区間 Min-Hash を用いた時系列データに対する近似最近傍探索

友田　涼太†　　古賀　久志†

† 電気通信大学大学院情報理工学研究科　〒 182–8585 東京都調布市調布ケ丘 1–5–1
E-mail: †t2231102@gl.cc.uec.ac.jp, ††koga@sd.is.uec.ac.jp

あらまし　Dynamic Time Warping (DTW) は時系列データ間の非類似度を測る指標である．DTW は，長さや周期が異なる時系列間で類似性を算出できる一方で計算時間が大きい．そこで，DTW を近似して類似検索を高速化する手法が出現している．従来手法である SSH では，時系列を部分時系列の集合として表し，集合に対するハッシュ (Min-Hash) を利用して高速類似検索を表現する．しかし，ハッシュ値が時系列内の位置と無関係であるため，ハッシュ値の一致と DTW との相関が低かった．本研究では，データベースの量子化をユークリッド距離ベースで適応的に計算し，時系列内でハッシュ値を生成する区間を指定することで，DTW が許容する位置ずれにも対応しつつハッシュ値の一致と DTW との相関を高める手法を提案する．

キーワード　時系列データ処理，データ構造・索引，確率的データ処理，効率性・スケーラビリティ

1　はじめに

近年，時系列データはロボット工学，医学，気象学，音声処理，物体検出など多くの分野で用いられるようになり，その量も増加している．その結果，類似時系列検索の重要性が高まっている．2 つの時系列データ間の類似度は，データ長や周期がなるためユークリッド距離では計測が困難である．これに対して，時系列データ間の非類似度を測る指標として Dynamic Time Warping（DTW）がある．DTW は，2 つの時系列データの距離を最小化するように時間軸を伸長させるので，時系列データ同士の長さや周期が異なっていても類似度を測れるという利点がある．しかし，DTW は 2 つの時系列データの各点間で距離を総当たりで求めるため，計算時間が大きい．したがって，DTW の高速計算は重要な課題である．

Kim ら [1] と Keogh ら [2] は，クエリ時系列データとの DTW 非類似度（以下では DTW 距離と呼ぶ）が小さい時系列データをデータベースから求める問題に対して，分枝限定法を用いた．Rakthanmanon ら [3] は，Kim らや Keogh らが提案した手法などを組み合わせた UCR suite という手法を提案した．同様に，Li ら [4] や Choi ら [5] も枝刈りベースの高速化手法を考案した．こうした枝刈りベースの厳密解法は DTW 距離を正確に計算できる一方で，長い時系列に対しては枝刈り効率が低下して実行速度が遅くなることが知られている．

そこで，DTW の近似値を算出することで高速化を目指す手法が出現している．近似類似検索を高速化する代表的なフレームワークに Locality-Sensitive Hashing (LSH) があるが，時系列や文字列といった順序情報を持つデータに対しては理論的な性能保証に裏付けされた実用的なハッシュ関数がないため，まだまだ発展途上な研究分野である．Yu ら [6] は多次元ベクトルを要素とする時系列を対象とする LSH を考案した．また，Astefanoaei [7] らは 2 次元空間上の，つまり 2 次元座標を要素とする軌跡に対して，空間内に配置されたディスクを通過する順番をスケッチとする手法を開発した．さらに，Luo ら [8] は，Sketch, Shingle & Hash (SSH) という手法を提案した．SSH は数値を要素とする時系列に対して，ハッシュベースで DTW を高速近似するアルゴリズムであり，厳密解法である UCR suite と比較して計算時間を大きく減らすことに成功した．しかし，SSH はランダムフィルタを使ってハッシュを生成するため，データベースに適応的でなく選択された乱数により検索効率が左右される．

また，ハッシュ値を時系列全体から生成するため，大きく位置が離れた部分時系列同士のマッチングや，複数のハッシュ値を比較する時に DTW が禁じる時間順序が逆転したマッチングが起こる．このように，ハッシュ値が一致したことと DTW との相関が低い．

本研究では，SSH を改善することを研究目的とする．データベースの量子化をユークリッド距離ベースで適応的に計算し，時系列内でハッシュ値を生成する区間を定めることで，DTW が許容する位置ずれにも対応しつつハッシュ値の一致と DTW との相関を高める手法を提案する．

本稿は以下の通りに構成される．2 節で背景知識である DTW と既存手法である SSH について述べる．3 節で SSH の抱える問題点を指摘し，4 節で問題点を改善した提案手法について述べる．5 節で提案手法の評価実験及び考察を述べる．最後に 6 節でまとめを述べる．

2　DTW

本節では，Dynamic Time Warping (DTW) について述べる．DTW は，2 つの時系列データ Q，C に対して距離を測る手法である．ここでは Q の長さを m，C の長さを n とする．ここで，q_i は Q の i 番目の要素を表す数値である．

$$Q = (q_1, q_2, ..., q_m) \tag{1}$$

$$C = (c_1, c_2, ..., c_n) \tag{2}$$

DTW はユークリッド距離と異なり時系列データの位置ずれを許してマッチングを行う．具体的には Q と C の各点 q_i $(1 \leqq i \leqq m)$ と c_j $(1 \leqq j \leqq n)$ の距離を総当たりで求めておいて，ある制約条件下で Q と C をアライメントさせた時の最小コストを Q と C の DTW 距離と定義する．アライメントの制約条件は 3 つある．

