G-18

変換を考慮した時系列データの高速類似検索

Fast Similarity Search of Time Series Data Considering Transformations

岩下　俊一郎† 宝珍　輝尚† 野宮　浩揮†
Toshiichiro Iwashita Teruhisa Hochin Hiroki Nomiya

1　はじめに

計算機技術の急速な発展に伴い，画像，音楽，動画など様々なマルチメディアデータを扱うことができるようになってきている．しかし，膨大な量のデータから適切なデータを正確かつ迅速に求めることはまだまだ困難である．このようなデータの中には，株価の変動，気温の変化，科学実験の測定データ等，時系列の形式をとるものがある．

ここで，ある時系列データと類似した時系列データを求めるという問題について，様々な研究がなされている [1–6]．単純な方法としては，時系列データをユークリッド空間の 1 点とみなし，その点間のユークリッド距離を類似度として検索を行うことが考えられる．しかし，時系列データをユークリッド空間の 1 点とみなすと，その空間の次元は非常に高いものになってしまう．多次元空間のオブジェクトを高速に求めるための多次元インデックス構造は，10 次元程度が実用の限界であることが実験的に明らかになっており [3]，次元を削減させるための方法が研究されている．一つの方法は，時系列データを時間領域から周波数領域へと変換し，その系列の特徴をよく表す低周波数の数個の係数を利用する方法である [1–3].

ここで，時系列にある種の変換を施した結果と類似したものを求めたいという要求がある．例えば，動きを平滑化するために株価の 10 日間の移動平均を取ったものと類似しているものを求める，といった問い合わせである．Rafiei らはこのような問い合わせを効率的に処理する手法を提案している [1]．この手法は，時系列を多次元空間中の 1 点で管理する多次元インデックスにおいて，その管理単位である最小外接矩形 (MBR) に変換を施すことで上記の問い合わせを処理するという方法である．

この手法は非常に興味深い方法であるが，クラスタリングの手法が論文には明記されていない等，実装においてはいくつかの問題がある．

そこで本稿では，Rafiei らの手法を実装し，実装上の不明瞭な点を明確化する．さらに，実機上で性能評価を行い，Rafiei らの手法の良好性を検証する．

以下，2 節では Rafiei らの手法について述べる．3 節では 2 節で述べた手法の実装について述べる．4 節では実際に実機上で性能評価を行うための実験方法，および，実験結果について述べる．5 節で考察を行い，6 節でまとめとする．

† 京都工芸繊維大学，Kyoto Institute of Technology

2　先行研究 [1]

2.1　変換

世の中に多く存在する時系列データに対し，類似した時系列を見つけたいという要求が多く存在する [4, 5]．例えば，同じような変化をする株価や，去年と似たような売れ行きの製品，世界の 2 つの地域において気温変化のパターンが似ていた年，を見つけたいといった要求が挙げられる．そういった要求を満たすために，時系列間の類似度を考える．

2 つの時系列間で考えられる類似度を決定するための簡単な方法として，2 系列間のユークリッド距離を計算する方法があり，その距離がユーザが定義する閾値よりも低い場合に 2 つの時系列は似ているとする．しかし，単純にユークリッド距離を計算するだけでは，ユーザが求めたい類似系列を得られない場合がある．例として，次の 2 つの系列を考える．

$$\vec{s_1} = [36, 38, 40, 38, 42, 38, 36, 36, 37, 38, 39, 38, 40, 38, 37]$$
$$\vec{s_2} = [40, 37, 37, 42, 41, 35, 40, 35, 34, 42, 38, 35, 45, 36, 34]$$

この 2 系列間のユークリッド距離は $D(\vec{s_1}, \vec{s_2}) = 11.92$ であり，あまり類似しているとはいえない．しかし，この 2 系列の 3 日間の移動平均を考えると，それらの間のユークリッド距離は 0.47 となり，極めて類似しているとみられる．このことから，系列に変換を施すことを考える．

