情報処理学会論文誌 Vol.60 No.7 1260–1269 (July 2019)

ストリーミング時系列データに対するモチーフモニタリング

加藤 慎也[1,a)] 天方 大地[1,b)] 西尾 俊哉[1,c)] 原 隆浩[1,d)]

受付日 2018年10月25日, 採録日 2019年4月9日

概要：近年，多くの IoT 機器はストリーミング時系列データを生成しており，それらを環境モニタリングやイベント検知に応用することが考えられる．これを実現する 1 つの技術として，時系列データの中に最も多く現れるサブシーケンスであるレンジモチーフの発見が注目を集めている．本論文では，カウントベースのスライディングウィンドウ上でストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む．ウィンドウがスライドした際，新たにサブシーケンスが生成され，最も古いサブシーケンスが削除される．生成および削除されるサブシーケンスとウィンドウ内のすべてのサブシーケンスを比較する単純な方法は，多くの類似度の計算を必要とするため効率的でない．そのため，効率的にモチーフを更新するアルゴリズム SRMM を提案する．SRMM の時間計算量は生成および削除されるサブシーケンスと類似するサブシーケンスの数にのみ依存し，効率的にレンジモチーフをモニタリングできる．4 つの実データを用いた実験により，SRMM の有効性を確認した．

キーワード：ストリーミング時系列データ，モチーフモニタリング

Motif Monitoring for Streaming Time-series

Shinya Kato[1,a)] Daichi Amagata[1,b)] Shunya Nishio[1,c)] Takahiro Hara[1,d)]

Received: October 25, 2018, Accepted: April 9, 2019

Abstract: Recent IoT-based applications generate time-series in a streaming fashion, and they often require techniques that enable environmental monitoring and event detection from generated time-series. Discovering a range motif, which is a subsequence that repetitively appears the most in a time-series, is a promising approach for satisfying such a requirement. This paper tackles the problem of monitoring a range motif of a streaming time-series under a count-based sliding-window setting. Whenever a window slides, a new subsequence is generated and the oldest subsequence is removed. A straightforward solution for monitoring a range motif is to scan all subsequences in the window while computing their occurring counts measured by a similarity function. Because the main bottleneck is similarity computation, this solution is not efficient. We therefore propose an efficient algorithm, namely SRMM. SRMM is simple and its time complexity basically depends only on the occurring counts of the removed and generated subsequences. Our experiments using four real datasets demonstrate that SRMM scales well and shows better performance than a baseline.

Keywords: streaming time-series, motif monitoring

1. 序論

モチーフ発見は時系列データを分析する最も重要な技術の 1 つである [21]．ある時系列データ t が与えられたとき，t のレンジモチーフとは，t の中で最も多く現れるサブシーケンスである [6], [18]．つまり，レンジモチーフは頻繁に発生するサブシーケンスを表す．たとえば**図 1** は，温室効

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] kato.shinya@ist.osaka-u.ac.jp
[b)] amagata.daichi@ist.osaka-u.ac.jp
[c)] nishio.syunya@ist.osaka-u.ac.jp
[d)] hara@ist.osaka-u.ac.jp

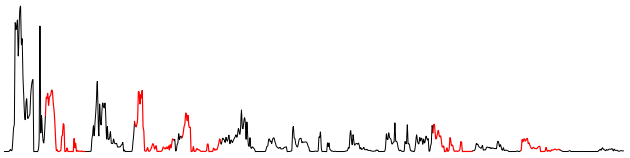


図 1 レンジモチーフの例
Fig. 1 Example of range motif.

果ガスの排出量のストリーミング時系列データ [13] の中で繰り返し現れているサブシーケンス（赤いサブシーケンス）を図示しており，最も左の赤いサブシーケンスが現在のレンジモチーフである（本論文では，サブシーケンス間の類似度を z 正規化ユークリッド距離を用いて計算するため，この図における値のスケールは問題ではない）．近年，多くの IoT 機器はストリーミング時系列データを生成するため [14]，本論文では，ストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む．今後，特に明記する必要がない場合，レンジモチーフを単にモチーフと呼ぶ．

アプリケーション例． センサ機器が定期的にデータを収集し，サーバに送信すると仮定する．さらに，専門家が時系列データをモニタリングすると仮定する．このデータをモニタリングし，時間の経過とともにモチーフが変化した場合，様々な潜在的事象を分析したり，センサデータは環境的な要因だけでなく，時間的な要因と相関があるといった仮説を立てることができる．また，イベント検知への応用も考えられる．モチーフをモニタリングし，そのモチーフを 1 分ごとに保存すると仮定する．ある時刻におけるモチーフが前日の同時刻に得られたモチーフと大きく異なる場合やそれ以前のモチーフと著しく異なる場合，何らかの異常が発生していることが予測できる．

提案アルゴリズムの概要． 上記のようなアプリケーションでは，最新の値のみを考慮し，それらのモチーフをリアルタイムにモニタリングする必要がある．そのため，カウントベースのスライディングウィンドウを用いて最新の w 個の値のみを考慮し，それらのモチーフを効率的にモニタリングするアルゴリズム SRMM（Streaming Range Motif Monitoring）を提案する [7]．ウィンドウがスライドした際，新たな値がウィンドウに挿入され，最も古い値がウィンドウから削除される．つまり，新たな値を含むサブシーケンス s_n が生成され，最も古い値を含むサブシーケンス s_e が削除される．このとき，モチーフを更新する単純な手法として，s_n および s_e とウィンドウ内のすべてのサブシーケンスを比較する手法が考えられる．この手法は，s_n および s_e と類似するサブシーケンスの数（類似サブシーケンス数）を正確に取得できるが，多大な計算コストがかかる．そこで，SRMM は，ウィンドウがスライドした際，モチーフになりうるサブシーケンスにのみ注目することにより，不要な計算を削減する．SRMM は PAA（Piecewise Aggregate Approximation）[8] および kd 木 [2] を用いて，s_n の類似サブシーケンス数の上界値を高速に計算し，正確な類似サブシーケンス数を計算する回数を削減する．具体的には，PAA により次元削減されたウィンドウ内のサブシーケンスを kd 木で管理する．そして，s_n に対して kd 木を用いた範囲検索を実行することで，s_n の類似サブシーケンスになりうるサブシーケンスの集合を得る．つまり，この集合のサイズは s_n の類似サブシーケンス数の上界値であり，また，すべてのサブシーケンスとの比較を行うことなく，正確な類似サブシーケンスの数を計算できる．さらに，ウィンドウ内の各サブシーケンスは自身と類似するサブシーケンスのリストおよび自身と類似するサブシーケンスの候補のリストを保持する．これにより，s_e が削除される際，類似サブシーケンス数が減少する可能性のあるサブシーケンスを効率的に特定できる．

