\_lb}(R^Q,R^C)$
7. **if** $D_{tw\_lb}(R^Q,R^C) < mindist$
8. 　　get a time series C from database
9. 　　calculate $D_{tw}(Q,C)$
10. 　　**if** $D_{tw}(Q,C) < mindist$
11. 　　　　$result = C$，$mindist = D_{tw}(Q,C)$
12. go to 3.

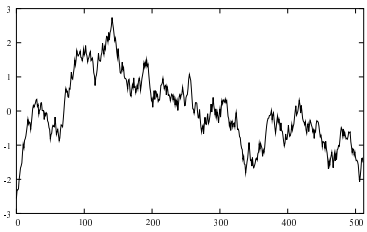

図 7　RandomWalk データ
Fig. 7　A RandomWalk data.

時系列 $Q = q_1, \cdots, q_N$ が与えられると，表 2 のアルゴリズムを用いて最近傍検索を行う．

まずは，Q から領域ボックス列 R^Q を作成する．次に，圧縮ファイルから領域ボックス列 R^C を 1 つ取り出して下界距離 $D_{tw\_lb}(R^Q, R^C)$ を計算する．下界距離がこれまでの最小距離より小さい場合には，データベースから時系列を取り出して本来の距離 $D_{tw}(Q, C)$ を計算する．これも最小距離より小さい場合には最小距離を更新する．そして，圧縮ファイルから次の領域ボックス列を取り出して同様の処理を行う．これを全ての時系列に対して行って，最終的に最小距離となった時系列を検索結果として返す．

5. 実　　験

5.1 データセット

今回の実験では合成データセットと株価データセットの 2 種類を使用する．

- 合成データセット：　以下の式で生成される RandomWalk データ（図 7）を使用してデータセットを作成した．

$$v_t = v_{t-1} + r_t \quad （r_t は -1〜1 までの乱数） \qquad (38)$$

長さが 256，512，1024 の RandomWalk データをそれぞれ 100，1000，10000 個生成してデータセットを作成することで，合計 9 つの合成データセットを作成した．

- 株価データセット：　S&P 500 (http://biz.swcp.com/stocks/) から 479 個の銘柄の始値，高値，安値，終値の 4 種類のデータを 252 日分入手してデータセットを作成した．データ長が 252，データ数が 479 × 4 のデータセットになる．

検索を行うときの問い合わせデータには，データセット中のデータと同じ長さの RandomWalk データを新たに生成して使用する．

また，それぞれのデータに対して平均値が 0，標準偏差が 1 になるように正規化を行う．時系列の類似検索を行うときには，値が近いものではなく値の変化の仕方が近いものを検索結果として欲しいことが多いと考えられる．正規化を行うことによって，値の大小に関係なく変化の仕方が近いものを検索することが可能になる．

5.2 評　価　値

検索効率を評価するための値として以下の 3 つを使用する．

- フィルタリング率：　データベース中の全てのデータの中で，下界関数によるフィルタリングで解候補から取り除かれて，実際にタイムワーピング距離を計算する必要の無かったデータの割合．この値が大きいほどタイムワーピング距離の計算に要する CPU コストが小さくなるので，検索効率が良いと判断できる．
- ディスクアクセス数：　データベースやインデックスファイル，圧縮ファイルからデータを取り出すためにディスクにアクセスした回数．ページサイズを 1024 バイトとしてページ単位でディスクアクセスを行う．この値が小さいほどデータの取り出しに要する I/O コストが小さくなるので，検索効率が良いと判断できる．
- 実行時間：　問い合わせを与えてから検索結果が得られるまでの時間．検索効率を総合的に評価することができる．

50 個の異なった問い合わせデータで最近傍検索を行って平均結果を計測する．

5.3 比 較 手 法

以下の 4 つの手法の検索効率を比較する．

- 逐次検索手法：　下界関数でのフィルタリングを行わずに，直接タイムワーピング距離を計算する．
- Yi 検索手法：　第 3.2 章で述べた Yi らの提案した下界関数をフィルタリングに使用する．
- Kim 検索手法：　第 3.2 章で述べた Kim らの提案した下界関数をフィルタリングに使用する．インデックスには R-tree を使用する．ノードサイズはページサイズと同じ 1024 バイトとして，1 回のディスクアクセスでノードを取り出せるようにする．
- RB 検索手法：　本研究で提案した領域ボックスを使用した下界関数をフィルタリングに使用する．領域ボックス列を作成するときに指定する許容振幅率には，0.01〜0.20 の 0.01 刻みの値を使用する．