移動平均とは株価のデータ解析において，短期間の価格変動の引き伸ばしや，潜在的な動向を示すため等に広く用いられる．株価における m 日間の移動平均とは，ある日付 M における直近の m 日間の終値の平均である．それらの終値を $p_M, p_{M-1}, \ldots, p_{M-m}$ としたとき，ある日付 M における m 日間の移動平均 S_M は次式により計算される．

$$S_M = \frac{p_M + p_{M-1} + \cdots + p_{M-m}}{m} \tag{1}$$

なお，系列の長さが n であるとき，$p_{-m} = p_{n-m+1}$ とする．すなわち，系列の初めに達したとき終端へ循環させるものとする．

2.2　変換の適用

時系列には，それぞれをそのまま比較する前に，取り除いたり減少させたりする必要のある相違点が存在する場合がある．これらの相違点を取り除く 1 つの方法として，それらに何らかの変換を適用させることが挙げられる．考える変換は，時系列に対して広い場面で有用な表現が可能で，更にその変換の集合を用いて表現される問い合わせは効率的に処理される，といったものが望ましい．そこで，ここではスケーリングとトランスレーションという 2 つの変換を用いる．具体的には，n 次元空間での変換 t を，スケーリングを表す $\vec{a}$ とトランスレーション

を表す $\vec{b}$ によるベクトル組 $(\vec{a}, \vec{b})$ によって表されるものと定義する．長さ n の系列 $\vec{x}$ に変換 $t = (\vec{a}, \vec{b})$ を適用することを $t(\vec{x})$ と表し，次式により変換適用の計算を行う．

$$t(\vec{x}) = Conv(\vec{a}, \vec{x}) + \vec{b} \tag{2}$$

ここで，$Conv(\vec{x}, \vec{y})$ は $\vec{x}$ と $\vec{y}$ のたたみ込み積を表し，長さ n の系列に対し，要素 j について次式により計算される．

$$Conv(\vec{x}, \vec{y})_j = \sum_{k=0}^{n-1} x_k y_{j-k} \quad j = 0, 1, \cdots, n-1 \tag{3}$$

式 (2) が時間領域での計算であるのに対し，周波数領域での計算を考える．$\vec{A}, \vec{B}, \vec{X}$ をそれぞれ $\vec{a}, \vec{b}, \vec{x}$ の DFT 後の周波数表現であるとすると，$t(\vec{X})$ は周波数領域で次のように計算される．

$$t(\vec{X}) = \vec{A} * \vec{X} + \vec{B} \tag{4}$$

ここで演算記号 '$*$' はベクトルの要素同士の乗算を意味する．式 (4) は式 (2) に比べて計算コストが低くなっている．

一般に，長さ n の系列の m 日間の移動平均は $t_{mavg} = (\vec{a}, \vec{b})$ によって表される．ここで，

$$\vec{a} = [\underbrace{\underbrace{\frac{1}{m}, \frac{1}{m}, \cdots, \frac{1}{m}}_{m}, 0, 0, \cdots, 0}_{n}] \tag{5}$$

であり，$\vec{b} = \vec{0}$ である．$\vec{0}$ はサイズ n のゼロベクトルである．

2.3 インデックスの構築

系列の索引付けに用いることのできる多次元インデックスには R 木やグリッドファイルなどがある．これらのインデックスはすべて次元の呪いの影響を受け，高次元での検索性能を悪化させる．この問題を避ける方法としては，ある系列を他のものとおおよそ区別できるような重要な特徴を数個選び，インデックスの中でそれらの特徴のみを保つといったものがある．この近似法では系列が比較されるときに数点の差異が生じるが，その差異は後処理によって容易になくすことができる．

特徴を選び出すために，ここでは離散フーリエ変換 (DFT) を用いて系列を時間領域から周波数領域に変換し，それぞれの系列に対して初めの数個だけの DFT 係数を保持する．DFT を採用する主な理由は以下の通りである．