貢献． 以下に本研究の貢献を示す．

- カウントベースのスライディングウィンドウ上でストリーミング時系列データのレンジモチーフをモニタリングする問題に取り組む．筆者らの知る限り，この問題はこれまでに取り組まれていない．
- ウィンドウがスライドした際，モチーフを効率的に更新するアルゴリズム SRMM を提案する．
- 4 つの実データを用いた実験により，SRMM の有効性を確認する．

本論文の構成． 2 章で本論文の問題を定義し，3 章で関連研究について述べる．4 章で SRMM について説明し，5 章で実データを用いた実験の結果を示す．最後に 6 章で本論文のまとめと今後の課題について述べる．

2. 予備知識

2.1 定義

ストリーミング時系列データ t は実数値の系列であり，$t = (t[1], t[2], \ldots)$ と表現する．まず，t の中に繰り返し現れる特徴的なパターンを発見するため，t の一部を表すサブシーケンスを定義する．

定義 1（サブシーケンス） t および長さ l が与えられたとき，p 番目のデータを始点とするサブシーケンス s_p は式 (1) により定義される．

$$s_p = (t[p], t[p+1], \ldots, t[p+l-1]) \tag{1}$$

ここで，s_p の x 番目のデータの値を $s_p[x]$ と表現する．つまり，$s_p = (s_p[1], s_p[2], \cdots, s_p[l])$ である．次に，t の中で s_p と類似したサブシーケンスの数を計算するため，本研究では，時系列データ間の類似度を測る基本的な指標であるピアソン相関を用いる [11], [16]．

定義 2（ピアソン相関） 長さ l の 2 つのサブシーケンス s_p および s_q が与えられたとき，これらのピアソン相関 $\rho(s_p, s_q)$ は式 (2) により定義される．

$$\rho(s_p, s_q) = 1 - \frac{\|\hat{s}_p, \hat{s}_q\|^2}{2l} \quad (2)$$

$\rho(s_p, s_q) \in [-1, 1]$ であり，$\|\hat{s}_p, \hat{s}_q\|$ は $\hat{s}_p$ と $\hat{s}_q$ 間のユークリッド距離を計算したものである．また，$\hat{s}_p$ は s_p を z 正規化したものであり，

$$\hat{s}_p[i] = \frac{s_p[i] - \mu(s_p)}{\sigma(s_p)}$$

である．ここで，$\mu(s_p)$ および $\sigma(s_p)$ はそれぞれ $(s_p[1], s_p[2], \ldots, s_p[l])$ の平均および標準偏差である．ピアソン相関は z 正規化ユークリッド距離 $d(\cdot,\cdot) = \|\cdot,\cdot\|$ に変換でき，式 (3) で与えられる．

$$d(\hat{s}_p, \hat{s}_q) = \sqrt{2l(1 - \rho(s_p, s_q))} \quad (3)$$

ピアソン相関の時間計算量は $O(l)$ である．次に，サブシーケンス s_p と類似するサブシーケンス（類似サブシーケンス）を定義する．

定義 3（類似サブシーケンス） s_p，s_q，およびある閾値 θ が与えられたとき，サブシーケンスどうしが類似することと次の式を満たすことは同値である．

$$\rho(s_p, s_q) \geq \theta \Leftrightarrow d(\hat{s}_p, \hat{s}_q) \leq \sqrt{2l(1-\theta)} \quad (4)$$

s_p と s_{p+1} は $l-2$ 個の値を共通に保持しており，このようなサブシーケンスは類似していたとしても，時系列データに現れるパターンとはいい難い．そのため，有用な結果を得るために，このようなサブシーケンスを考慮すべきではない．そこで，互いに重なり合うサブシーケンスをトリビアルマッチと定義する [5], [18]．

定義 4（トリビアルマッチ） s_p が与えられたとき，s_p とトリビアルマッチであるサブシーケンスの集合 S_p は次の条件を満たす．

$$S_p = \{s_q \mid p - l + 1 \leq q \leq p + l - 1\} \quad (5)$$

ここで，1 章で述べたアプリケーションを含む多くのアプリケーションでは最新の値のみを考慮している [9], [15]．そのため，ストリーミング時系列データに関する既存の研究 [4], [10] と同様に，本研究においてもカウントベースのスライディングウィンドウを用いて，最新の w 個の値のみをモニタリングする．つまり，ウィンドウ内のストリーミング時系列データ t は $t = (t[i], t[i+1], \ldots, t[i+w-1])$ のように表され，$t[i+w-1]$ が最新の値である．また，l が与えられたとき，ウィンドウ内には $w-l+1$ 個のサブシーケンスが含まれる．ウィンドウがスライドしたとき，最新の l 個の値を含む新たなサブシーケンスが生成される．同時に，最も古い値がウィンドウから削除されるため，最も古いサブシーケンスが削除される．そこで，ウィンドウサイズ w のウィンドウに含まれるすべてのサブシーケンスの集合を S としたとき，ウィンドウ内のあるサブシーケンス s_p と類似したサブシーケンスの数をスコアと定義する．

定義 5（スコア） S，l，および θ が与えられたとき，あるサブシーケンス $s_p \in S$ のスコアは式 (6) により定義される．

$$score(s_p) = |\{s_q \mid s_q \in S \backslash S_p, \rho(s_p, s_q) \geq \theta\}| \quad (6)$$

本研究では，カウントベースのスライディングウィンドウ上でスコアが最大となるサブシーケンスをモニタリングする．つまり，本研究の問題は以下のように定義される．

問題定義 t，l，θ，および w が与えられたとき，式 (7) で表されるレンジモチーフ s^* をモニタリングする．

$$s^* = \arg\max_{s \in S} score(s) \quad (7)$$

2.2 ベースラインアルゴリズム

本論文は，本問題に初めて取り組むため，まず，ベースラインとなるアルゴリズムについて考える．ウィンドウがスライドした際，新たに生成されるサブシーケンスおよび削除されるサブシーケンスに対して，ウィンドウ内のすべてのサブシーケンスとのピアソン相関を計算し，スコアを更新する．そして，ウィンドウ内のスコアが最大のサブシーケンスをモチーフとする．前述したように，ウィンドウ内には $w-l+1$ 個のサブシーケンスが含まれ，ピアソン相関の計算には $O(l)$ の時間計算量がかかる．そのため，ベースラインアルゴリズムの時間計算量は $O((w-l)l)$ である．

