CD-002

相関時系列データ集合の計算のための高速アルゴリズム
Fast Algorithms for Correlated Time-series Set Computation

天方 大地‡　　原 隆浩‡
Daichi Amagata　Takahiro Hara

1. はじめに

時系列データは、連続的に観測されたデータのシーケンス（列）である。時系列データの解析およびマイニングは、行動分析、経済市場、センサネットワーク、およびデータセンタなどに代表される、大量のデータを生成する多くのアプリケーションにとって必要不可欠である。そのため、クラスタリング[1]やクラス分類[2]、類似検索[3]等の分析技術に関する研究が盛んに行われている。本論文では、時系列データの相関の発見に関する問題に着目する。時系列データの相関の発見も時系列データ分析の重要なツールとして知られている[4]。

多くのデータマイニングタスクにとって、与えられたデータ集合から未知のパターンを発見するツールは、そのデータ集合に潜在するルールや特徴を抽出する重要な役割を果たす。互いに相関し合うデータの集合は、そのようなパターンを示すことが直感的に分かる。一般的に、データ集合のサイズは非常に大きいため、ビジュアライゼーションにより相関し合うデータを探索することは非現実的である。そのため、与えられたデータ集合から、相関し合うデータのみを効率的に抽出する問題に取り組む。ある 2 つの時系列データtおよびt'のピアソン相関を$\rho(t,t')$とする。ユーザが指定する出力の集合のサイズをk、ピアソン相関の閾値をθとする。時系列データの集合 T が与えられた時、本問題は、T の部分集合であるAを計算する。Aは、$|A|=k$であり、$\forall t,t' \in A$ に対して、$\rho(t,t') \geq \theta$を満たし、$\rho(t,t')$の最小値が最大化されたものである。また、Aを相関時系列データ集合と呼ぶ．（2 章にて定義を紹介する。）本問題は、パターン検出、データ探索、および科学的観察のサポートに役立つ。例えば図 1 は、StarLightCurves[1] データにおける相関時系列データ集合（$k=50, \theta=0.8$）を表しており、50 個の時系列データ（光度曲線）がこのパターンを表している。光度曲線は、恒星で起きているイベントや内部で起きている処理を表していることが知られている。そのため、この問題で得た結果（図 1）から、専門家は頻繁に起きているイベントやデータが観測された時に何が起きていたかを知ることができる。

課題． 2 章で示すが、本問題は NP 困難である。そのため、厳密解を求めることは非現実的であり、本論文では近似解を計算するヒューリスティックアルゴリズムについて考える。アプリケーションの要求に答えるためには、以下の課題を解決する必要がある。

- 高精度性：厳密解を求めることは現実的でないため、互いに相関している時系列データが存在している空間を発見する必要がある。ここで、指定される閾値はユーザごとに異なるため、そのような空間を事前処理（オフライン処理）で求めることはできない。
- 高速性：先述したアプリケーションでは、k やθを変えながら相関時系列データ集合を計算することが考えられる。そのようなインタラクティブな探索を可能とするためには、高精度な解を高速に計算するアルゴリズムが必要である。

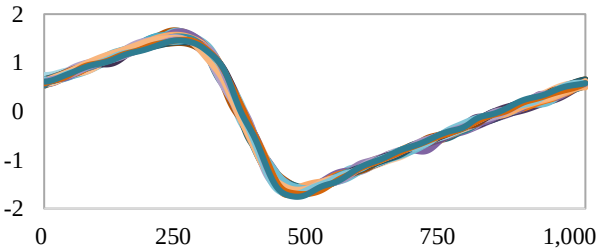


図 1　StarLightCurves データにおける相関時系列データの集合（$k=50, \theta=0.8$）

本研究では、上記の課題を解決し、3 つのグリーディアルゴリズムを提案する。1 つ目のアルゴリズムは、従来のグリーディアルゴリズム[5]に相関行列[6]を統合したものである。このアルゴリズムの計算量は$O(|T|^2 l)$である（lは時系列データの長さ）ため、スケーラビリティに欠ける。そこで、2 つ目および 3 つ目のアルゴリズムは局所性鋭敏型ハッシュ関数（LSH, Locality-Sensitive Hashing）[7]を利用し、計算量を$|T|$の自乗オーダから線形オーダの計算量に抑制している。特に、3 つ目のアルゴリズムは LSH を用いて相関時系列データ集合が存在する空間を推定し、高精度かつ高速に解を出力する。

本研究の貢献． 本研究の貢献は以下の通りである。

- 本論文では、相関時系列データ集合を計算する問題に取り組む。筆者らの知る限り、本問題はこれまでに着手されていない。
- 本問題は NP 困難であることを示し、近似アルゴリズムを設計する。提案アルゴリズムの計算量は、$|T|$、k、およびl に対して線形であることを示す。
- 実データを用いた実験により、提案アルゴリズムの有効性を示す。

論文の構成． 以下では、2 章において本問題の詳細な定義を示す。3 章で提案アルゴリズムを紹介し、4 章で実験の結果を示す。5 章で関連研究を紹介し、6 章で結論を述べる。

2. 予備知識

問題定義． ある時系列データtは、$t=(t[1], t[2], \ldots, t[l])$であり、$t[i]$は実数、$l$ は時系列データの長さである。時系列データ集合Tにおける全ての時系列データは z-正規化されているものとし、長さはlとする[6]。$\|t,t'\|$をtとt'のユークリッド距離とすると、tとt'のピアソン相関$\rho(t,t')$は、