5.4 実 験 結 果

5.4.1 フィルタリング率

Yi 検索手法と Kim 検索手法でのフィルタリング率を表 3 に，RB 検索手法で許容振幅率を変えたときのフィルタリング率を図 8 に示す．

どの手法でもデータ数が多くなるとフィルタリング率が大きくなっている．これは，検索が進むほど問い合わせデータと最近傍データのタイムワーピング距離が小さくなっていくので，下界関数によるフィルタリング効果が上がるためと考えられる．

Yi 検索手法と Kim 検索手法を比較すると，データ長が短いときは Kim 検索手法の方が，長いときは Yi 検索手法の方がフィルタリング率が大きくなっている．これは，Kim 検索手法では次元縮小を行って 4 次元空間上で下界関数を定義しているため，データ長が長くなって高次元になると情報量の損失が大きくなるためと考えられる．

RB 検索手法では許容振幅率を大きくするとフィルタリング率が小さくなっている．これは，許容振幅率を大きくすると領域ボックスの面積が大きくなるので，下界距離が小さくなってフィルタリング効果が下がるためと考えられる．しかし，許容振幅率を 0.2 まで大きくしてもフィルタリング率は 40〜60%であり，Yi 検索手法や Kim 検索手法の 10〜40%に比べて高いことがわかる．

表 3　Yi 検索手法と Kim 検索手法のフィルタリング率
Table 3　Filtering rate with Yi search method and Kim search method.

dataset	synthetic dataset									stock dataset
length of data	256			512			1024			252
number of data	100	1000	10000	100	1000	10000	100	1000	10000	1916
Yi search method	6.74	13.85	17.66	10.78	18.94	24.99	9.34	19.12	23.92	10.59
Kim search method	21.44	30.81	36.37	12.30	20.13	26.46	2.72	10.14	15.93	24.42

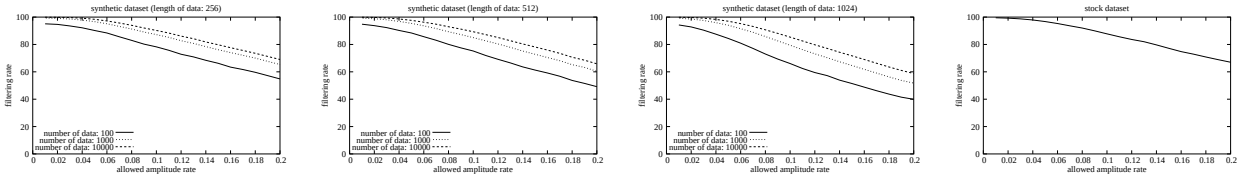

図 8　RB 検索手法のフィルタリング率
Fig. 8　Filtering rate with RB search method.

表 4　逐次，Yi，Kim，RB 検索手法のディスクアクセス数
Table 4　Disk access with serial, Yi, Kim, and RB search method.

dataset	synthetic dataset									stock dataset
length of data	256			512			1024			252
number of data	100	1000	10000	100	1000	10000	100	1000	10000	1916
serial search method	201	2004	20040	401	4004	40040	801	8004	80040	3780
Yi search method	201	2004	20040	401	4004	40040	801	8004	80040	3780
Kim search method	241.58	2127.62	19581.26	444.50	4050.30	37285.86	881.52	8144.50	76232.52	4399.60
RB search method	154.24	1250.54	11504.76	236.80	1860.60	16313.20	417.82	3119.40	26402.52	2102.08
(allowed amplitude rate)	(0.13)	(0.13)	(0.13)	(0.10)	(0.12)	(0.12)	(0.08)	(0.09)	(0.09)	(0.13)