- DFT を適用すると，多くの時系列に対して数個の低周波数係数に系列の特徴が集約されるため．その数個の低周波数係数は，インデックスのキーとして利用することができる．
- DFT は正規直交変換であるため，時間領域と周波数領域とでユークリッド距離が変化しないため．

ある時系列データが与えられたとき，インデックスは次のように構築される．

1. 各系列に DFT を適用し，系列の特徴を表す初めの数個の DFT 係数をそれぞれ実部と虚部に分けて保持する．
2. はじめの k 個の DFT 係数のみを保持する場合，1 つの系列は $2k$ 次元空間における 1 点となる．
3. 複数の系列を表す複数の点を R 木等の多次元インデックスで組織化する．

ここで扱う R 木は，次のようなものである．

- 2 次元以上においての B 木の自然な拡張として見られる．
- 葉でないノードは (MBR, ptr) の形のエントリを含む．
  - MBR は子孫ノードの中のすべてのエントリの最小外接矩形
  - ptr は子孫ノードへのポインタ
- 同じレベルでの矩形はオーバーラップし得る．
- 点データに対する葉ノードは点データの集合を含む．

2.4 単一の変換による問い合わせ

ここでは，インデックス中の系列にある 1 つの変換を適用し，問い合わせを行う手法について述べる．例として，次の近接問い合わせを考える．

クエリ 1　問い合わせ系列 $\vec{q}$，変換 t，閾値 ϵ が与えられたとき，データ集合中でユークリッド距離 $D(t(\vec{x}), \vec{q}) < \epsilon$ を満たすすべての点 $\vec{x}$ を見つける．

この問い合わせに対し，次のアルゴリズムにより変換を適用しつつ検索が可能となる．

アルゴリズム 1　R 木インデックスが系列の初めの k 個の DFT 係数において構築されているものとし，根ノードは N で表されるものとする．

1. 前処理
   a. 変換 t と $\vec{q}$ を DFT により時間領域から周波数領域へ変換する．t，$\vec{q}$ の初めの k 個の DFT 係数をそれぞれ t_k，$\vec{q_k}$ と表す．
   b. $\vec{q_k}$ に対する検索矩形 q_{rect} を構築する．検索矩形とは，$\vec{q}$ とのユークリッド距離が ϵ 以下であるすべての点を含む最小外接矩形である．DFT 係数を実部と虚部に分解，すなわち直交座標系で扱う場合，検索矩形は各次元 $i = 1, \cdots, 2k$ に対して $(q_i - \epsilon, q_i + \epsilon)$ として構築される．
2. 検索
   a. N が葉でない場合，N のすべての矩形エントリに変換 t を適用し，適用後の矩形が q_{rect} にオーバーラップするかをチェックする．すべてのオーバーラップするエントリに対し，根ノードがオーバーラップするエントリに指されているインデックス上で手順 2 を呼び出す．
   b. N が葉である場合，N のすべての点エントリに変換 t を適用し，適用後の矩形が q_{rect} にオーバーラップするかをチェックする．もしするなら，そのエントリを解候補に加える．
3. 後処理
   すべての解候補 $\vec{x}$ に対し，$D(t(\vec{x}), \vec{q}) < \epsilon$ かどうかを全データベースをチェックする．真であればそのエントリをアンサーとする．

このアルゴリズムはあらゆる系列に対して正常に動作することが保証されている．

2.5 多数の変換による問い合わせ

データセット S 上での R 木インデックスが与えられたとき，次の近接問い合わせを考える．

クエリ 2 問い合わせ系列 $\vec{q}$ および変換の集合 T が与えられたとき，ユークリッド距離が $D(t(\vec{s}), \vec{q}) < \epsilon$ であるようなすべての時系列 $\vec{s} \in S$ と変換 $t \in T$」を見つける．

こういった問い合わせの例としては，T を $m \in \{1 \ldots 40\}$ に対する m 日間の移動平均の集合であるとし，ある株価の m 日間の移動平均に似ているすべての株を見つけたいとする，といったものが挙げられる．