‡ 大阪大学大学院情報科学研究科マルチメディア工学専攻
[1] www.cs.ucr.edu/~eamonn/time_series_data/

$$\rho(t,t') = 1 - \frac{\|t,t'\|^2}{2l} \quad (1)$$

である。ここで、相関時系列データ集合T_θを定義する。

定義 1（相関時系列データ集合）． 閾値θおよび時系列データ集合 T が与えられた時、相関時系列データ集合$T_\theta \subset T$は、$\forall t,t' \in T_\theta$に対して$\rho(t,t') \geq \theta$を満たす。

$\rho(t,t') \in [-1,1]$であるため、閾値θの指定は困難でない。

ユーザが出力サイズkを指定できることは多くのアプリケーションにとって有用であり、相関時系列データ集合の中で最も望ましいものは、相関時系列データ集合中のある時系列データのペアのピアソン相関の最小値が最も大きいものである。これは以下の式で表される。

$$T_\theta^* = \text{argmax}_{T_\theta \subseteq T, |T_\theta| = k} f(T_\theta) \quad (2)$$

$$f(T_\theta) = \min_{t,t' \in T_\theta} \rho(t,t') \quad (3)$$

もし、Tの中にサイズがkの相関時系列データ集合が存在しない場合、

$$T_\theta^* = \text{argmax}_{T_\theta \subseteq T} f(T_\theta) \quad (4)$$

である。以下に本論文で考える問題を定義する。

定義 2（相関時系列データ集合計算問題）． 時系列データ集合T、出力サイズk、および閾値θが与えられた時、サイズがkの相関時系列データ集合が存在すれば、式(2)を満たす集合Aを計算する。そのような集合が存在しない場合、式(4)を満たす集合Aを計算する。

ここで、本問題の特徴を紹介する。

定理 1. 本問題は NP 困難である。

証明. まず、T中にサイズがkの相関時系列データ集合が存在すると想定する。この時、本問題は k-dispersion 問題[5]と同義となる。この k-dispersion 問題は、ノードの集合$V = \{v_1, v_2, \ldots, v_{|V|}\}$が与えられた時、$|V'| = k$かつ$dist(v,v')$の最小値が最大である$V'$を計算するものである。この問題は NP 困難であることが示されており、本論文で考える問題に対して、時系列データtはノードv、$\rho(t,t')$は$dist(v,v')$に相当する。次に、T中にサイズがkの相関時系列データ集合が存在しないと想定する。このとき、本問題はグラフにおけるクリーク探索問題と同義であり、NP 困難である。

定理 1 から、厳密解を求めることは困難であるため、高精度の近似解を求める高速アルゴリズムが必要となる。ここで、Tにサイズがkの相関時系列データ集合が存在するかどうかは事前に知り得ないため、インクリメンタルにAを計算し、Aが相関時系列データ集合であることを保証するアルゴリズムを考える。

基礎技術. 提案アルゴリズムを紹介する前に、ピアソン相関が閾値以上であるかを効率的に計算する技術を紹介する。もし単純にピアソン相関を計算すると、$\Theta(l)$時間が必要である。lは基本的に大きいため、$\rho(t,t') < \theta$である場合に非効率的である。そこで、次元数（時系列データの長さ）を削減するために PAA（Piecewise Aggregate Approximation）[7]と呼ばれる技術を用いる。この技術は、長さlの時系列データtを長さϕ（$\phi \ll l$）のもの$\bar{t}$に圧縮する。

$$\bar{t}[i] = \frac{\phi}{l} \sum_{j=\frac{l}{\phi}i}^{\frac{l}{\phi}(i+1)-1} t[i]$$

補助定理 1 [7]. 以下の不等式が成り立つ。

$$\sqrt{\frac{l}{\phi}} \|\bar{t},\bar{t}'\| \leq \|t,t'\| \quad (5)$$

補助定理 1 は、式(1)および(5)から、ピアソン相関の上界値が計算可能であることを示している。ここで、各時系列データの PAA は事前に計算されているものとする。

Algorithm 1: PEARSONCORRCOMP(t,t',θ)

```
1 if sqrt(l/φ)‖t̄,t̄'‖ > sqrt(2l(1−θ)) then
2     return −1
3 else
4     τ ← 0
5     for i = 1 to l do
6         τ ← τ + (t[i] − t'[i])^2
7         if 1 − τ/2l < θ then
8             return −1
9 return 1 − τ/2l
```

アルゴリズム 1 に、$\rho(t,t') \geq \theta$であれば$\rho(t,t')$を、$\rho(t,t') < \theta$であれば-1を返すアルゴリズムを示す。1 行目で、tとt'のユークリッド距離（ピアソン相関）の下界（上界）値を計算する。もし$\rho(t,t') \geq \theta$であれば、正確なユークリッド距離を計算する。ユークリッド距離はインクリメンタルに計算できるため、閾値を満たさないことが分かり次第計算を終了し（7-8 行目）、そうでなければピアソン相関の値を返す。

3. 提案アルゴリズム

本論文の提案アルゴリズムは、k-dispersion 問題のグリーディアルゴリズムに基づいて設計されている。このグリーディアルゴリズムは以下のように動作する。出力サイズkおよびノード集合Vが与えられた時、

(1) 距離が最大となるノードのペア(v,v')をV'に入れる。
(2) $f(V',v) = \min_{v' \in V'} dist(v,v')$ とし、$f(V',v)$が最大となる$v \in V \backslash V'$をV'に追加する。
(3) $|V'| = k$となるまで(2)を繰り返す。

既存研究により、このアプローチは高精度の解を計算することが示されている。また、インクリメンタルに解を計算するため、効率的に相関時系列データ集合を計算できる。さらに、このアプローチは近似率を保証する。

定理 2 [5]. 文献[5]で提案されたグリーディアルゴリズムは、

$$f(A) \geq \frac{f(A^*)}{2}$$

を満たす。ここでA^*は最適解を示している。

まず、上のフレームワークおよび相関行列を用いたベースラインアルゴリズム Greedy-M（Greedy algorithm with Matrix）を提案する。このアルゴリズムはスケーラビリティに欠け、メモリ使用量も大きいことを示す。そこで、よ