ここで，あるサブシーケンスのスコアに影響を与えるのは，そのサブシーケンスと類似したサブシーケンスのみであるため，すべてのサブシーケンスのスコアを更新する必要はない．そのため，ウィンドウがスライドした際，スコアを更新する必要があるサブシーケンスを効率的に特定するアルゴリズムを提案する．

3. 関連研究

本章では，モチーフ発見に関する既存研究について紹介する．モチーフという用語は，文献によって 2 つの異なる意味で用いられている [6]．1 つ目のモチーフの定義は，本論文におけるモチーフの定義と同様である．一方，文献 [11], [15], [16] では，時系列データの中で最も類似するサブシーケンスのペアをモチーフと定義している．本章では，最も類似するサブシーケンスのペアを発見する問題をペアモチーフ発見と呼ぶ．

3.1 ペアモチーフ発見

ペアモチーフ発見はサブシーケンスの数の二乗の時間計算量がかかるため，文献 [16] では，三角不等式を用いて実践的な計算時間を削減するアルゴリズム MK を提案している．MK は，いくつかのサブシーケンスを基準点とし，あるサブシーケンスのペアが与えられたときに，それらの距

離の上界値を取得する．しかし，時間計算量は依然としてサブシーケンスの数の二乗の時間計算量がかかる．より性能を向上させるため，文献 [11] では，Quick-Motif と呼ばれるアルゴリズムを提案している．Quick-Motif はオンラインでサブシーケンスのインデックスを作成することで，サブシーケンスどうしの比較回数を削減する．評価実験では，Quick-Motif は MK よりも優れていることを示している．文献 [22], [23] では，Matrix Profile と呼ばれるオフラインでインデックスを作成するアプローチを提案している．このインデックスは，すべてのサブシーケンスに対してそのサブシーケンスと最も類似するサブシーケンスとの距離を保持する．このインデックスを用いることでペアモチーフを高速に発見できる．

これらの研究は，静的な時系列データを対象としている．一方，文献 [15] では，ペアモチーフをモニタリングする問題に初めて取り組んでいる．文献 [15] の提案アルゴリズムでは，ペアモチーフを高速に更新するため，各サブシーケンスは最近傍と逆最近傍のサブシーケンスのリストを保持する．また，文献 [9] では，ペアモチーフをモニタリングするためにデータ構造を最適化したアルゴリズムを提案しており，文献 [15] のアルゴリズムより優れた性能を示している．

3.2 レンジモチーフ発見

Patel らはレンジモチーフを効率的に発見するための近似アルゴリズムを提案している [18]．このアルゴリズムでは，各サブシーケンスを SAX [12] を用いて記号列に変換する．このアルゴリズムと同様に，Castro と Azevedo は iSAX [20] を用いてレンジモチーフを発見するアルゴリズムを提案している [3]．SAX および iSAX は時系列データを記号列に近似するため，発見されたモチーフが正確であることが保証されない．また，いくつかの確率的アルゴリズムが提案されているが [5], [21]，これらのアルゴリズムも発見されたモチーフが正確であることは保証されない．文献 [6] では，学習ベースのモチーフ発見アルゴリズムを提案している．しかし，このアルゴリズムは前処理を行う必要があるため，ストリーミング時系列データに適用できない．また，これらの研究は，静的な時系列データを対象としている．

文献 [1] では，ストリーミング時系列データを対象としているが，短期間ではモチーフが出現しない場合に取り組んでいる．さらに，文献 [1] で提案されたアルゴリズムも近似手法（SAX および Bloom filter）を用いている．文献 [17] も，ストリーミング時系列データを対象としているが，時系列データを SAX で記号化したサブシーケンス間の距離に基づいてモチーフをモニタリングする．また，これらの既存研究において，解の精度保証はされていない．本論文では，正確なモチーフをモニタリングするための効率的なアルゴリズムを提案する．

4. SRMM：Streaming Range Motif Monitoring

ウィンドウがスライドした際，ウィンドウ内の各サブシーケンスのスコアは最大で 1 増加する．これは，定義 5 およびカウントベースのスライディングウィンドウの特性から明らかである．そのため，モチーフは頻繁に変化せず，新たに生成されるサブシーケンスのスコアが頻繁に $score(s^*)$ を超えることは非常にまれである．

ここで，新たに生成されるサブシーケンスを s_n としたとき，高速に $score(s_n) < score(s^*)$ であることが分かれば，正確なモチーフを効率的にモニタリングできる．これを実現するため，4.1 節において $score(s_n)$ の上界値を効率的に取得し，不要なスコアの計算を枝刈りするアルゴリズムを提案する．また，ウィンドウがスライドした際，ウィンドウからサブシーケンスが削除され，そのサブシーケンスと類似したサブシーケンスのスコアが 1 だけ減少するため，s^* が変化する可能性がある．そこで，4.2 節において，スコアが減少する可能性のあるサブシーケンスを効率的に特定するアルゴリズムを紹介する．最後に，4.3 節において SRMM の全体的なアルゴリズムを紹介し，SRMM の時間計算量について述べる．

4.1 スコアの上界値の取得

まず，s_n と $s \in S$ のピアソン相関の上界値，つまり z 正規化ユークリッド距離の下界値を取得するため，次元削減アルゴリズム PAA [8] を用いる．このとき，長さ l のサブシーケンス $s_p = (s_p[1], s_p[2], \ldots, s_p[l])$ は l 次元上の点として表現できる．ある次元 $\phi < l$ が与えられたとき，PAA により l 次元上の点を ϕ 次元上の点に変換できる．このとき，$\hat{s}_p$ を ϕ 次元上の点に変換したものを $\hat{s}_p^\phi$ とすると，$\hat{s}_p^\phi$ の各値は以下の式で表される．

$$\hat{s}_p^\phi[i] = \frac{\phi}{l} \sum_{j=\frac{l}{\phi}i}^{\frac{l}{\phi}(i+1)-1} \hat{s}_p[j]$$

また，PAA では以下の補題が成り立つ [8]．

補題 1 2 つのサブシーケンス $\hat{s}_p$ および $\hat{s}_q$ が与えられたとき，以下の関係が成り立つ．

$$\sqrt{\frac{l}{\phi}} dist(\hat{s}_p^\phi, \hat{s}_q^\phi) \leq dist(\hat{s}_p, \hat{s}_q) \tag{8}$$