この問い合わせを処理するための 1 つの方法として，すべての変換 $t \in T$ の 1 つ 1 つに対して，アルゴリズム 1 を用いる方法が考えられる．この方法では，検索結果の和集合が問い合わせのアンサーとなる．ここではこのアルゴリズムを **ST インデックス**と呼ぶ．ここで，ST は"a Single Transformation at a time"を意味する．しかし，ST インデックスのコストは，インデックスを走査するコストに $|T|$ を掛けたものになり，あまり効率が良いとはいえない．

そこで，変換をまとめてグループ化を行い，その変換のグループごとに検索を行う手法を考える．ここではその手法を **MT インデックス**と呼ぶ．ここで，MT は"a Multiple Transformation at a time"を意味する．以下では，変換のグループ化，変換の適用について述べる．

まず，変換のグループ化について，すべての変換がまとめて 1 つのグループとなる場合を考える．この場合，1 回の変換の適用のみで問い合わせを処理できる．しかし，変換の MBR が大きな範囲を覆う場合，多くの誤った結果を解候補としてしまう．一方，数個のグループとなる場合，それぞれの MBR の領域は小さくなり，誤った結果の個数は減少するが，インデックスが数回走査される必要がある．また，グループの数が変換の数と同じになる場合，すなわち 1 つの変換が 1 つのグループになる場合，それは ST インデックスと同じパフォーマンスとなる．以上のことを踏まえた上で，変換のグループ化を行う．

次に変換の適用について述べる．変換のグループが与えられたとき，そのグループの中でのすべての変換に対して 1 つの最小外接矩形 (MBR) を構成することができる．ここではその矩形を変換矩形と呼ぶ．データ矩形に変換矩形を適用するために，$2n$ 次元の変換矩形を *mult-MBR* と *add-MBR* という 2 つの n 次元 MBR に分解する．ここで，mult-MBR は $\vec{a}$ の各点に対する MBR，add-MBR は $\vec{b}$ の各点に対する MBR である．mult-MBR: $<(M_1 l, M_1 h), \ldots>$，add-MBR: $<(A_1 l, A_1 h), \ldots>$，データ矩形 X: $<(X_1 l, X_1 h), \ldots>$が与えられたとき，矩形 X に mult-MBR と add-MBR を適用した結果は，矩形 Y: $<(Y_1 l, Y_1 h), \ldots>$となる．ここで，すべての次元 i に対し，

$$\begin{aligned} Y_i l &= A_i l + \min(M_i l * X_i l, M_i l * X_i h, M_i h * X_i l, M_i h * X_i h) \\ Y_i h &= A_i h + \max(M_i l * X_i l, M_i l * X_i h, M_i h * X_i l, M_i h * X_i h) \end{aligned} \quad (6)$$

である．

ここで，クエリ 2 を処理する MT インデックスによる検索アルゴリズムを次に示す．

アルゴリズム 2 時系列の初めの k 個のフーリエ係数上で構築された R 木インデックス，その根 N，変換の集合 T，閾値 ϵ，検索系列 $\vec{q}$ が与えられたとき，T の要素による変換後に $\vec{q}$ との距離が ϵ になるすべての変換と系列を見つける．

1. クラスタリングアルゴリズムを用いて T を集合 $T_1, T_2, \ldots$ に分解する．
2. すべての集合 T_i に対し次の Step3~6 を行う．
3. T_i を mult-MBR および add-MBR に分解する．
4. N が葉でない場合，(6) 式を用いて N のすべてのエントリに mult-MBR および add-MBR を適用し，結果の矩形が q_{rect} に交差するかどうかをチェックする．すべての交差するエントリに対し，交差するエントリのノードを根とするインデックス上で Step4 を行う．
5. N が葉である場合，N のすべてのエントリに mult-MBR および add-MBR を適用し，結果の矩形が q_{rect} に交差するかどうかをチェックする．もし交差するなら，そのエントリは候補になる．
6. すべての候補であるエントリに対し，その全データベース中で T_i の中のすべての変換を系列に適用し，データ系列とクエリ系列間のユークリッド距離が ϵ 以下となる変換と系列を決定する．