り効率的なアルゴリズムである Greedy-L（Greedy algorithm with Locality sensitive hashing）を提案する。このアルゴリズムは局所性鋭敏型ハッシュ関数 LSH を用いており、近似率を確率的に保証する。しかし Greedy-L はデータ集合全体を何度もスキャンする必要があるため、データ集合が大きい場合に性能が低下する。そのため、さらに効率的なアルゴリズムである Greedy-L+を最後に提案する。

3.1 Greedy-M

まず、相関行列を定義する。

定義 3（相関行列）． 時系列データ集合Tが与えられた時、相関行列のセル$c_{i,j}$は以下の通りである。

$$c_{i,j} = \begin{cases} \rho(t_i, t_j) & (\text{if } \rho(t_i, t_j) \geq \theta \\ -1 & (\text{otherwise}) \end{cases}$$

	0	1	2	3	4	5	6	7	8	9
0	-	-1			-1	-1	-1	-1		
1	-	-	-1	-1	-1			-1	-1	-1
2	-	-	-			-1	-1			-1
3	-	-	-	-			-1	-1	-1	-1
4	-	-	-	-	-	-1	-1			-1
5	-	-	-	-	-	-		-1	-1	-1
6	-	-	-	-	-	-	-	-1	-1	-1
7	-	-	-	-	-	-	-	-		-1
8	-	-	-	-	-	-	-	-	-	
9	-	-	-	-	-	-	-	-	-	-

図 2　相関行列の一例。灰色のセルが$c_{i,j} \geq \theta$を満たす。

図 2 は$|T| = 10$の場合の相関行列の一例を示している。灰色のセルは$c_{i,j} \geq \theta$を満たしている。$c_{i,j} = c_{j,i}$であるため、$c_{j,i}$は計算されない。

Greedy-M（アルゴリズム 2）は、相関行列を作成した後、ピアソン相関が最大となる時系列データのペアを探索し（2-6 行目）、そのペアをAに入れる（7 行目）。ここで、

$$f(A, t) = \min_{t \in A} \rho(t, t')$$

とすると、Greedy-M は次にTをスキャンし、

$$t^* = \text{argmax}_{t \in T \setminus A} f(A, t)$$

をAに追加する。これは相関行列から定数時間で求めることができる。この操作を$|A| = k$となるまで繰り返す。

Greedy-M は文献[15]における提案アルゴリズムの単純な拡張のため、$|A| = k$であれば、$f(A) \geq f(A^*)/2$である。また、Aが相関時系列データ集合であることも保証されている。ここで、Greedy-M の欠点は、計算量とメモリ使用量である。

定理 3. Greedy-M のメモリ使用量は$O(|T|^2)$であり、計算量は$O(|T|^2 l)$である。

Algorithm 2: Greedy-M

```
1  t, t' ← ∅, τ ← −1
2  for ∀t_i ∈ T do
3      for ∀t_j ∈ T do
4          c_{i,j} ← PearsonCorrComp(t_i, t_j, θ)
5          if τ < c_{i,j} then
6              τ ← c_{i,j}, ⟨t, t'⟩ ← ⟨t_i, t_j⟩
7  A ← {t, t'}
8  while |A| < k do
9      t* ← argmax_{t∈T\A} f(A, t)
10     if f(A, t*) ≥ θ then
11         A ← A ∪ {t*}
12     else
13         break
14 return A
```

証明． 相関行列のメモリ使用量が$O(|T|^2)$である。また、相関行列の計算量が$O(|T|^2 l)$であり、t^*の計算時間は$O(k|T|)$である。この計算は$k-2$回行われるため、Greedy-M の計算量は、$O(|T|^2 l + k^2|T|) = O(|T|^2 l)$となる。

3.2 Greedy-L

Greedy-M は計算量がデータ集合のサイズの自乗オーダのため、スケーラビリティに欠ける。しかし、ピアソン相関が最大の時系列データのペアを得るためには、総当りによるアプローチが最も有効であることが示されている[8]。そのため、既存研究は計算時間を削減するため、厳密解ではなく近似解を計算するアプローチを提案している。

局所性鋭敏型ハッシュ関数（LSH）は高次元データの近似類似検索に対するアプローチとして最も有効であることが示されている。そこで LSH を用いてピアソン相関が大きい時系列データのペアの高速計算を実現する。以下に LSH を定義する。

定義 4（LSH）． 距離r、近似率c（$c > 1$）、および確率p_1とp_2（$p_1 > p_2$）が与えられた時、ハッシュ関数hは以下の条件を満たす場合、(r, cr, p_1, p_2)-sensitive である。

- $\|t, t'\| < r$であれば、$\Pr[h(t) = h(t')] \geq p_1$
- $\|t, t'\| \geq cr$であれば、$\Pr[h(t) = h(t')] < p_2$

ユークリッド空間で用いられる LSH は以下の通りである。

$$h(t) = \left\lfloor \frac{a \cdot t + bw}{w} \right\rfloor \quad (7)$$

式(7)中のaは、各要素が標準正規分布に従うランダムな値のベクトルであり、大きさはlである。また、bは$[0, w)$から選ばれたランダムな実数であり、wは定数である。本研究では$\rho(t, t') \geq \theta$となる時系列データtおよびt'に注目しているため、これらのハッシュ値が同じとなる（または類似している）必要がある。そこで、$w = \sqrt{2l(1-\theta)}$とする。以降、$\theta_E = \sqrt{2l(1-\theta)}$および$d = \|t, t'\|$とする。$\Pr[h(t) = h(t')]$は以下のように求められる[9]。

$$p(d) = \Pr[h(t) = h(t')]$$
$$= \int_0^{\theta_E} \frac{1}{d} f_2\left(\frac{x}{d}\right)\left(1 - \frac{x}{\theta_E}\right) dx$$
$$= 2norm\left(\frac{\theta_E}{d}\right) - 1 - \frac{2}{\sqrt{2\pi}}\frac{d}{\theta_E}\left(1 - e^{-\frac{\theta_E^2}{2d^2}}\right) \quad (8)$$