PAA により，$\hat{s}_p$ と $\hat{s}_q$ の z 正規化ユークリッド距離の下界値，つまり $\rho(s_p, s_q)$ の上界値を $O(\phi)$ で得ることができる．$\sqrt{\frac{l}{\phi}} dist(\hat{s}_p^\phi, \hat{s}_q^\phi) > \sqrt{2l(1-\theta)}$ ならば，定義 3 より s_q は s_p と類似しないため，正確性を失うことなく s_p と s_q の正確な距離の計算を枝刈りできる．$\hat{s}_n$ が与えられたとき，$\forall s_p \in S \backslash S_n$ に対して，$\sqrt{\frac{l}{\phi}} dist(\hat{s}_n^\phi, \hat{s}_p^\phi)$ を計算することで，

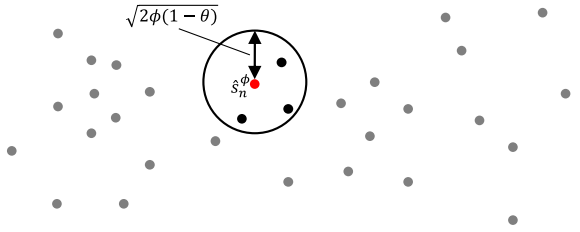


図 2 $score(s_n)$ の上界値を求める例
Fig. 2 Example of upper-bounding of $score(s_n)$.

$\hat{s}_n$ のスコアの上界値を取得できるが，これには $O(\phi(w-l))$ の時間計算量がかかる．ここで，s_n のスコアの上界値を求めるためには，$\sqrt{\frac{l}{\phi}}dist(\hat{s}_n^{\phi},\hat{s}_p^{\phi}) \leq \sqrt{2l(1-\theta)}$ を満たす s_p のみに注目すればよい．そこで，このような s_p を効率的に取得するために kd 木 [2] を用いる．kd 木は k 次元上の点を二分木で管理するデータ構造であり，データの削除，挿入および範囲検索を効率的に実行できる．

ウィンドウ内の PAA により変換されたサブシーケンスを kd 木で管理した場合，$\sqrt{\frac{l}{\phi}}dist(\hat{s}_n^{\phi},\hat{s}_p^{\phi}) \leq \sqrt{2l(1-\theta)}$ を満たす s_p は，$\hat{s}_n^{\phi}$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索により取得できる．このとき，以下の定理が成り立つ．

定理 1 新たに生成されるサブシーケンス s_n，距離の閾値 $\sqrt{2l(1-\theta)}$，および最新の l 個のサブシーケンスを除くウィンドウ内のすべてのサブシーケンスを管理する kd 木が与えられたとする．このとき，$\hat{s}_n^{\phi}$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索は，$\sqrt{\frac{l}{\phi}}dist(\hat{s}_n^{\phi},\hat{s}_p^{\phi}) \leq \sqrt{2l(1-\theta)}$ を満たす s_p の集合 S_n^{in} を返し，$|S_n^{in}| = m_n$ とすると，$m_n \geq score(s_n)$ である．

証明 補題 1 より，s_p は $\sqrt{\frac{l}{\phi}}dist(\hat{s}_n^{\phi},\hat{s}_p^{\phi}) \leq \sqrt{2l(1-\theta)}$ を満たす必要がある．この不等式から，$dist(\hat{s}_n^{\phi},\hat{s}_p^{\phi}) \leq \sqrt{2\phi(1-\theta)}$ が導かれる．次に，最新の l 個のサブシーケンスは s_n とトリビアルマッチであるため $score(s_n)$ には影響しない．以上より，定理 1 が成り立つ． □

例 1 図 2 は，2 次元に次元削減されたサブシーケンスを 2 次元上の点として表現している．$score(s_n)$ の上界値を求める場合，$\hat{s}_n^{\phi}$（赤い点）を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索を実行すると，範囲内に 3 個のサブシーケンスが存在するため，$m_n = 3$ となり，これが $score(s_n)$ の上界値となる．

定理 1 より，以下の系が成り立つ．

系 1 $\exists s \in S\backslash\{s_n\}$ に対して $score(s) \geq m_n$ ならば，s_n はモチーフになり得ない．つまり，正確な $score(s_n)$ の計算を枝刈りできる．

定理 1 を用いるため，SRMM では最新の l 個のサブシーケンスは kd 木で管理しない．ここで，n を kd 木で管理するデータの数，d をデータの次元数，m を範囲内に存在するデータの数としたとき，kd 木による範囲検索の最悪時間計算量は $O(n^{1-\frac{1}{d}}+m)$ である．つまり，スコアの上界値を取得する最悪時間計算量は $O((w-l)^{1-\frac{1}{\phi}}+m_n)$ であり，さらに，$(w-l)^{1-\frac{1}{\phi}}+m_n < w$ である．

4.2 スコアが減少するサブシーケンスの特定

ウィンドウがスライドした際，サブシーケンスが削除され，そのサブシーケンスと類似したサブシーケンスのスコアが減少する．スコアが減少するサブシーケンスを特定するため，削除されるサブシーケンスに対して範囲検索を実行することが考えられるが，これは効率的ではない．そこで，スコアが減少するサブシーケンスを効率的に特定するため，2 つのリスト SL（**S**imilar Subsequence **L**ist）および PL（**P**ossible Similar Subsequence **L**ist）を用いる．

定義 6（SL） s_p の SL SL_p は，サブシーケンスの識別子 q と $\rho(s_p,s_q)$ のタプルの集合であり，SL_p に含まれるあるサブシーケンス s_q は以下の条件を満たす．

$$s_q \in S\backslash S_p, \rho(s_p,s_q) \geq \theta \tag{9}$$

定義 7（PL） s_p の PL PL_p は，サブシーケンスの識別子 q の集合であり，PL_p に含まれるあるサブシーケンス s_q は以下の条件を満たす．

$$s_q \in S\backslash S_p, dist(\hat{s}_p^{\phi},\hat{s}_q^{\phi}) \leq \sqrt{2\phi(1-\theta)}, \langle q,\cdot\rangle \notin SL_p \tag{10}$$

つまり，範囲検索により s_p のスコアの上界値を求めるとき，$dist(\hat{s}_p^{\phi},\hat{s}_q^{\phi}) \leq \sqrt{2\phi(1-\theta)}$ を満たす q を PL_p に，p を PL_q に追加する．さらに，$\rho(s_p,s_q)$ を計算するとき，q (p) を PL_p (PL_q) から取り除き，$\rho(s_p,s_q) \geq \theta$ ならば q (p) を SL_p (SL_q) に追加する．ここで，2 つの補題が成り立つ．