（1） q_1 は c_1 と，q_m は c_n と必ずマッチングすること

（2） 任意の 2 つのマッチングペアで時間順序が逆転しないこと

（3） Q と C の全要素が，マッチングに参加していること

以上 3 つの条件を満たすアライメントは，ワーピングパスで表現できる．ワーピングパスとはインデックスペアの集合である．

定義 1 (ワーピングパス) 時系列 Q, C 間のワーピングパス P は，長さを K として，以下のように表される．

$$P(Q, C) = ((i_1, j_1), (i_2, j_2), ..., (i_K, j_K)) \tag{3}$$

i_k と j_k はそれぞれ時系列データ Q, C のインデックスであり，以下の *3* 条件を満たす必要がある．

（*1*） 先頭と終端同士は必ずマッチングすること，すなわち $P_1 = (1, 1)$ かつ $P_K = (m, n)$。

（*2*） ワーピングパスのインデックスが広義単調増加すること．すなわち，

- $i_{k-1} \leqq i_k$
- $j_{k-1} \leqq j_k$

（*3*） 連続的であること，すなわち，自分より *1* つ前のインデックスとの差は *0* または *1* であること

- $i_{k-1} \leqq i_k \leqq i_{k-1} + 1$
- $j_{k-1} \leqq j_k \leqq j_{k-1} + 1$

ワーピングパス P のコスト $C(P)$ は式 (4) のようにマッチングした点間の距離の合計と定義される．

$$C(P) = \sqrt{\sum_{(i,j) \in P} d(q_i, c_j)^2} \tag{4}$$

定義 2 (Dynamic Time Warping（DTW）) *DTW*はコスト最小のワーピングパスのコストとして定義される．$A(Q, C)$ を Q, C 間の全てのワーピングパスの集合とすると，時系列 Q, C 間の *DTW* は式 *(5)* となる．

$$DTW(Q, C) = \min_{P \in A(Q,C)} \sqrt{\sum_{(i,j) \in P} d(q_i, c_j)^2} \tag{5}$$

コスト最小となるワーピングパスは動的計画法で求められる．DTW の計算コストは，時系列データの各点の距離を総当たりで求めるので $O(mn)$ となる．

2.1 ワーピング制約

現実のアプリケーションに DTW を適用する際には，ワーピングの範囲を制限し極端な伸縮によるマッチングを防ぐワーピング制約が導入される．代表的なワーピング制約に Sakoe-Chiba band がある．Sakoe-Chiba band はワーピングの範囲を $b(> 0)$ とすると，ワーピングパスに含まれる任意のインデックスペア (i_k, j_k) に対し $|i_k - j_k| \leqq b$ という制約条件を追加する．ワーピング制約を導入した場合，起こり得るインデックスペアの種類数が nb または mb 個に制限されるため，計算時間は $O(nb)$ または $O(mb)$ になる．これは $O(mn)$ よりは小さいが，データが長くなると計算時間が増大する．

2.2 SSH

Luo らによる SSH [8] は，クエリ時系列データ Q との DTW 距離が小さい時系列データを時系列データベース D からハッシュを使って高速に求める近似解法である．SSH では各時系列データ $X \in D$ に対してスケッチヒストグラム S_X と呼ばれるデータ構造を抽出し，S_X に対してハッシュ値を計算する．そして，クエリ Q とハッシュ値が一致したデータベース内の時系列データは DTW 距離が小さい可能性が高いとみなす．DTW 距離が小さい可能性が高い時系列データに対してのみ実際に DTW を計算することで，計算時間を削減する．

時系列データ $X = (x_1, x_2, ..., x_m)$ に対するハッシュ値は以下の 3 ステップで計算される．各ステップを以下に詳述する．

Step1　部分時系列に対するスケッチの抽出
Step2　スケッチ集合からのスケッチヒストグラムの構築
Step3　スケッチヒストグラムに対するハッシュ値の算出

2.2.1 Step1: 部分時系列に対するスケッチの抽出

Step1 ではまず，サイズ w のスライディングウィンドウを δ ずつ動かすことにより，X から長さ w の部分時系列

$$X_S^{(i)} = (x_{i*\delta}, x_{i*\delta+1}, ..., x_{i*\delta+w-1}) \tag{6}$$

を取り出す．i は部分時系列の番号であり，全部で $N = \frac{(m-w)}{\delta}$ 個の部分時系列が生成される．これらの部分時系列グループにフィルタベクトル r を適用してスケッチを生成する．r は確率的に生成される w 次元ベクトルである．平均 0，分散 σ^2 で確率密度関数が $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{x^2}{2})$ である 1 次元正規分布から w 個の確率変数を選択し，$r = (r_1, r_2, ..., r_w)$ とする．