ここで、$f_2(z) = \frac{2}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$であり、$norm(\cdot)$は標準正規分布に従う乱数の累積分布関数である。また、$h(t)$は以下の性質を持つ[10]。

補助定理 2. 式(7)に従う LSH は$(\theta_E, c\theta_E, p(\theta_E), p(c\theta_E))$-sensitive である。

2 つの時系列データが類似している場合、$h(\cdot)$は同じ（または似た）ハッシュ値を与えるため、全く異なるハッシュ値となる時系列データ同士を比較する必要がないことが分かる。しかし、単一の$h(\cdot)$だけでは不要な比較を削減するには十分でない。これは、類似していない時系列データも同じハッシュ値となる可能性があるためである[11]。そのため、混合 LSH $G(t) = \left(h_1(t), h_2(t), \dots, h_m(t)\right)$を用いる。$G(t)$中の各 LSH は互いに独立である。$G(t)$は$t$のキーであり、ピアソン相関が大きい 2 つの時系列データは同じ、または類似したキーとなることが分かる。

Algorithm 3: Greedy-L

```
1  for ∀t ∈ T do
2      B_K ← B_K ∪ {t} where K = (h_1(t), h_2(t), ..., h_m(t))
3  t, t' ← ∅, τ ← −1
4  for ∀B_K ∈ B where |B_K| ≥ 2 do
5      for ∀t_i ∈ B_K do
6          for ∀t_j ∈ B_K do
7              τ' ← PearsonCorrComp(t_i, t_j, θ)
8              if τ < τ' then
9                  τ ← τ', ⟨t, t'⟩ ← ⟨t_i, t_j⟩
10 Execute lines 7–14 in Algorithm 2
```

アルゴリズム. Greedy-L の詳細をアルゴリズム 3 に示す。まず、各時系列データtのキーKを計算し、バケットB_Kにtを挿入する（1-2 行目）。ここで、バケットの集合をBとする。次に、あるバケット$B_K \in B$において、$\forall t, t' \in B_K$に対してアルゴリズム 1 を実行する（つまり、ある時系列データは同じバケットに存在している時系列データとのみ比較される）。これを全てのバケットに対して実行し、計算された時系列データペアの中で、最もピアソン相関が大きいものをAに入れる。以降は Greedy-M と同様の処理を行う。

最適化. Greedy-L は相関行列を保持していないため、t^*の計算に$O(k|T|l)$時間かかる。これをキャッシュを用いて削減する。端的に言うと、各時系列データ$t \in T \backslash A$に対して$f(A,t)$をキャッシュし、解の更新毎に$f(A,t)$を更新する。これにより、t^*は$O(|T|l)$で得られる．理由は以下の通りである。時系列データtがAに存在し、$A = \{t\}$とする。$\forall t_i \in T \backslash \{t\}$に対してアルゴリズム 1 を実行し、$t^* = t'$とする。各時系列データが$f(A,t)$をキャッシュする場合、次の$t^*$を検索する際には、各時系列データは$t'$に対してのみアルゴリズム 1 を実行すれば良い。この計算量は$O(l)$である。これにより、以下の定理が得られる。

定理 4. Greedy-L のメモリ使用量は$O(m|T|)$であり、計算量は$O((m+k)l|T|)$である。

証明. 各バケットはキーKを持ち、バケットの数の上界値は$|T|$である。そのため、メモリ使用量は$O(m|T|)$となる。次に計算量について考える。各時系列データのキーの計算には$O(lm)$時間必要である。そのため、ハッシングに必要な時間は$O(lm|T|)$である。次に、$|B_K| \geq 2$となるバケットの数をβとし、B_K内の時系列データの数の平均をnとすると、$\langle t, t' \rangle$の計算コストは$O(\beta n^2)$である。ここで、mを十分に大きくすると、nが十分小さくなる。その結果、$O(\beta n^2) \ll O(lm|T|)$となる。また、$t^*$の計算には$O(k|T|)$時間かかり、これが$k-2$回行われることから、Greedy-L の計算量は$O((m+k)l|T|)$となる。

さらに、Greedy-L は以下の特徴を持つ。

系 1. Greedy-L によって求められるAは、少なくとも$p(\theta_E)^m$の確率で、Greedy-M によって求められるものと同様である。

証明. $\langle t, t' \rangle$が最大のピアソン相関となるペアである限り、Greedy-L と Greedy-M が同じAを返すことは自明である。$\|t_1, t_2\| = \theta_E$であるt_1およびt_2が与えられた時、式(8)から$\Pr[h(t_1) = h(t_2)] = p(\theta_E)$である。$\|t, t'\| \leq \theta_E$であるため、系 1 は成り立つ。

3.3 Greedy-L+

Greedy-L はデータ集合のサイズの線形オーダで計算できる。また、$f(A)$の値の下界値を確率的に制限することもできる。これらの特長は理論的には意味をなすが、実践的な問題が残されている。

(1) ピアソン相関が大きい 2 つの時系列データのペアは、互いに相関している時系列データのグループに存在していることが直感的ではあるが、その保証はなく、データ集合の分布に大きく依存する。また、最悪の場合、そのペアはデータ集合の外れ値となるようなデータである可能性もある。しかし、Greedy-L はそのようなペアを探索してしまうため、サイズがkの相関時系列データ集合が存在する場合にも、探索に失敗してしまう場合がある。