補題 2 $|SL_p| + |PL_p| \geq score(s_p)$

補題 3 サブシーケンス s_e の削除によりスコアが減少する可能性のあるサブシーケンス s_q は，$q \in PL_e$ または $\langle q,\cdot\rangle \in SL_e$ を満たす．

補題 2 および 3 は，定義 6 および 7 から導かれる．また，補題 3 より，SL_p および PL_p の更新の時間計算量は $O(1)$ であるため，サブシーケンスの削除に対する時間計算量の合計は $O(|SL_e| + |PL_e|)$ である．

4.3 アルゴリズム

本節では SRMM の詳細を紹介する．ウィンドウがスライドした際，まず，削除されるサブシーケンス s_e に対する処理を行い，暫定のモチーフ s_{temp}^* を取得する．その後，新たに生成されるサブシーケンス s_n に対する処理を行い，s^* を取得する．

s_e に対する処理． アルゴリズム 1 は，SRMM の s_e に対する処理を行うアルゴリズムを示している．s_e が与えられたとき，kd 木から $\hat{s}_e^{\phi}$ を削除し，フラグ f を 0 とする（1 行）．次に，補題 3 より，$p \in PL_e$ または $\langle p,\cdot\rangle \in SL_e$ を満たすすべての SL_p および PL_p から e を削除する（2–9 行）．ただし，$score(s^*)$ が減少する場合，または $s^* = s_e$ である場合，

Algorithm 1: SRMM (expiration case)

Input: s_e: the expired subsequence
Output: s^*_{temp}: a temporal motif

```
1  Delete ŝ_e^φ from kd-tree, f ← 0
2  for ∀p ∈ SL_e do
3      SL_p ← SL_p\⟨e, ·⟩
4      if s_p = s* then
5          f ← 1
6  for ∀p ∈ PL_e do
7      PL_p ← PL_p\{e}
8  if s* = s_e then
9      f ← 1, s* ← ∅
10 s*_temp ← s*
11 if f = 1 then
12     for ∀s_p ∈ S such that |SL_p| + |PL_p| ≥ score(s*_temp)
       do
13         s*_temp ← Motif-Update(s_p, s*_temp)
```

Algorithm 2: SRMM (insertion case)

Input: s_n: the new subsequence, s^*_{temp}: a temporal motif
Output: s^*: the current motif

```
1  Compute ŝ_n^φ by PAA
2  Insert ŝ_{n−l}^φ to kd-tree
3  SL_n ← ∅
4  PL_n ← Range-Search(ŝ_n^φ, √(2φ(1 − θ)))
5  for ∀p ∈ PL_n do
6      if s_p = s*_temp then
7          Compute ρ(s_p, s_n)
8          if ρ(s_p, s_n) ≥ θ then
9              SL_p ← SL_p ∪ ⟨n, ρ(s_p, s_n)⟩,
                 SL_n ← SL_n ∪ ⟨p, ρ(s_p, s_n)⟩
10         PL_n ← PL_n\{p}
11     else
12         PL_p ← PL_p ∪ {n}
13         if |SL_p| + |PL_p| ≥ score(s*_temp) then
14             s*_temp ← Motif-Update(s_p, s*_temp)
15 if |SL_n| + |PL_n| ≥ score(s*_temp) then
16     s* ← Motif-Update(s_p, s*_temp)
17 else
18     s* = s*_temp
```

$f = 1$ とする．最後に，$f = 1$ の場合，モチーフが変化する可能性がある．補題 2 より，$|SL_p| + |PL_p| \geq score(s^*_{temp})$ を満たす場合，s_p がモチーフになる可能性があるため，Motif-Update(s_p, s^*_{temp}) を用いて，s_p の正確なスコアを計算し，s^*_{temp} を得る（13 行）．Motif-Update(s_p, s^*_{temp}) の詳細は後述する．

次に，s^*_{temp} が現在のモチーフであるか，または新たに生成されるサブシーケンス s_n がモチーフとなるかを確認する．

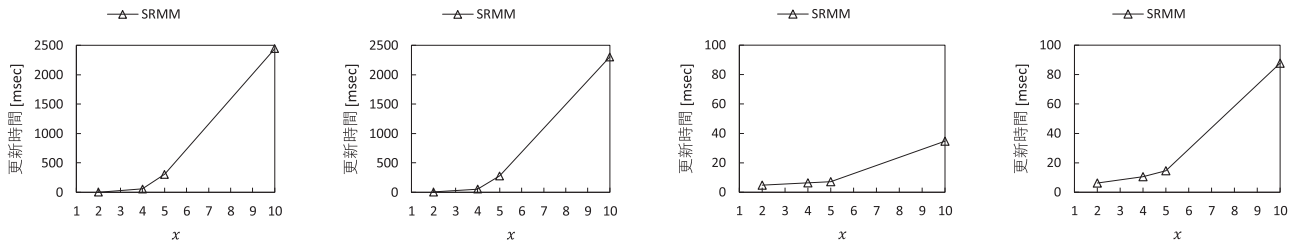
s_n に対する処理． アルゴリズム 2 は，SRMM の s_n に対する処理を行うアルゴリズムを示している．まず，PAA により $\hat{s}^{\phi}_n$ を計算し，$\hat{s}^{\phi}_{n-l}$ を kd 木に挿入する（1–2 行）．ここで，s_{n-l} は s_n とトリビアルマッチでない最新のサブシーケンスである（前述したとおり，最新の l 個のサブシーケンスは kd 木で管理しない）．次に，$SL_n = \varnothing$ とし，$\hat{s}^{\phi}_n$ を中心とする半径 $\sqrt{2\phi(1-\theta)}$ の範囲検索を実行することにより PL_n を取得する（3–4 行）．その後，$\forall p \in PL_n$ に対して，PL_p を更新する．$s_p = s^*_{temp}$ の場合，$\rho(s_p, s_n)$ を計算して，$score(s_p)$ を取得し，SL_p，SL_n，および PL_n を更新する（6–10 行）．一方，$s_p \neq s^*_{temp}$ の場合，PL_p を更新し，$|SL_p| + |PL_p| \geq score(s^*_{temp})$ を満たすかどうかを確認する．$|SL_p| + |PL_p| \geq score(s^*_{temp})$ を満たす場合，Motif-Update(s_p, s^*_{temp}) を実行し，必要ならばモチーフを更新する（14 行）．最後に，$|SL_n| + |PL_n| \geq score(s^*_{temp})$ を満たす場合，Motif-Update(s_n, s^*_{temp}) を実行し，s_n がモチーフになるかを確認する（15–16 行）．そうでない場合，s^*_{temp} が現在のモチーフであることが保証される（18 行）．