スケッチを生成する具体的な手順は次のようになる．N 個の部分時系列 X_S^i $(1 \leqq i \leqq N)$ とフィルタベクトル r との間で内積 $r \cdot X_S^{(i)}$ を計算してその正負からビット $B_X^{(i)}$ を式 (7) により計算する．

$$B_X^{(i)} = \begin{cases} 1 & : r \cdot X_S^{(i)} \geqq 0 \\ 0 & : r \cdot X_S^{(i)} < 0 \end{cases} \tag{7}$$

この $B_X^{(i)}$ は，多次元ベクトルに対する cos 類似度に対する Locality-Sensitive Hashing (LSH) [9] そのものである．したがって，2 つの時系列 X, Y に対して $B_X^{(i)}$ と $B_Y^{(i)}$ が一致する確率は $P[B_X^{(i)} = B_Y^{(i)}] = 1 - \frac{\theta(X_S^{(i)}, Y_S^{(i)})}{\pi}$ となる．$\theta(X_S^{(i)}, Y_S^{(i)})$ は 2 つのベクトル $X_S^{(i)}$ と $Y_S^{(i)}$ がなす角である．

次に $B_X^{(i)}$ を並べた符号ストリーム $B_X = (B_X^{(1)}, B_X^{(2)}, ..., B_X^{(N)})$ を得る．図 1 に B_X の計算手順を図示する．

最終的に B_X に含まれる長さ n の部分系列（n-gram）をス

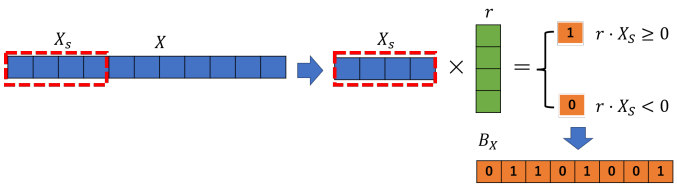

図 1　B_X の抽出

ケッチとする．n-gram は X に含まれる長さ $w+(n-1)\delta$ の部分時系列を n 次元バイナリベクトルとして表したものと見なせる．

2.2.2　Step2: スケッチ集合からのヒストグラムの構築

ここでは，抽出したスケッチ集合（n-grams）から，スケッチヒストグラム $S_X = \{(S_i, w_i)\}$ を構築する．w_i は，時系列に存在するビットパターン S_i の頻度である．ここで，ビットパターンの種類数は最大で 2^n となる．図 2 は，$n=2$ のときの 2-gram からヒストグラムを構築した例である．

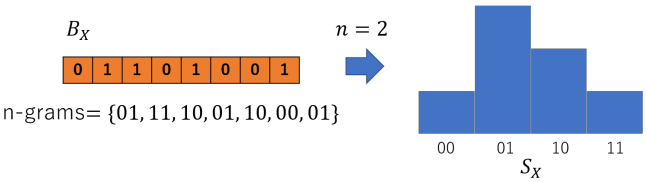

図 2　$n=2$ のときの 2-grams のヒストグラム

SSH において S_X は時系列データ X を長さ $w+(n-1)\delta$ の部分時系列の多重集合として表したものとなる．

2.2.3　Step3: ヒストグラムに対するハッシュ値の算出

Step3 ではヒストグラム S_X に対して，頻度 w_i を重みとする重み付き Min Hash を計算し，ハッシュ値とする．重み付き Min Hash は，Weighted Jaccard similarity と等しい衝突確率を持つハッシュ関数である．このハッシュ関数を $mh(X)$ とする．重み付き Min Hash の計算には Consistent Weighted Sampling [10] を利用する．

2.2.4　類 似 検 索

時系列データベース $D = \{X_i | 1 \leqq i \leqq |D|\}$ からクエリ Q に対する DTW 距離に基くハッシュを利用した top-k 検索の手順を述べる．

前処理として，全ての $X_i \in D$ に対して l 個のハッシュ関数を使いハッシュ値 $mh_1(X_i), ..., mh_l(X_i)$ を計算する．LSH による検索のため，これらをインデックスとし，l 個のハッシュテーブルに登録する．クエリ Q を処理する時には，Q に対して l 個のハッシュ値 $mh_1(Q), ..., mh_l(Q)$ を計算し，対応する l 個のバケットを探す．そして，バケットに含まれる候補時系列データの中から DTW を計算して top-k を出力する．