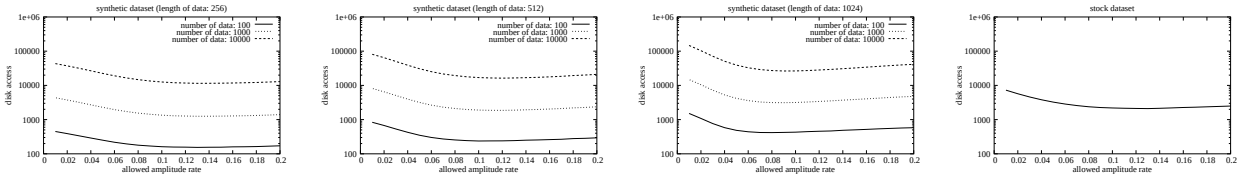

図 9　RB 検索手法のディスクアクセス数
Fig. 9　Disk access with RB search method.

5.4.2　ディスクアクセス数

逐次検索手法，Yi 検索手法，Kim 検索手法でのディスクアクセス数を表 4 に，RB 検索手法で許容振幅率を変えたときのディスクアクセス数を図 9 に示す．また表 4 には，RB 検索手法で許容振幅率を変えて最もディスクアクセス数が少ないときの結果も示す．

Yi 検索手法では下界距離を計算するためにデータベース中のデータが必要なので，逐次検索手法と同じディスクアクセス数になる．

Kim 検索手法では下界距離を計算するためにインデックスファイルにアクセスして，タイムワーピング距離を計算する必要がある場合にのみデータベースにアクセスするので，フィルタリング効果が十分に高ければディスクアクセス数は減ると考えられる．データ数が多いときはフィルタリング率が高いのでディスクアクセス数は減少しているが，データ数が少ないときはフィルタリング率が低いためにむしろ増加している．

RB 検索手法では，許容振幅率を大きくしていくとディスクアクセス数は減少していくが，ある箇所から増加していくようになる．許容振幅率を大きくしていくと，領域ボックスの数が減るので圧縮ファイルへのアクセス数は減少するが，フィルタリング率が小さくなっていくためにデータベースへのアクセス数が増加する．よって，圧縮ファイルへのアクセスとデータベースへのアクセスのバランスが取れた箇所でディスクアクセス数が最小になる．また，データの数が多く長さが長いときほど，逐次検索手法を基準にしたディスクアクセス数の減少率が高くなっている．

5.4.3　実 行 時 間

逐次検索手法，Yi 検索手法，Kim 検索手法での実行時間を表 5 に，RB 検索手法で許容振幅率を変えたときの実行時間を図 10 に示す．また表 5 には，RB 検索手法で許容振幅率を変えて最も実行時間が短いときの結果も示す．

Yi 検索手法と Kim 検索手法を比較すると，表 4 に示した通りディスクアクセス数には大きな差がないので，表 3 に示したフィルタリング率が実行時間に大きく影響している．フィルタ

表 5　逐次，Yi，Kim，RB 検索手法の実行時間

Table 5　Execution time with serial, Yi, Kim, and RB search method.

dataset	synthetic dataset									stock dataset
length of data	256			512			1024			252
number of data	100	1000	10000	100	1000	10000	100	1000	10000	1916
serial search method	0.34	3.55	35.62	1.33	13.61	136.41	5.20	54.35	543.85	6.30
Yi search method	0.32	2.98	29.59	1.20	11.55	105.32	5.11	43.83	419.92	5.65
Kim search method	0.27	2.39	22.91	1.18	11.40	102.67	5.14	48.77	459.88	4.82
RB search method	0.10	0.75	6.26	0.33	2.21	17.31	1.26	7.04	49.31	1.16
(allowed amplitude rate)	(0.10)	(0.09)	(0.10)	(0.07)	(0.08)	(0.09)	(0.06)	(0.06)	(0.06)	(0.09)

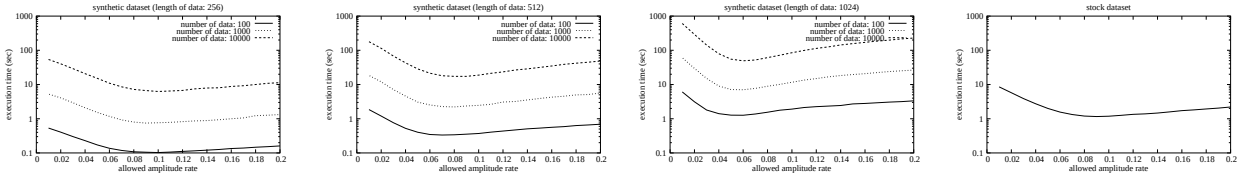

図 10　RB 検索手法の実行時間

Fig. 10　Execution time with RB search method.