モチーフの更新． Motif-Update(s_p, s^*_{temp}) はモチーフの更新を実行するアルゴリズムである．$|SL_p| + |PL_p| \geq score(s^*_{temp})$ を満たす場合，s_p はモチーフになり得るため，s_p の正確なスコアを計算する必要がある．そのため，s_p と PL_p に含まれるサブシーケンスとのピアソン相関を計算することで，s_p の正確なスコアを計算する．s_p のスコアが s^*_{temp} のスコアより大きい場合，s_p がモチーフとして返却され，それ以外の場合，s^*_{temp} が返却される．

Top-k モチーフモニタリング． 1 章であげたアプリケーションにおいてスコアの大きい複数のサブシーケンスをモニタリングすることが考えられる．たとえば，扱う時系列データにノイズが多く含まれる場合，ノイズがモチーフとして発見され，有用なモチーフをモニタリングできない．そこで，上位 k 個のスコアを持つサブシーケンスを Top-k モチーフとし，Top-k モチーフをモニタリングできるように SRMM を拡張する．

アルゴリズム 1 では，いずれかの Top-k モチーフのスコアが減少する場合，または s_e が Top-k モチーフに含まれる場合，Top-k モチーフが変化する可能性があるため $f = 1$ とする．そして，Top-k モチーフの k 番目のスコアを $score_k$ とすると，$|SL_p| + |PL_p| \geq score_k$ である場合，Motif-Update(s_p, s^*_{temp}) を実行する．アルゴリズム 2 では，s_p が暫定の Top-k モチーフに含まれる場合，SL_p，SL_n，および PL_n を更新する（6–10 行）．また，$|SL_p| + |PL_p| \geq score_k$，および $|SL_n| + |PL_n| \geq score_k$ である場合，Motif-Update(s_p, s^*_{temp}) を実行する．

時間計算量． 前述したとおり，kd 木へのサブシーケンスの挿入および削除には $O(\log(w - l))$ かかる．アルゴリズ



(a) Google-CPU (b) Google-Memory (c) GreenHouseGas (d) RefrigerationDevices

図 3 ϕ の影響
Fig. 3 Impact of ϕ.

ム 1 には，$m_e = |SL_e| + |PL_e|$ としたとき，少なくとも $O(\log(w-l)+m_e)$ 時間かかる．アルゴリズム 2 には，最大で $O((w-l)^{1-\frac{1}{\phi}}+m_n)$ 時間かかる．ここで，m_n は RangeSearch$(\hat{s}_n^{\phi}, \sqrt{2\phi(1-\theta)})$ によって返されるサブシーケンスの数である．s_p のスコアを正確に計算する場合，$|PL_p|$ 回ピアソン相関の計算が必要となるため，$O(l|PL_p|)$ 時間かかる．よって，ウィンドウがスライドした際，正確なスコア計算が必要なサブシーケンスの集合を S' としたとき，SRMM の時間計算量は，$O((w-l)^{1-\frac{1}{\phi}}+m_e+m_n+\sum_{S'} l|PL_p|)$ となる．実践的には，$|S'|$ は非常に小さい値であり，本論文で行った実験では平均で $|S'| \leq 1$ であった．

5. 性能評価

本章では，SRMM およびベースラインアルゴリズムの性能評価のために行った実験の結果を紹介する．本論文では，モチーフの更新時間を評価する．

5.1 実験環境

本実験は，Windows10，3.40 GHz Core i7 および 16 GB RAM を搭載した計算機で行い，すべてのアルゴリズムを C++ で実装した．

データセット．以下の 4 つの実データを用いた．これらのデータセットを用いることにより，様々なデータ分布で機能し，l や w に対してスケーラブルであり，生成頻度がさらに高くなっても使えることを示す．

- Google-CPU [19]：Google のデータセンタの 1 秒ごとの CPU 使用率の時系列データ（長さ 133,902）
- Google-Memory [19]：Google のデータセンタの 1 秒ごとのメモリ使用率の時系列データ（長さ 133,269）
- GreenHouseGas [13]：カリフォルニア州の 2 時間ごとの温室効果ガスの排出量の時系列データ（長さ 100,062）
- RefrigerationDevices [*1]：冷蔵庫の 2 分ごとの消費電力の時系列データ（長さ 270,000）

パラメータ．本実験で用いたパラメータを**表 1** に示す．太字で表されている値はデフォルトの値である．また，ϕ は次節の準備実験により決定し，あるパラメータの影響を調べるとき，他のパラメータは固定する．

[*1] http://timeseriesclassification.com/index.php

表 1 パラメータ設定
Table 1 Configuration of parameters.

パラメータ	値
モチーフ長，l	50, **100**, 150, 200
ウィンドウサイズ，w[×1000]	5, **10**, 15, 20
閾値，θ	0.75, 0.8, 0.85, **0.9**, 0.95
k	**1**, 2, 4, 8, 16, 32

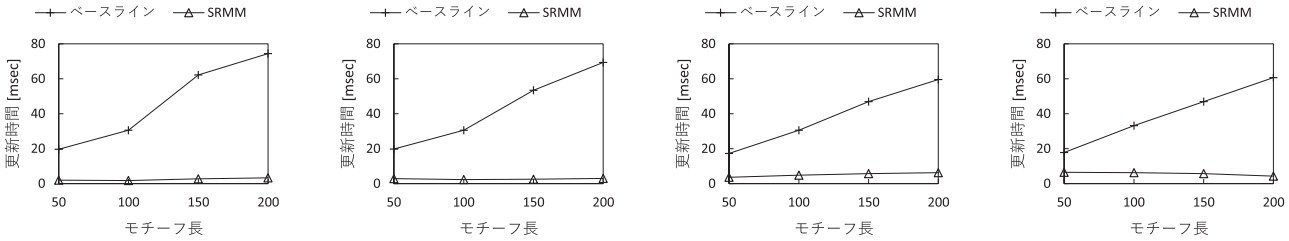
5.2 準備実験

最適な ϕ を決定するため，$\phi = \frac{l}{x}$ $(x = 2, 4, 5, 10)$ に対して SRMM の 1 スライドあたりの平均更新時間を評価する準備実験を行う．SRMM ではサブシーケンスの次元を PAA で圧縮するため，$x = 1$ の場合は評価しない．**図 3** に結果を示す．いずれのデータセットにおいても，$\phi = \frac{l}{2}$ のとき，更新時間が最も小さい．これは，x の増加にともない，得られるスコアの上界値が大きくなり，Motif-Update の実行回数が増加するからである．そのため，これ以降，$\phi = \frac{l}{2}$ とし評価実験を行う．