3　SSH の問題点

まず，SSH の 1 つ目の問題点として Step 1 で生成される n-gram と DTW の相関が弱いことが挙げられる．n-gram を

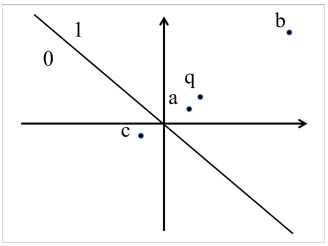

図 3　ビットが不一致となる 2 つの部分時系列の距離が近い事例

構成する 1 ビットが一致した，つまり $B_X^{(i)} = B_Y^{(i)}$ となったということは，部分時系列 $X_S^{(i)}$，$Y_S^{(i)}$ をベクトルと見なした時に向きが近いということである．しかし，これは $X_S^{(i)}$ と $Y_S^{(i)}$ とが空間的に近いことを意味しない．同様に，$B_X^{(i)} \neq B_Y^{(i)}$ となっても $X_S^{(i)}$ と $Y_S^{(i)}$ が空間的に離れているとは限らない．図 3 に，部分時系列 q とビットが一致しない部分時系列との距離が近くなってしまう例を示す．この図では a, b, q がフィルタベクトルが定める分離面の同じ側に存在し，c のみ反対側に存在する．すなわち，a, b のビット値はクエリ q と一致するが，c のビット値は q と異なる．しかし，c は b よりクエリ q との距離が明らかに近い．DTW は要素間の距離から求められるので，本例は，SSH による n-gram と DTW との相関が弱いことを示している．

また，SSH では，Step1 のフィルタベクトル r をデータベース D とは無関係に確率的に生成する．つまり，r は D に対して適応的に選択されていない．この結果，フィルタベクトルの値によっては内積値 $r \cdot X_S^{(i)}$ の正負が偏り，ほとんどの時系列に対してビット値 $B_X^{(i)}$ が等しくなってしまう危険性がある．この場合 n-gram が部分時系列を識別する能力が低下する．

SSH の 2 つ目の問題点は，top-k 検索でハッシュ値を照合する際に部分時系列が持つ時刻情報を完全に捨てていることである．この性質のため，ハッシュ値を時系列全体から生成する SSH では，Sakoe-Chiba band 等で指定されたワーピング制約を満たさない部分時系列がマッチングする可能性がある．さらに，複数個のハッシュ値を照合する時に図 4 に示すような DTW で禁じられている時間順序が逆転したマッチングが起こる可能性がある．この図は 2 つの時系列データ X, Y 間に対して，2 個のハッシュ関数 mh_1, mh_2 のハッシュ値が一致する状況を描いているが，mh_1 と mh_2 のマッチングする部分時系列ペアが交差してしまっている．DTW ではワーピングパスは単調でなければならないのでこうしたマッチングを許さないが，SSH では時間順序が逆転したマッチングを許容しており，DTW に忠実でない．

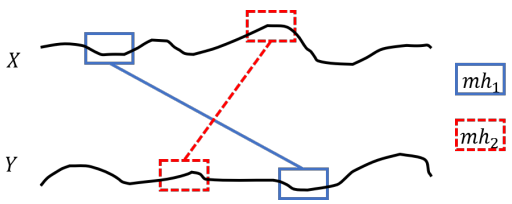

図 4　時間順序が逆転したマッチング例

4 提案手法

前節では

（1） n-gram が (a) 部分時系列（ベクトル）の向きに基づいており，時系列データの距離から計算される DTW との相関が弱いこと，及び，(b) ランダムに選択されたフィルタベクトルがデータベース D に適応的でないこと

（2） ハッシュ値を時系列全体から 1 つ生成するため，大きく離れた位置同士でのマッチングや，複数のハッシュ値を比較する時に時間順序が逆転したマッチングが起こり，DTW に忠実でないこと

という SSH の問題点を 2 つ指摘した．本稿ではこれらを改善する Bag-of-sketch, Section Hash (BSecH) という手法を提案する．1 つ目の問題点については，部分時系列をユークリッド距離ベースで D に対して適応的に量子化することで改善する．2 つ目の問題点については，時系列内でハッシュ値を生成する区間を定める区間 Min Hash を提案し，ハッシュ値の一致と DTW との相関を高めることで改良する．以下では，それぞれの手法について説明する．

4.1 ユークリッド距離ベースの部分時系列の量子化

本手法は，画像処理で使われてきた Bag-of-Visual-Words (BoVW) [11] から発想した．BoVW では画像データベース I から抽出された局所特徴ベクトルをクラスタリングし I に適応的に量子化する．そして，1 枚の画像は Visual Word のヒストグラムとして特徴表現される．

我々の手法は，時系列データ X から切り出された長さ w の部分時系列を w 次元の局所特徴ベクトルと見なしてクラスタリングすることにより代表 Word を計算し，X を代表 Word のヒストグラムとして表現する．

具体的には，ウィンドウサイズ w のスライディングウィンドウをステップサイズ δ ずつ動かしながら，X より部分時系列 $X_S^{(i)} = (x_{i*\delta}, x_{i*\delta+1}, ..., x_{i*\delta+w-1})$ を切り取る．部分時系列の長さは w なので，$X_S^{(i)}$ を w 次元ベクトルと見なす．そして，部分時系列の集合 $\{X_S^{(1)}, X_S^{(2)}, \ldots, X_S^{(N)}\}$ を X の特徴点集合とする．この時点では SSH と全く同じ部分時系列が切り出されることに注意されたい．