(2) 解をインクリメンタルに計算するために、Greedy-L はt^*を探索するごとに全てのデータをスキャンしている。しかし、各$t \in A$に対して、$\rho(t^*, t)$は大きいため、解に含まれる時系列データは同じまたは類似したキーを持つはずである。

本論文の主要なアルゴリズムである Greedy-L+は、上記の問題を解決する。ここで、最初に解に入れられる時系列データのペア$\langle t_i, t_j \rangle$は最終的な解に大きな影響を与えるため、以下の要件を満たす必要がある。

- $\rho(t_i, t_j)$を可能な限り大きくする：式(3)の$f(A)$は劣モジュラ性をもつため、つまり、$f(A) \geq f(A \cup \{t\})$であるため、$A$に初めて入れる 2 つの時系列データのピアソン相関の値は大きくなければならない。
- $\langle t_i, t_j \rangle$と相関している時系列データのグループを効率的に探索する：この要件は$|f(A)| = k$を満たすために必要である。

以下で、どのように上記のペアを探索するかについて述べる。ある時系列データ$t \in T$が、キー$G(t)$を持つとする。各バケット$B_K \in B$において、B_Kでピアソン相関が最大となる時系列データのペアを検索し、その値をρ_Kとする。この値は大きいほど優れている。次に最近傍バケットを定義し、そのサイズについて考える。

定義 5（最近傍バケット）． $B_K \in B$が与えられた時、B_Kの最近傍バケットである$B_{K'}$は以下を満たす。

$$\left|\{i | h_i^K = h_i^{K'}\}\right| = m - 1$$

ここで、h_i^K（$h_i^{K'}$）は、K（K'）の i 番目のハッシュ値である。

$\rho_K \geq \theta$となるバケットの集合をB_θとする。B_θ内の各バケットB_Kに対して、その最近傍バケットを検索し、s_Kを計算する。これは以下の式で求められる。

$$s_K = |B_K| + \sum |B_{K'}|$$

キーが同じ、または類似したバケット内の時系列データは相関関係にある可能性が高い。そのため、s_Kが大きい場合、B_Kにはサイズが大きい相関時系列データ集合が含まれている可能性が高い。このアイデアを基に、Greedy-L+は解に最初に入れる時系列データのペアを選択する。具体的には、B_θの中で$\rho_K \cdot \frac{s_K}{|T|}$が最大となるバケットにおいてピアソン相関が最も大きい時系列データのペアを検索する。（$\rho_K \in [\theta, 1]$であるため、s_Kを正規化する必要がある。）

次に、効率的にt^*を検索する方法について述べる。これは最近傍バケットを用いる。ある時系列データとのピアソン相関が大きい時系列データは、同じまたは類似したキーのバケットに存在しているため、Aに含まれる時系列データが存在するバケットおよびその最近傍バケットの集合の中から探索すれば良い。この集合をSとすると、$f(A) \geq \theta$となるtがSに存在する確率は、以下で与えられる。

補助定理 3. 以下の不等式が成り立つ。

$$\Pr[t \in S, f(A,t) \geq \theta] \geq p(\theta_E)^m + p(\theta_E)^{m-1}(1 - p(\theta_E))m$$

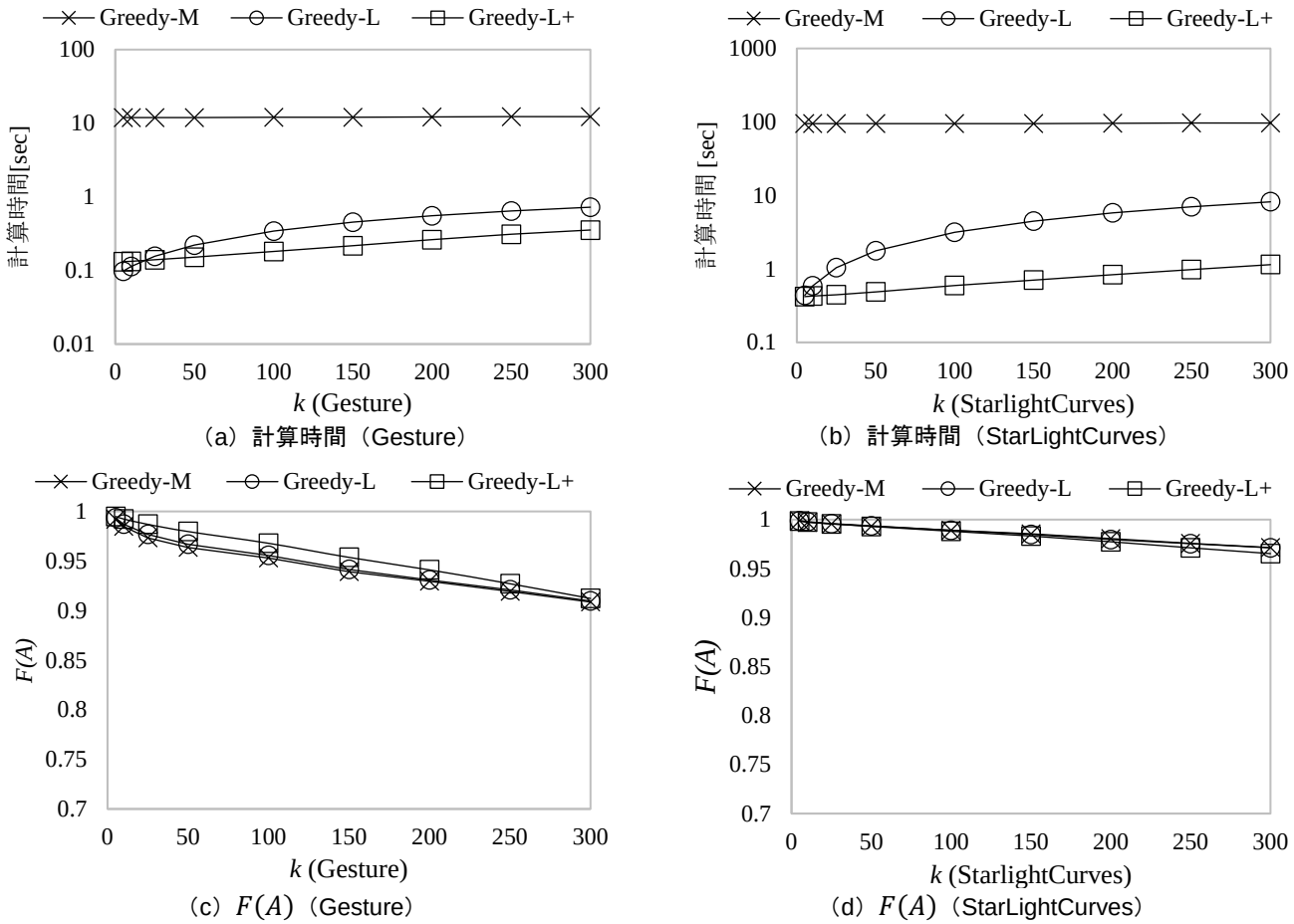
アルゴリズム． アルゴリズム 4 に Greedy-L+の詳細を示し、定理 5 にメモリ使用量と計算量を示す。Greedy-L と同様に各時系列データのキーを計算する（1-2 行）。その後、$\rho_K \cdot s_K/|T|$が最大となるバケットにおけるピアソン相関が最大の時系列データのペアを計算し、Aに入れる（3-21 行）。次に、Sから$t^* = \text{argmax}_{t \in S \setminus A} f(A,t)$を計算し、$A$に入れる。同時に、$t^* \in B_K$である$B_K$とその最近傍バケットの集合を$S$に加える。（Greedy-L+は Greedy-L と同様にキャッシュも用いる。）この操作を、$|A| = k$となるまで、または$f(A,t) \geq \theta$となるtがSに存在しないことが分かるまで繰り返す。