5.3 評価結果

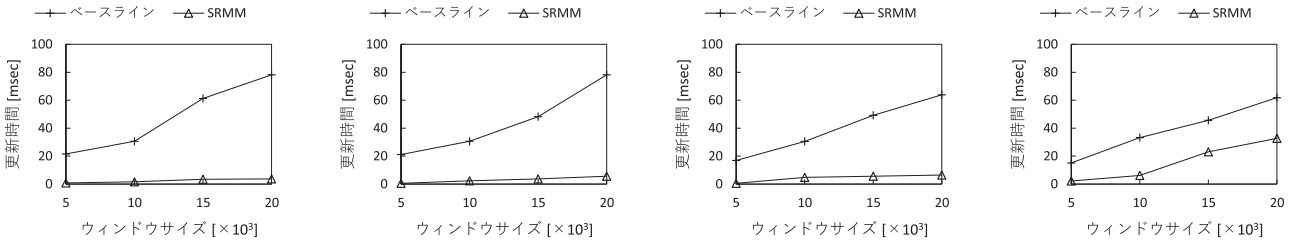
本節では，各アルゴリズムにおける 1 スライドあたりの平均更新時間の結果を示す．

l の影響．図 4 に l を変化させたときの結果を示す．ベースラインアルゴリズムにおける更新時間は，l の増加にともない，線形に増加する．これは，ベースラインアルゴリズムの時間計算量が $O((w-l)l)$ であるからである．一方，SRMM は l によらずほぼ一定値となる．ここで，l の増加にともない，ピアソン相関の計算時間は増加する．しかし，θ を固定した場合，l の増加にともない，2 つのサブシーケンス間の距離が大きくなる傾向があり，サブシーケンス間のピアソン相関は小さくなりやすいため，m_e および m_n が減少する．以上の理由から，SRMM は l によらず高速にモチーフを更新できる．SRMM はベースラインアルゴリズムよりも最大で 24.5 倍高速である．



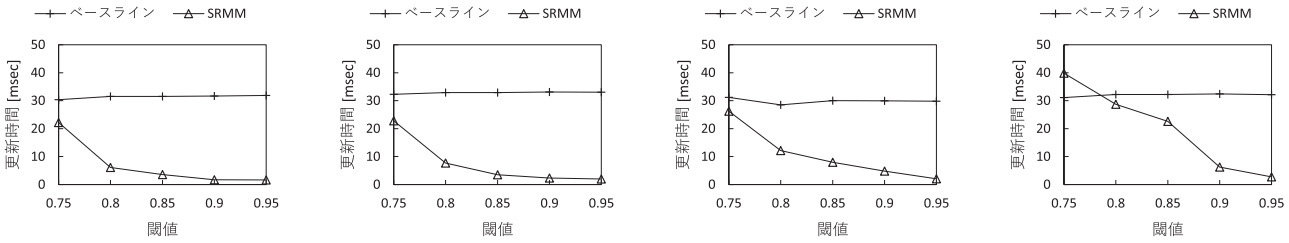
(a) Google-CPU (b) Google-Memory (c) GreenHouseGas (d) RefrigerationDevices

図 4 l の影響
Fig. 4 Impact of l.



(a) Google-CPU (b) Google-Memory (c) GreenHouseGas (d) RefrigerationDevices

図 5 w の影響
Fig. 5 Impact of w.



(a) Google-CPU (b) Google-Memory (c) GreenHouseGas (d) RefrigerationDevices

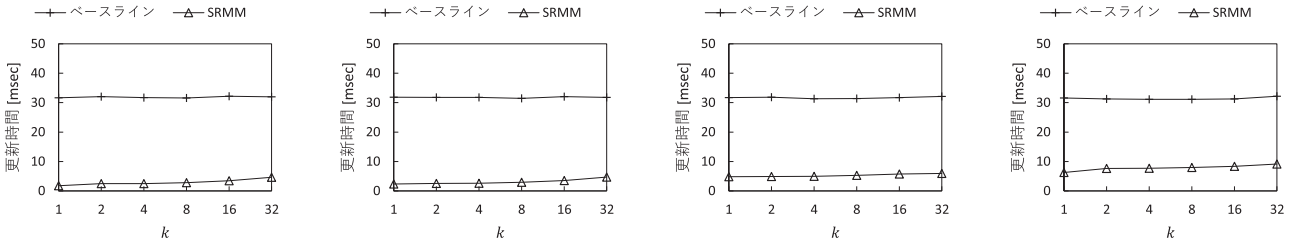
図 6 θ の影響
Fig. 6 Impact of θ.

w の影響．図 5 に w を変化させたときの結果を示す．図 5 に示すとおり，図 3 と同様の結果が得られていることが分かる．ベースラインアルゴリズムの時間計算量は w に対して線形であるため，この結果は妥当である．一方，l を変化させたときの結果と異なり，w の増加にともない SRMM の更新時間が増加する．これは，w の増加にともなって，各サブシーケンスのスコアが大きくなりやすく，m_e および m_n が大きくなりやすいためである．

θ の影響．図 6 に θ を変化させたときの結果を示す．ベースラインアルゴリズムでは，ウィンドウがスライドした際，新しく生成されるサブシーケンスおよび削除されるサブシーケンスとウィンドウ内のすべてのサブシーケンスとのピアソン相関を計算するため，更新時間は θ によらず一定である．一方で，SRMM は θ の増加にともなって更新時間は減少する．式 (4) より，θ が大きくなるにともなって距離の閾値は小さくなり，SRMM において範囲検索を実行する際，範囲内に存在するサブシーケンス数が減少する．つまり，m_e および m_n が減少するため，SRMM の更新時間は減少する．

図 6 (d) において $\theta = 0.75$ のとき，SRMM の更新時間はベースラインアルゴリズムよりも大きい．ここで，RefrigerationDevices では，θ が小さいとき，多くのサブシーケンスが互いに類似している．このような場合，正確なスコアの計算回数が削減できず，更新時間が大きくなる．しかし，多くのアプリケーションでは，大きく相関したサブシーケンスを発見することが求められる．図 5 で示すとおり，SRMM は θ が大きいとき，非常に高速にモチーフを更新できる．

k の影響．図 7 に k を変化させたときの結果を示す．ベースラインアルゴリズムは，ウィンドウ内のすべてのサブシーケンスのスコアを正確に保持するため，更新時間は k によらず一定である．一方，SRMM は k の増加にともな



(a) Google-CPU (b) Google-Memory (c) GreenHouseGas (d) RefrigerationDevices

図 7 k の影響
Fig. 7 Impact of k.