この過程を全時系列データ $\forall X \in D$ に対して繰り返し，D 全体に対する特徴点集合 Ω を得る．代表 word は Ω を k-means 法でクラスタリングすることで決定される．クラスタ数を k，i 番目のクラスタを C_i とすると，k-means 法では C_i の代表 Word を C_i に所属する特徴点の重心と定める．したがって，代表 Word の総数はクラスタ数と同じ k になる．また，各特徴点は最もユークリッド距離が近い代表 Word (のクラスタ ID) に量子化される．部分時系列は SSH では n-gram に変換されていたが，BSecH ではクラスタ ID に量子化される．したがって，SSH における n-gram は提案手法におけるクラスタ ID に相当する．

時系列データ X の部分時系列 $X_S^{(i)}$ が所属するクラスタを $C_X^{(i)}$ とすると，X は所属クラスタの集合 $\{C_X^{(1)}, C_X^{(2)}, ..., C_X^{(N)}\}$ として表現できる．この集合に対するヒストグラムを，X に対するスケッチヒストグラム S_X と定める．

部分時系列の代表 Word への量子化は，部分時系列と代表 Word のユークリッド距離に基づいている．したがって，同じ代表 Word に量子化された部分時系列同士は，距離が近い可能性が高い．本手法は，代表 Word をクラスタリングによって決定している．つまり，代表 Word は D のデータ分布に適応的に決定されており，各時系列データ X のヒストグラム S_X も D のデータ分布に適応的になる．

4.2 区間 Min Hash

区間 Min Hash は，時系列データ X の部分区間に対してハッシュ値を生成する．具体的には X のスケッチ $\{C_X^{(1)}, C_X^{(2)}, ..., C_X^{(N)}\}$ に対して，始点 u と幅 d を選択して部分スケッチ $\{C_X^{(u)}, C_X^{(u+1)}, ..., C_X^{(u+d-1)}\}$ を取り出す．部分スケッチに対して重み付き Min Hash を計算する．このハッシュ値を $mh(X)$ とする．ハッシュを生成する区間を定めることで，一定以上離れた部分時系列同士のマッチングや，時間順序が逆転したマッチングを防止する．時系列データ X, Y 間において，ハッシュ値が一致したとき，すなわち $mh(X) = mh(Y)$ のとき，対応する部分時系列のマッチングのずれは必ず d 以下になる．また，2 つの異なる区間 $[u_1, u_1 + d - 1], [u_2, u_2 + d - 1]$ $(u_1 < u_2)$ に対するハッシュ値が同時に一致したとき，部分スケッチの先頭と最後尾の位置は $u_1 + d - 1 < u_2$ となるので，時間順序が逆転したマッチングが起こらない．

X に対してハッシュ値を生成するステップは以下の通りである．

（1） X のスケッチを t 個の部分区間に等間隔に分割する．

（2） 各部分区間の中からランダムに始点 u を選択する．

（3） 始点 u から区間幅 d の範囲の部分スケッチを取り出す．

（4） 各部分スケッチに対して重み付き Min Hash を計算する．

図 5 は，区間 Min Hash におけるハッシュ値の生成を図示したものである．X のスケッチ C_X を縦の黒線で t 個の部分区間に分割している．赤の枠線が区間幅 d の部分スケッチであり，各部分スケッチからハッシュ値 $hv_1, hv_2, ..., hv_t$ を生成する．各区

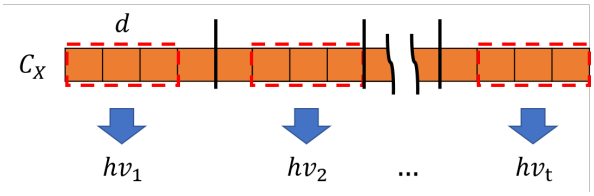

図 5 区間 Min Hash のハッシュ値の生成

間から 1 つのハッシュ値を生成するため，t 個のハッシュ値が生成される．区間幅 d と分割数 t が区間 Min Hash のパラメータとなる．また，区間の始点 u はすべての時系列データに対して共通であり，時系列データ毎に値を変えてはいない．

実装では SSH と同様に l 個のハッシュ関数 $mh_1, mh_2, \cdots, mh_l$

を用意する．ハッシュ関数 mh_i で j 番目の部分区間で取り出される部分スケッチを X^{ij} とする．ハッシュ関数によって取り出す部分区間の位置は異なるため，部分スケッチの位置は全部で tl 箇所になる．1 個のハッシュ関数に対して t 個のハッシュ値が生成されるため，X に対して合計で tl 個のハッシュ値が生成される．

4.3 k-NN 探索アルゴリズム

ハッシュを利用した時系列データベース $D = \{X_i | 1 \leqq i \leqq |D|\}$ からクエリ Q に対する DTW 距離に基く top-k 検索の手順を述べる．