このアルゴリズムはどのような系列に対しても正常に動作することが保証されている．

3 実装

3.1 インデックス

すべての時系列にはあらかじめ正規化を行っておくものとする．ここで正規化とは，系列の全体をその標準偏差で割り，さらにその平均を引くことを指す．あらかじめ正規化を行う理由は，系列の正規形間のユークリッド距離はそれらの相互相関に直接関係するという性質があるためである [1]．また，正規形の平均は 0 となるため，DFT を適用すると 1 番目の DFT 係数は 0 となり，保持する必要がなくなるという利点もある．ここで，系列ごとに初めの 1 ~ 3 個の DFT 係数を保持すれば，良いパフォーマンスを生むのに十分であることが実証されている [2]．これらを踏まえ，ここでは系列ごとに 2 番目と 3 番目の DFT 係数を保持するものとする．なお，変換は，$\vec{a}$ と $\vec{b}$ のそれぞれについて FFT を行い，正規化は行わず，後の計算のために 2 番目と 3 番目の DFT 係数を保持する．

高速化のために，DFT には高速フーリエ変換 (FFT) を用い，扱う系列の長さは 2 の累乗とする．FFT により得られる DFT 係数は複素数となるので，それぞれ実部と虚部に分けて管理する．すなわち，1 つの系列は，2 番目の DFT 係数の実部，2 番目の DFT 係数の虚部，3 番目の DFT 係数の実部，3 番目の DFT 係数の虚部という 4 つの実数値を持つ，4 次元の 1 点として表される．

それらの点を索引付けするのに，R*木インデックスを用いる．ここでは，R*木の実装のために汎用検索木 (GiST)[7] を用

いる．GiST は B+ 木を一般化したもので，R 木などの良く利用される多次元インデックスの実装に利用できる．また，新たなインデックスを開発することも容易である．

以降では，変換の適用方法の統一のため，1 つの系列を点データとして索引付けする代わりに，最小値と最大値を同じ値とした矩形データとして索引付けを行うものとする．

3.2 変換矩形の適用

MT インデックスでの変換の適用において，式 (6) 中の各値は複素数であるため，min(...) および max(...) 部分の計算方法が明瞭でない．そこで，式 (6) の代わりに以下の方法により矩形に変換を適用する．

ここで，簡単化のために 2 番目 DFT 係数についての次元のみを考える．データ矩形 $\vec{X}$ の 2 番目の DFT 係数の実部，虚部の最小値，最大値をそれぞれ $X_{real2l}, X_{real2h}, X_{imag2l}, X_{imag2h}$ と表す．また，mult-MBR と add-MBR，変換適用後の矩形 $\vec{Y}$ についても同様に表すものとする．

ここでまず，データ矩形の四隅 $X_{real2l} + X_{imag2l}i$，$X_{real2l} + X_{imag2h}i$，$X_{real2h} + X_{imag2l}i$，$X_{real2h} + X_{imag2h}i$ と mult-MBR の四隅 $M_{real2l} + M_{imag2l}i$，$M_{real2l} + M_{imag2h}i$，$M_{real2h} + M_{imag2l}i$，$M_{real2h} + M_{imag2h}i$ をそれぞれ掛け合わせ，計 16 個の複素数を算出する．次に，その 16 個の複素数を実部と虚部に分解し，それぞれの軸で最小値と最大値を算出する．それらの値をそれぞれ $MX_{real2l}, MX_{real2h}, MX_{imag2l}, MX_{imag2h}$ とすると，次式により変換適用後の矩形 $\vec{Y}$ は計算される．

$$\begin{aligned} Y_{real2l} &= MX_{real2l} + A_{real2l} \\ Y_{real2h} &= MX_{real2h} + A_{real2h} \\ Y_{imag2l} &= MX_{imag2l} + A_{imag2l} \\ Y_{imag2h} &= MX_{imag2h} + A_{imag2h} \end{aligned} \tag{7}$$