い，更新時間が若干増加する．これは，k の増加にともない，$score_k$ が小さくなり，Motif-Update を実行する回数が増加するためである．

6. 結論

近年注目されている IoT 機器では，ストリーミング時系列データを生成するため，時系列データをリアルタイムに解析することがより重要になっている．本論文では，レンジモチーフ（時系列データの中で最も多く現れるサブシーケンス）をモニタリングする問題に初めて取り組んだ．効率的にモチーフをモニタリングするため，SRMM を提案した．SRMM は PAA および kd 木を用いることにより，不要なスコアの計算を削減することができる．4 つの実データを用いた実験により，SRMM の有効性を確認した．

本論文では，1 次元の時系列データのみを対象とした．しかし，近年，機器は複数のセンサを持ち，多次元の時系列データを生成する．今後は，多次元のストリーミング時系列データのレンジモチーフを効率的にモニタリングする手法を検討している．

謝辞 本研究の一部は，基盤研究（A）（18H04095），基盤研究（B）（JP17KT0082），および若手研究（B）（JP16K16056）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] Begum, N. and Keogh, E.: Rare time series motif discovery from unbounded streams, *PVLDB*, Vol.8, No.2, pp.149–160 (2014).
[2] Bentley, J.L.: Multidimensional binary search trees used for associative searching, *Comm. ACM*, Vol.18, No.9, pp.509–517 (1975).
[3] Castro, N. and Azevedo, P.: Multiresolution motif discovery in time series, *SDM*, pp.665–676 (2010).
[4] Chen, Y., Nascimento, M.A., Ooi, B.C. and Tung, A.K.: Spade: On shape-based pattern detection in streaming time series, *ICDE*, pp.786–795 (2007).
[5] Chiu, B., Keogh, E. and Lonardi, S.: Probabilistic discovery of time series motifs, *KDD*, pp.493–498 (2003).
[6] Grabocka, J., Schilling, N. and Schmidt-Thieme, L.: Latent time-series motifs, *TKDD*, Vol.11, No.1, p.6 (2016).
[7] Kato, S., Amagata, D., Nishio, S. and Hara, T.: Monitoring Range Motif on Streaming Time-Series, *DEXA*, pp.251–266 (2018).
[8] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases, *KIS*, Vol.3, No.3, pp.263–286 (2001).
[9] Lam, H.T., Pham, N.D. and Calders, T.: Online discovery of top-k similar motifs in time series data, *SDM*, pp.1004–1015 (2011).
[10] Li, Y., Zou, L., Zhang, H. and Zhao, D.: Computing longest increasing subsequences over sequential data streams, *PVLDB*, Vol.10, No.3, pp.181–192 (2016).
[11] Li, Y., Yiu, M.L., Gong, Z., et al.: Quick-motif: An efficient and scalable framework for exact motif discovery, *ICDE*, pp.579–590 (2015).
[12] Lin, J., Keogh, E., Wei, L. and Lonardi, S.: Experiencing SAX: A novel symbolic representation of time series, *Data Mining and Knowledge Discovery*, Vol.15, No.2, pp.107–144 (2007).
[13] Lucas, D., Kwok, C.Y., Cameron-Smith, P., Graven, H., Bergmann, D., Guilderson, T., Weiss, R. and Keeling, R.: Designing optimal greenhouse gas observing networks that consider performance and cost, *Geoscientific Instrumentation, Methods and Data Systems*, Vol.4, No.1, p.121 (2015).
[14] Moshtaghi, M., Leckie, C. and Bezdek, J.C.: Online Clustering of Multivariate Time-series, *SDM*, pp.360–368 (2016).
[15] Mueen, A. and Keogh, E.: Online discovery and maintenance of time series motifs, *KDD*, pp.1089–1098 (2010).
[16] Mueen, A., Keogh, E., Zhu, Q., Cash, S. and Westover, B.: Exact discovery of time series motifs, *SDM*, pp.473–484 (2009).
[17] Nguyen, H.-L., Ng, W.-K. and Woon, Y.-K.: Closed motifs for streaming time series classification, *KIS*, Vol.41, No.1, pp.101–125 (2014).
[18] Patel, P., Keogh, E., Lin, J. and Lonardi, S.: Mining motifs in massive time series databases, *ICDM*, pp.370–377 (2002).
[19] Reiss, C., Wilkes, J. and Hellerstein, J.L.: Google cluster-usage traces: format+ schema, Google Inc., White Paper, pp.1–14 (2011).
[20] Shieh, J. and Keogh, E.: i SAX: Indexing and mining terabyte sized time series, *KDD*, pp.623–631 (2008).
[21] Yankov, D., Keogh, E., Medina, J., Chiu, B. and Zordan, V.: Detecting time series motifs under uniform scaling, *KDD*, pp.844–853 (2007).
[22] Yeh, C.-C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A. and Keogh, E.: Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and

shapelets, *ICDM*, pp.1317–1322 (2016).
[23] Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.-C.M., Funning, G., Mueen, A., Brisk, P. and Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins, *ICDM*, pp.739–748 (2016).

加藤 慎也 （学生会員）

2018 年大阪大学工学部電子情報工学科卒業．同大学大学院情報科学研究科博士前期課程在学中．時系列データにおけるデータ検索技術に関する研究に従事．

天方 大地 （正会員）

2012 年大阪大学工学部電子情報工学科卒業．2014 年同大学大学院情報科学研究科博士前期課程修了．2015 年同大学院情報科学研究科博士後期課程修了後，同年同大学院情報科学研究科マルチメディア工学専攻助教となり，現在に至る．情報科学博士．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．IEEE，ACM，日本データベース学会各会員．

西尾 俊哉 （学生会員）

2016 年大阪大学工学部電子情報工学科卒業．2018 年同大学大学院情報科学研究科博士前期課程修了後，同年同大学院情報科学研究科博士後期課程在学中．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．

原 隆浩 （正会員）

1995 年大阪大学工学部情報システム工学科卒業．1997 年同大学大学院工学研究科博士前期課程修了．同年同大学院工学研究科博士後期課程中退後，同大学院工学研究科助手，2004 年同大学院情報科学研究科准教授．2015 年より同大学院情報科学研究科教授となり，現在に至る．工学博士．2003 年本学会研究開発奨励賞受賞．2008 年，2009 年本学会論文賞，2015 年日本学術振興会賞受賞．ネットワーク環境におけるデータ管理技術に関する研究に従事．