前処理として，全ての $X_i \in D$ に対して l 個のハッシュ関数 $mh_1, mh_2, ..., mh_l$ を使い l 個のハッシュ関数から t 個ずつハッシュ値を計算し，合計で tl 個のハッシュ値を計算する．LSH による検索のため，これらをインデックスとし，tl 個のハッシュテーブル T に登録する．クエリ Q に対する DTW 距離に基く top-k 検索のアルゴリズムを Algorithm 1 に示す．クエリ Q を処理する時には，Q に対して tl 個のハッシュ値を計算し，対応する tl 個のバケットを探す．ハッシュ値を照合する際には，ハッシュ値だけではなく区間も一致してはじめてハッシュ値が一致したと判定する．そして，クエリと少なくとも 1 個以上のハッシュ値が一致した候補時系列に対して DTW を計算して top-k を出力する．この際に，候補時系列をハッシュの一致した数の降順に並び替え，ハッシュの一致数が多い時系列から DTW を計算する．ハッシュの一致数が多いほど DTW の値が小さくなることが期待できるので，ハッシュの一致数の降順に DTW を計算することで枝刈りにおける DTW の lower bound を早期に減少させることが出来る．

Algorithm 1 Query Process

候補集合 R を null に初期化
for $i = 1$ to l **do**
  for $j = 1$ to t **do**
    クエリ Q のスケッチから区間 Min Hash で j 番目の区間のハッシュ値 $mh_i(Q^{ij})$ を計算
    ハッシュテーブル T_{ij} のバケツ $mh_i(Q^{ij})$ の時系列を R に追加
  end for
end for
return R から UCR suite を用いて top-k 探索

5 実験評価

実データを用いて，従来手法である SSH と提案手法である BSecH を実験により性能を比較した．また，提案手法において，問題による最適なパラメータについての実験を行った．

全ての実験は，以下に示す環境で行った．

- Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz × 16
- 16GB メモリ
- Ubuntu20.04

また，実装は全て C++で行った．SSH は，著者が公開している C++プログラム [12] を使用して実装をした．提案手法の実装において，k-means の実装は OpenCV を用いて行った．

データセットは，SSH の論文で用いられていた実データ (ECG データセット) を使用した．ECG データセットは，22 時間 23 分の ECG データ（20,140,000 データポイント）で構成されている．ECG データに対して，z-score normalization を行い，1024 データポイントごとに区切ることで，それぞれ長さ 1024 の時系列データにした．時系列データは 4000 個使用した．z-score normalization は，以下の式よりデータを正規化した．

$$x_{new} = \frac{(x - \bar{x})}{\sigma(x)} \tag{8}$$

$\bar{x}$ は時系列データの平均，$\sigma(x)$ はデータの標準偏差である．クエリ時系列データは，ECG データセットから長さ 1024 の時系列データを切り出して作成した．

5.1 SSH との性能比較

本節では，提案手法 BSecH と従来手法 SSH との性能を比較する．ECG データセットに対して，それぞれのアルゴリズムで top-20 近傍探索を行った．各実験において，クエリ時系列を変えて 20 回，乱数 seed を変えて 5 回実行した時の平均をとった．評価指標として，D の全データに対して DTW 計算を行った際の top-20 厳密解と top-20 近傍探索により出力された解が一致した割合を再現率とした．

SSH の各パラメータは，SSH の論文で ECG データセットに対して最適だとしている $w = 80, \delta = 3, n = 15$ とした．BSecH では，$w = 80, \delta = 1, k = 1024, t = 10, d = 20$ とした．ワーピング制約は $b = 10$ とした．このパラメータ設定を BSecH のデフォルトパラメータとする．また，SSH，BSecH とも DTW 計算では，UCR suite による枝刈りを行った．

ハッシュ値の一致と DTW 距離との相関性を測る目的で，ハッシュ値が一致したデータに対する DTW の計算回数と再現率の関係を調べた．結果を図 6 に示す．横軸は DTW 距離の計算回数であり，ハッシュ値の一致回数が大きい順に何回 DTW 距離を計算したのか表す．計算回数が少なく再現率が高いほどハッシュ値が DTW 距離に忠実ということになる．BSecH をフルに使用したもの，BoVW に基づく量子化と重み付き Min Hash を使用したもの (BoVW)，フィルタベクトルによる量子化と区間 Min Hash を使用したもの (区間 Min Hash)，SSH について実行した．このとき，乱数のばらつきによる性能への影響を抑えるため，ハッシュ関数の数を増やし BSecH と区間 Min Hash を $l = 20$ とし，BoVW と SSH は $l = 60$ とした．BoVW と区間 Min Hash は，単体でも SSH より常に高い再現率となった．BSecH は，これらよりさらに高い再現率を記録し，組み合わせることでより性能が高まることが示された．また，SSH は DTW 計算した時系列データ数にほぼ比例して再現率が上がっているのに対して，BSecH では SSH の場合より早期に再現率が上がっている．これは，区間 Min Hash のハッシュの一致と DTW の間に強い相関があることを示している．