以上の計算を 3 番目の DFT 係数についての次元に対しても同様に行うことで，変換が適用される．

3.3 クラスタリング

先行研究 [1] では MT インデックスにおけるクラスタリング手法が明記されていないため，ここでは変換のクラスタリング手法を考える．変換 t は $\vec{a}$ と $\vec{b}$ の組であるので，クラスタリングの際には $\vec{a}$ と $\vec{b}$ を一括りにして考慮する必要がある．そこで，クラスタリング時には，$\vec{a}$ と $\vec{b}$ を結合して 1 つの変換とする．すなわち，1 つの変換 t を，$\vec{a}$ の 2 番目の DFT 係数の実部，虚部，3 番目の DFT 係数の実部，虚部，$\vec{b}$ の 2 番目の DFT 係数の実部，虚部，3 番目の DFT 係数の実部，虚部という要素を持つ 8 次元ベクトルであるとして扱う．このとき，$\vec{a}$ と $\vec{b}$ の値間に大きな差がある場合には，クラスタリングの際に値の小さいものが無視されてしまうという問題が考えられる．そこで，$\vec{a}$ と $\vec{b}$ に正規化を行ったのちに FFT を適用して得られる DFT 係数を利用することで，$\vec{a}$ と $\vec{b}$ の値間の格差をなくす方法をとる．

変換 t_i $(i = 1, \ldots, m)$ をまとめて MBR とする手法のアルゴリズムを以下に示す．

1. 変換の点の中から，線形コスト法により最も遠い 2 点をシードとして選ぶ．
2. t_i がシードであれば手順 3 へ．シードでなければ t_i が各シードからある閾値 α 以内の距離にあるかを計算する．すべてのシードとの距離が α を超えていれば，その点を新たなシードとする．
3. $i = i + 1$ として手順 2 を実行．$i \geq m$ となれば $i = 0$ として手順 4 へ．この時点で 2 個以上のシードが選出されており，それぞれをクラスタとする．
4. t_i がシードであれば手順 5 へ．シードでなければ t_i がどのクラスタに含まれるかを，ウォード法により得られた距離の最も小さいクラスタに加え，MBR とする．
5. $i = i + 1$ として手順 4 を実行．$i \geq m$ となればクラスタリングが完了する．

手順 1 の線形コスト法では，各次元について最小値と最大値の差を求め，その差が最大となる 2 点をシードとする．また，手順 4 のウォード法では，点 p とクラスタ矩形 R に対し，次式により距離を計算する．

$$d(p, R) = E(p \cup R) - E(R) \tag{8}$$

で定義される $d(p, R)$ を p と R の距離とする．ここで，$E(A)$ は A のすべての点から A の重心までの距離の二乗の総和である．なお，$p \cup R$ も矩形であるとして計算を行う．

4 実験

ここでは，先行研究 [1] の良好性の検証のための実験について述べる．実験は，先行研究 [1] で行われているものの中から，変換の適用によるオーバヘッドが無いかを確かめるものと，逐次検索，ST インデックス，MT インデックスの近接問い合わせの実行時間を比較するものの 2 つを行った．それぞれに共通して，以下の条件で実験を行った．

- 実験には以下のマシンを用いた．
  OS:CentOS 5.7，CPU:Intel Xeon 2.40GHz(4 コア，8 スレッド ×2 基)
- 検索対象とする系列は，以下の式で表される人工系列を使用した．

$$\begin{gathered} \vec{x} = [x_t], \quad x_t = x_{t-1} + z_t \\ x_0,\ z_t : [-500, 500] \text{ の範囲の乱数} \end{gathered} \tag{9}$$

- 系列の長さは 128 とし，系列の個数は 500 ~ 12000 で変化させた．
- 各実験を 1000 回ずつ実行し，実行時間にはその平均値を採用した．
- 問い合わせ系列は，毎回検索対象となる人工系列の中からランダムに 1 つを設定した．