Algorithm 4: Greedy-L+

```
1  for ∀t ∈ T do
2      B_K ← B_K ∪ {t} where K = (h_1(t), h_2(t), ..., h_m(t))
3  B_θ ← ∅, P ← ∅
4  for ∀B_K ∈ B where |B_K| ≥ 2 do
5      t_K, t'_K ← ∅
6      ρ_K ← −1
7      for ∀t_i ∈ B_K do
8          for ∀t_j ∈ B_K do
9              τ ← PearsonCorrComp(t_i, t_j, θ)
10             if ρ_K < τ then
11                 ρ_K ← τ
12                 ⟨t_K, t'_K⟩ ← ⟨t_i, t_j⟩
13     if τ ≥ θ then
14         B_θ ← B_θ ∪ B_K
15         P ← P ∪ ⟨ρ_K, t_K, t'_K⟩
16 t, t' ← ∅, μ = 0
17 for ∀B_K ∈ B_θ do
18     s_K ← |B_K| + Σ|B_K'| where B_K' ∈ B_θ is the
       nearest bucket of B_K
19     if ρ_K · s_K/|T| > μ then
20         μ ← ρ_K · s_K/|T|
21         ⟨t, t'⟩ ← ⟨t_K, t'_K⟩
22 A ← ⟨t, t'⟩
23 S ← B_K ∪ B^N_K' where t, t' ∈ B_K and B^N_K' is the set of
     the nearest bucket of B_K in B
24 while |A| < k do
25     t* ← argmax_{t∈S\A} f(A, t)
26     if f(A, t*) ≥ θ then
27         A ← A ∪ {t*}
28         S ← S ∪ B_K ∪ B^N_K' where t* ∈ B_K and B^N_K'
             follows line 23
29     else
30         break
```

定理 5. Greedy-L+のメモリ使用量は$O(m|T|)$であり、計算量は$O((m+k)l|S|)$である。

証明． メモリ使用量については Greedy-L と同様のため、自明である。定理 4 の証明から、1-21 行目までの操作は$O(lm|T|)$である。各ハッシュ関数の出力値をキャッシュすると、s_Kは$O(m)$時間で計算できる。つまり、17-21 行目までの操作は、$O(m|B_\theta|)$時間かかるが、$O(m|B_\theta|) \ll O(lm|T|)$である。23 行目、25 行目、および 28 行目はそれぞれ$O(l|S|)$時間かかる。そのため、計算量は$O((m+k)l|S|)$となる。



（a）計算時間（Gesture）　（b）計算時間（StarLightCurves）

（c）$F(A)$（Gesture）　（d）$F(A)$（StarLightCurves）

図 3　kの影響

議論. 最近傍バケットは、解の候補を効果的に選択するために重要である。ここで、最近傍以外のバケットを使うことも考えられる。しかし、最近傍バケット以外にも注目すると、組み合わせの数が$\binom{m}{m'}$となり、Sのサイズが非常に大きくなってしまう。また、m'の指定も容易ではない。これらの理由から、Greedy-L+は最近傍バケットのみを用いている。

4. 実験

本章では、提案アルゴリズム（Greedy-M、Greedy-L、および Greedy-L+）の評価実験の結果について説明する。全てのアルゴリズムは C++で実装されており、全ての実験を、Intel Xeon E5-2687W v4（3.0GHz、12 コア）および 512GB メモリの PC 上で行った。