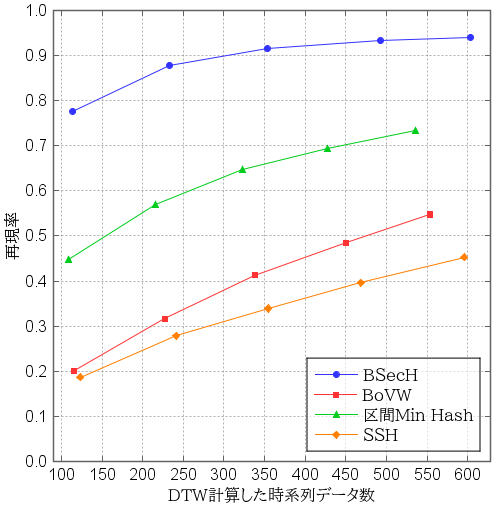

図 6　BSecH と SSH の DTW 計算した時系列データ数と再現率

次に，実行時間による評価実験を行った．BSecH と SSH における top-20 近傍探索を行ったときの実行時間とそのときの再現率の関係を図 7 に示す．実行時間による評価では，BSecH のハッシュ関数の数を $l = 10$ とし，SSH は，SSH の論文で最適としている $l = 20$ にして実行時間とそのときの再現率を調べた．横軸が実行時間であり，縦軸が再現率である．BSecH は SSH に比べて実行時間が短く，全体的に再現率も高い値を記録した．平均の実行時間は，BSecH は 22.26ms，SSH は 47.86ms であり，BSecH は SSH と比較して約 53% 短くなった．また，BSecH は SSH より実行時間のばらつきが少ない．実行時間は DTW の計算量に大きく影響される．SSH ではランダムなフィルタベクトルでスケッチを作成しているため，ハッシュの一致する時系列データ数が不安定であり，実行ごとの候補時系列データ数のばらつきが大きく実行時間も不安定である．一方で，BSecH では，BoVW による安定かつ適切なスケッチの作成により実行時間のばらつきも少ない．

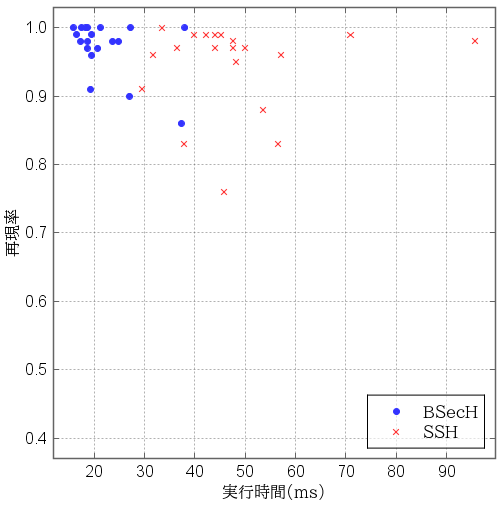

図 7　BSecH と SSH の実行時間 (ms) と再現率

5.2　パラメータの効果

本節では，BSecH でのパラメータが検索性能にどう影響を与えるのか調べるため，区間幅 d とウィンドウサイズ w を変化させる実験を行った．

5.2.1　区 間 幅 d

BSecH における，区間幅の大きさによる再現率への影響を調べた．Sakoe-Chiba band $b = 10$ としたとき区間幅 d を 10 から 40 まで変化させたときの再現率を図 8 に示す．再現率は $d = 10$ で低く，$d = 20$ で最も高くなっており，更に d を増やすと再現率は低くなった．このことから，Sakoe-Chiba band に対する BSecH の性能を最大に引き出すことができるパラメータが存在することが考えられる．

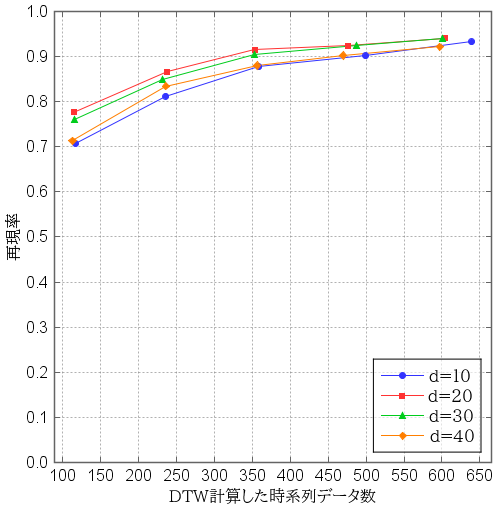

図 8　BSecH における Sakoe-Chiba band $b = 10$ の場合の区間幅 d を変化させたときの再現率

次に，BSecH における，Sakoe-Chiba band $b = 20, 50$ としたとき区間幅 d を変化させたときの再現率をそれぞれ図 9 及び図 10 に示す．異なる b のそれぞれの場合において，高い再現率を記録する d の値があり，適したパラメータが存在することが分かった．いずれの場合も，b より 10 大きい区間幅の再現率が高い結果となった．