4.1 変換の有無による比較

ここでは，変換を用いた近接問い合わせ，および，変換を用いていない近接問い合わせとの実行時間の比較を行う．ここでの実行時間とは，2.4 節でのクエリ 1 をアルゴリズム 1 を用いて処理を行ったときの，後処理部分を含まない検索に要する時間を指す．

比較のために，変換には次式で表される識別変換 $t_I = (\vec{I}, \vec{0})$ を用いる．ここで，$\vec{I}$ は全要素が 1 のベクトル，$\vec{0}$ は全要素が 0 のベクトルである．識別変換 t_I は，系列を変換前と同じ系列に変換するので，変換の有無にかかわらず問い合わせの結果は同じになる．

また，使用するアルゴリズム 1 において，n を系列の長さとしたとき，先行研究 [1] に従い，ϵ は次式により設定を行う．

$$\epsilon = \sqrt{2(n-1)(1-\rho)} \tag{10}$$

なお，ここでは $\rho = 0.85$ として ϵ を設定した．

変換の有無によるクエリ 1 の実行時間を図 1 に示す．図 1 より，変換の有無により実行時間にほとんど差がないことがみられ，変換の適用によるオーバヘッドがほとんど無いことがわかる．

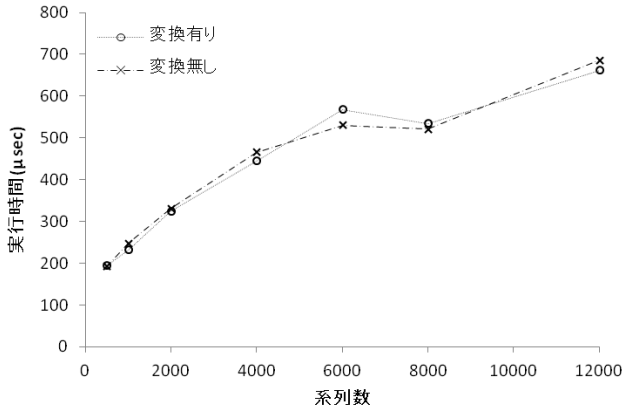

図 1　系列数ごとのクエリ 1 の実行時間

4.2　逐次検索，ST インデックス，MT インデックスの比較

ここでは，逐次検索，ST インデックス，MT インデックスの近接問い合わせの実行時間の比較を行う．MT インデックスについては，すべての変換を 1 つのクラスタとした場合と，3.3 節のアルゴリズム内の閾値を $\alpha = 0.5$ とした場合の 2 通りを比較した．なお，ST インデックスおよび MT インデックスについては，実行時間に後処理の時間も含んでいる．

変換の集合には，10 ~ 25 日間の移動平均，すなわち 16 個の変換を用いた．このとき，$\alpha = 0.5$ の MT インデックスでは，それぞれ 1 ~ 3，4 ~ 7，8 ~ 12，13 ~ 16 日間の移動平均の計 4 つのクラスタにクラスタリングされていた．

また，$\rho = 0.96$ とし，式 (10) により ϵ を設定した．

各手法のクエリ 2 の実行時間を図 2 に示す．なお，縦軸は対数目盛となっている．図 2 については，ST インデックス，$\alpha = 0.5$ の MT インデックス，1 クラスタの MT インデックス，逐次検索の順に良いパフォーマンスとなった．

5　考察

図 1 から分かるように，変換による検索性能上のオーバヘッドはほとんど無い．これは，先行研究 [1] と同様の結果である．先行研究 [1] の手法を採用する場合に，変換によるオーバヘッドが最も懸念される点であるが，変換を行っても問題ないことが分かる．また，系列数が 8000 の時点で実行時間に少し波ができるのは，そこでインデックスの階層が 2 層から 3 層へと変化しているためであると考えられる．