4.1 設定

データ. 本実験では、以下の実データを用いた。いずれのデータも UCR Archive から入手可能である。

- Gesture：ジェスチャー時の加速度の実データであり、長さが 315 のデータが 13,434 存在する。
- StarLightCurves：光度曲線の実データである。長さが 1,024 のデータが 9,236 個存在する。

また、データはメモリ上に置かれている。

パラメータ. $k = 100$、$\theta = 0.8$をデフォルトの値とした。文献[13]から、$\phi = 10$とした。また、予備実験から、各アルゴリズムにおけるmを表 1 のように設定した。

表 1　各アルゴリズムの各データに対するm

アルゴリズム	Gesture	StarLightCurves
Greedy-L	13	26
Greedy-L+	8	30

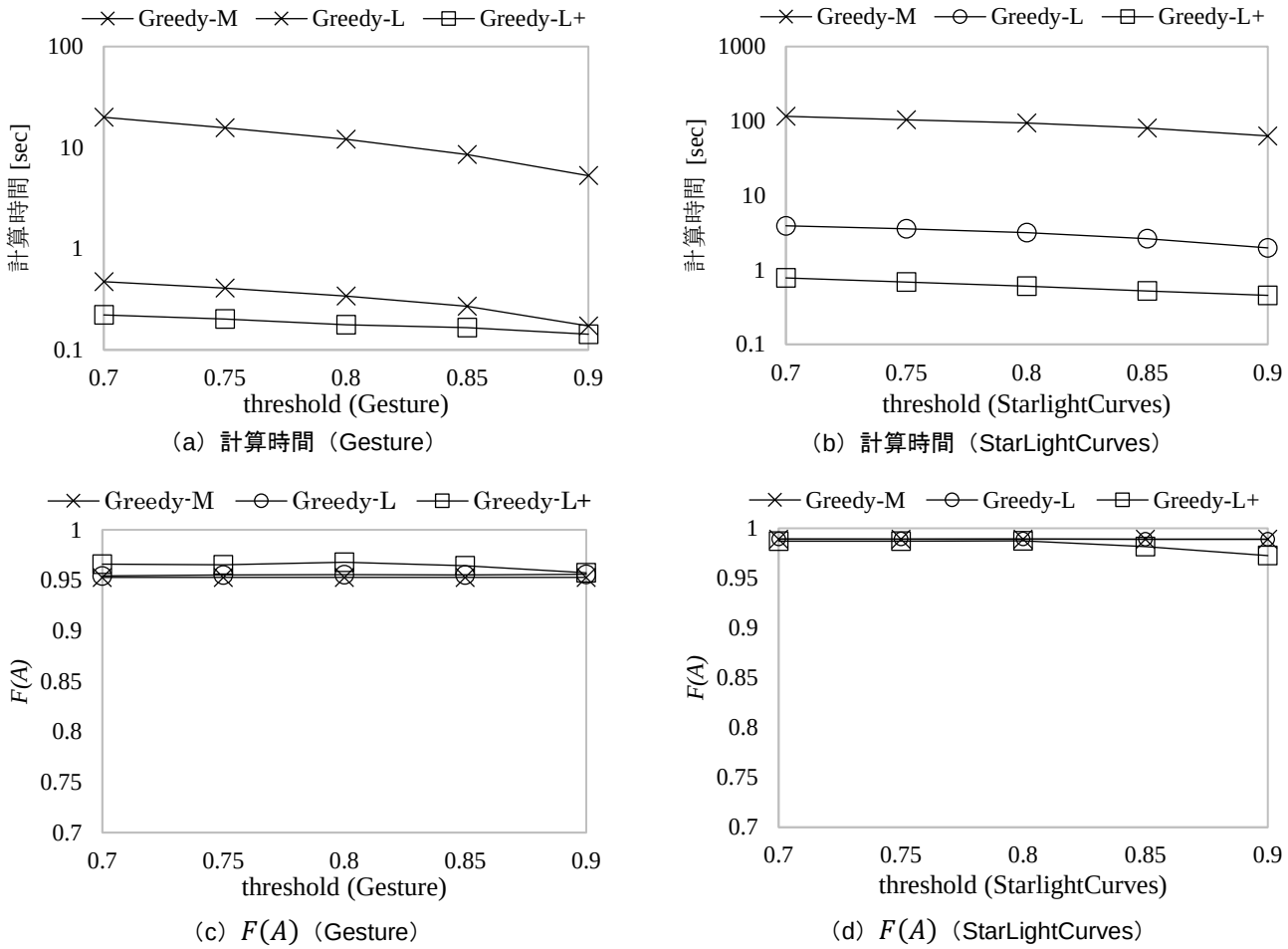
評価値. 各実験を 50 回行い、以下の評価値の平均値を調べた。

- 計算時間：解を得るまでにかかった時間。
- $F(A) = f(A) \cdot \frac{|A|}{k}$：各アルゴリズムは相関時系列データ集合を出力することを保証しているが、そのサイズは保証できない。つまり、各アルゴリズムが出力する集合のサイズが異なる可能性があるため、正規化した値を用いる。

ここで、定理 1 に示した通り、正確な$F(A)$を求めることは現実的でない。そのため、正確な$F(A)$との比較は行わない。

4.2 結果

***k*の影響.** 図 3 にkの影響の結果を示す。まず計算時間について考察する（図 3(a)-3(b)）。定理 3 が示す通り、Greedy-



（a）計算時間（Gesture）　（b）計算時間（StarLightCurves）

（c）$F(A)$（Gesture）　（d）$F(A)$（StarLightCurves）

図 4　θの影響

M はkの影響を受けず、一定の計算時間である。しかし、データ数に対して自乗オーダの計算量のため、Greedy-L と Greedy-L+に比べて計算時間が非常に長い。一方、定理 4 および 5 が示す通り、Greedy-L と Greedy-L+は計算時間がkに対して線形である。また、Greedy-L+の方が Greedy-L よりも計算時間が早いことが分かる。これは定理 5 の裏付けとなる結果でもあり、Greedy-L+のスケーラビリティが他のアルゴリズムよりも高いことを示している。