リング率の高い方が実行時間が短くなっている．RB 検索手法では，許容振幅率を大きくしていくと実行時間は減少していくが，ある箇所から増加していくようになる．許容振幅率を大きくしていくと，領域ボックスの数が減るので下界距離の計算コストは減少するが，フィルタリング率が小さくなって行くためにタイムワーピング距離の計算コストが増加する．よって，下界距離の計算コストとタイムワーピング距離の計算コストのバランスが取れた箇所で実行時間が最小になる．データの数や長さにもよるが，Yi 検索手法や Kim 検索手法では最大でも逐次検索手法の 65%程度までしか実行時間が短くならないが，RB 検索手法では 30%〜10%程度まで実行時間が短くなっている．

6. おわりに

本研究では，時系列から領域ボックス列を作成して，この領域ボックス列を使用して計算される下界関数を新しく定義した．そしてこの下界関数を使用して，false dismissal を生じることなく効率的な類似検索を行うことを試みた．合成データセットと株価データセットを使用した最近傍検索の実験によって，今回提案した手法が関連研究で提案されている手法よりも優れた性能を持っていることが実証された．

今後の課題としては，今回は時系列を領域ボックス列に次元縮小して圧縮ファイルを作成しただけなので，R-tree などのインデックスを構築して類似検索に利用する手法の提案ができればよいと考えている．インデックスを利用することで下界距離の計算回数を減らすことができ，より効率的な検索が可能になると考えられる．

文　献

[1] J. B. Kruskall and M. 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.

[2] R. Agrawal, C. Faloutsos and A. Swami, “Efficient Similarity Search In Sequence Databases,” In Proc. of 4th International Conference on Foundations of Data Organization and Algorithms, 1993

[3] F. K.-P. Chan, A. W.-C. Fu, and C. Yu, “Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping,” In Proc. of IEEE Transactions on Knowledge and Data Engineering, Vol.15, No.3, 2003.

[4] K. K. W. Chu and M. H. Wong, “Fast Time-Series Searching with Scaling and Shifting,” In Proc. of 18th ACM Symposium on Principles of Database Systems, 1999.

[5] S. Chu, E. Keogh, D. Hart, and M. Pazzani, “Iterative Deepening Dynamic Time Warping for Time Series,” In Proc. of 2 na SIAM International Conference on Data Mining, 2002.

[6] E. Keogh and M. Pazzani, “Scaling up Dynamic Time Warping for Datamining Applications,” In Proc. of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

[7] E. Keogh, S. Chu, D. Hart, and M. Pazzani, “An Online Algorithm for Segmenting Time Series,” In Proc. of 2001 IEEE ICDM, 2001.

[8] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, “Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases,” In Proc. of 2001 ACM SIGMOD, 2001.

[9] E. Keogh, “Exact Indexing of Dynamic Time Warping,” In Proc, of 28th International Conference on VLDB, 2002.

[10] S.-W. Kim, S. Park, and W. W. Chu, “An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases,” In Proc. of 17th ICDE, 2001.

[11] Q. Li, I. F. V. Lopez, and B. Moon, “Skyline Index for Time Series Data,” In Proc. of IEEE Transactions on Knowledge and Data Engineering, Vol.16, No.6, 2004.

[12] S. Park, W. W. Chu, J. Yoon, and C. Hsu, “Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases,” In Proc. of 16th ICDE, 2000.

[13] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, “Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures,” In Proc. of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[14] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, “Efficient Retrieval of Similar Time Sequences under Time Warping,” In Proc. of 14th ICDE, 1998.