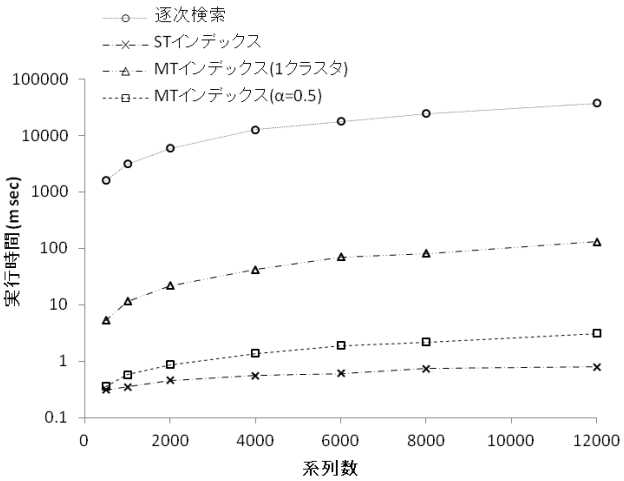

図 2　系列数ごとのクエリ 2 の実行時間

図 2 については，MT インデックスが ST インデックスよりも悪い結果となってしまっている．しかし，先行研究 [1] では，MT インデックスの方が ST インデックスよりも良い性能であることが実験的に示されている．後処理では，候補数に比例して処理に時間がかかっている．そして，MT インデックスの方が ST インデックスよりも長い実行時間がかかっているのは，MT インデックスがより多くの候補を算出し，その分後処理部分に時間を要したためではないかと考えられる．1 クラスタの MT インデックスでは，変換の領域が大きくなり，その結果，求める領域が大きくなってしまっている．この結果，多くの候補が選出されてしまい，実行時間が長くなってしまっている．従って，クラスタ数は多少多くなっても，候補数が少ないクラスタとする必要があると考えられる．

また，後処理部分のアルゴリズムの改善も必要と考えられる．後処理部分では，距離の計算のために元の系列に変換を適用するが，今回の実験では周波数領域で変換を適用している．そのためには，毎回 FFT を行った後にベクトル演算を行う必要があり，それが後処理部分に時間を要した原因であると考えられる．あらかじめ系列に FFT を施したものをメモリ上に記憶するなどの方法をとれれば，後処理部分の高速化が期待できる．

6　おわりに

本稿では，時系列にある種の変換を施した結果と類似したものを求めることを目的として，先行研究 [1] の手法を実装し，実装上の不明瞭な点の明確化を行った．また，実験により，変換のオーバヘッドがないことを証明し，ST インデックス，MT インデックスの双方が逐次検索よりも優れており，また数個にクラスタリングされた MT インデックスが 1 クラスタの MT インデックスよりも優れていることを示した．

今後の課題としては，先行研究 [1] の実験で行われている空間結合問い合わせや，クラスタ内の変換の個数を変化させた場合の実行時間の観察等を行い，さらなる有効性の検証を行うこ

とが挙げられる．

参考文献

[1] D. Rafiei and A. O. Mendelzon，"Querying Time Series Data Based on Similarity"，*IEEE Transactions on Knowledge and Data Engineering, Vol.12, No.5*，pp.675-693，2000．

[2] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases"，*Proc. Fourth Int'l Conf. Foundations of Data Organizations and Algorithms*，pp.69-84，1993．

[3] E. Keogh, K. Chakrabarti, and S. Mehrotra, M. Pazzani，"Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases"，*Proc. ACM SIGMOD2001*，pp.151-162，2001．

[4] R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Shim，"Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases"，*Proc. 21st Int'l Conf. Very Large Data Bases*，pp.490-501，1995．

[5] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait，"Querying Shapes of Histories"，*Proc. 21st Int'l Conf. Very Large Data Bases*，pp.502-514，1995．

[6] 櫻井　保志，C. Faloutsos，山室　雅司，"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"，電子情報通信学会論文誌 *D, Vol.J92-D, No.3*，pp.338-350，2009．

[7] J. Hellerstein, M. Kornacker, M. Shah, and M. Thomas，"The GiST Indexing Project"，*http://gist.cs.berkeley.edu/*，1999．