次に$F(A)$に着目する。図 3(c)から、Greedy-L+は他のアルゴリズムよりも高い値を示していることが分かる。これにより、最近傍バケットを用いたアプローチの有効性が示される。一方、図 3(d)では、いずれのアルゴリズムも同じような値となっている。これは StarLightCurves には高いピアソン相関となる時系列データのペアが大量に存在しているためである。

***θ*の影響.** 図 4 にθの影響の結果を示す。図 4(a)および 4(b)から、θが大きくなると計算時間が短くなっていることが分かる。これは、θが大きくなるとアルゴリズム 1 の実行時間が短くなるためである。また、θが大きい場合にも Greedy-M は低速であり、他のアルゴリズムの優位性が示されている。図 4(c)および 4(d)からはkの影響と同じような結果が得られていることが分かる。

5. 関連研究

既存研究において、多くの距離指標（p-ノルムや DTW 等）が提案されているが、十分に長い時系列データに対しては、ユークリッド距離と DTW はほぼ同様の結果となることが示されている[12]。

ピアソン相関は時系列データの重要な類似指標の一つであり、相関している時系列データのペアの検索に関する研究に注目が集まっている。文献[4, 6]は与えられた時系列データ集合の中の相関している時系列データの全てのペアを計算する問題に取り組んでいる。文献[13, 14]は効率的なモチーフ発見アルゴリズムを提案している。モチーフとは、ある時系列データの部分シーケンスのペアの中で最も類似しているものである。これらの研究は一つの時系列データに着目しているため、相関時系列データ集合を探索する問題とは異なる。

局所鋭敏型ハッシュ関数（LSH）も類似検索における重要な技術であり、計算幾何学、データベース、およびデータマイニング分野において盛んに研究されている。高次元データに対する類似検索の高速化のために式(7)の派生関数が数多く提案されている[9, 10, 11]。これらの研究では、LSH をインデックス（オフライン処理）として用いており、ハッシュ関数の計算にかかる時間は遅く、ストレージコストも大きい。本論文の提案アルゴリズムは LSH をオンラ

イン処理に用いており、ユーザが指定した閾値に対応できる。また、定理 4 および 5 に示されるように、メモリ使用量も大きくない。

6. おわりに

本論文では、相関時系列データ集合を高速に計算する問題に取り組んだ。本論文は NP 困難であることを示し、グリーディアルゴリズムを提案した。局所鋭敏型ハッシュ関数を有効に利用することにより、互いに相関している時系列データが多く存在している空間を発見する新しい技術を考案した。実データを用いた実験の結果から、提案アルゴリズムは効率的であり、スケーラブルであることを示した。

謝辞

本研究の一部は、文部科学省科学研究費補助金・基盤研究(A)(JP26240013)、若手(B)(JP16K16056)、および JST 国際科学技術共同研究推進事業（戦略的国際共同研究プログラム）の研究助成によるものである。ここに記して謝意を表す。

参考文献

[1] Paparirizons John, Gravano Luis, “k-shape: Efficient and accurate clustering of time-seris”, Proc. SIGMOD (2015).
[2] Ye Lexiang, Keogh Eamonn, “Time series shaplets: A new primitive for data minign”, Proc. KDD (2009).
[3] Peng Jinglin, Wang Hongzhi, Li Jianzhong, “Set-based similarity search for time series”, Proc. SIGMOD (2016).
[4] Guo Tian, Sathe Saket, Aberer Karl, “Fast distributed correlation discovery over streaming time-series data”, Proc. CIKM (2015).
[5] Ravi SS, Resenkrantz J Daniel, Tayi Kumar Giri, “Facility dispersion problems: Heuristics and special cases”, Proc. Algorithms and Data Structures (1991).
[6] Mueen Abdullah, Matj Suman, Liu Jie, “Fast approximate correlation for massive time-series”, Proc. SIGMOD (2010).
[7] Keogh Eamonn, Chakrabarti Laushik, Pazzani Michael, Mehrota Sharad, “Dimensionality reduction for fast similarity search in large time series databases”, KIS, 3, 3 (2001).
[8] Weber Roger, Schek Hans-Jorg, Blott Stephen, “A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces”, Proc. VLDB (1998).
[9] Datar Mayur, Immorlica Nicle, Indyk Pitor, Mirrokni S Vahab, “Locality-sensitive hashing scheme based on p-stable distribution”, Proc. SoCG (2004).
[10] Gan Juan, Feng Jianlin, Fang Qiong, Ng Wilfred, “Locality-sensitive hashing scheme based on dynamic collision counting”, Proc. SIGMOD (2012).
[11] Liu Yingfan, Cui Jiangtao, Huang Zi, Li Hui, Tao Heng, “Sk-lsh: An efficient index structure for approximate nearest neighbor search”, PVLDB, 7, 9 (2014).
[12] Sheih Jin, Keogh Eamonn, “iSAX: Indexing and mining terabyte sized time series”, Proc. KDD (2008).
[13] Li Yuhong, Yiu Lung Man, Gong Zhiguo, “Quick-motif: An efficient and scalable framework for exact motif discovery”, Proc. ICDE (2015).
[14] Mueen Abdullah, Keogh Eamonn, Zhu Qiang, Cash Sydney, Westover Brandon, “Eaxct discovery of time series motif”, Proc. SDM (2009).
[15] Li Yuhong, Yiu Lung Man, Gong Zhiguo, “Efficient discovery of longest-lasting correlation in sequence database”, The VLDB Journal, 25, 6 (2016).