5.2.2　ウィンドウサイズ w

BSecH における，ウィンドウサイズの大きさによる再現率への影響を調べた．Sakoe-Chiba band $b = 10$ としたときウィンドウサイズ w を 40 から 100 まで変化させたときの再現率をそれぞれ図 11 に示す．ウィンドウサイズが大きいほど再現率は高くなり，$w = 80$ でピークになり，$w = 100$ はほぼ同じ高さとなった．ウィンドウサイズ w は時系列データから切り出す部分時系列の長さである．BSecH においてハッシュ値が一致したという事は，部分時系列ペアがユークリッド距離が小さく似ていたために同一クラスタに所属したということである．そして，BSecH はハッシュ値が一致する時系列データペアほど類似性が高いと判定する．したがって，図 11 の実験結果は，部分時系列の類似性に着目して時系列データの類似性判定を行うアプローチにおいて，部分時系列の適切な長さが存在することを

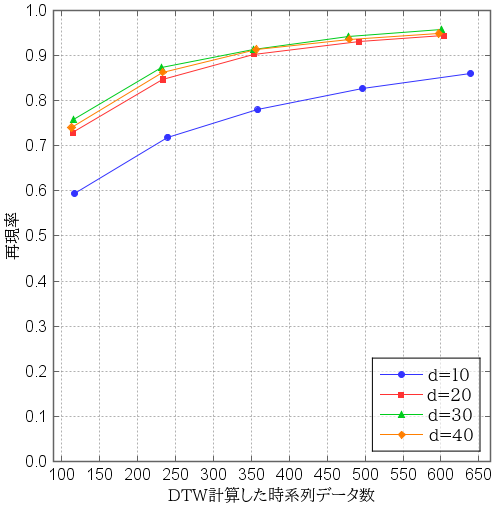

図 9　BSecH における Sakoe-Chiba band $b = 20$ の場合の区間幅 d を変化させたときの再現率

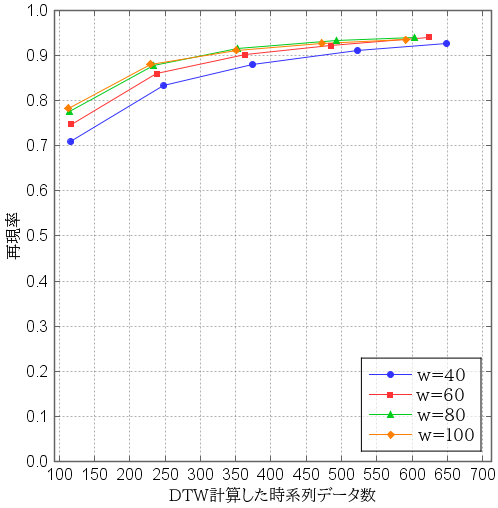

図 11　BSecH における Sakoe-Chiba band $b = 10$ の場合のウィンドウサイズ w を変化させたときの再現率

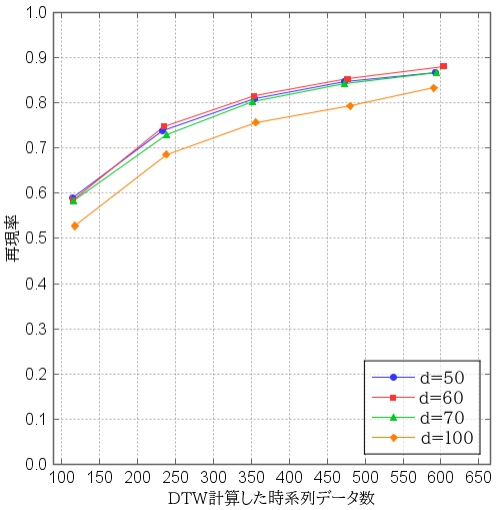

図 10　BSecH における Sakoe-Chiba band $b = 50$ の場合の区間幅 d を変化させたときの再現率

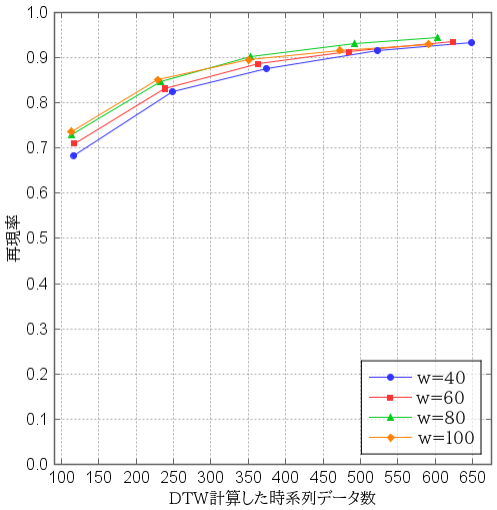

図 12　BSecH における Sakoe-Chiba band $b = 20$ の場合のウィンドウサイズ w を変化させたときの再現率

示している．

次に，Sakoe-Chiba band $b = 20, 50$ としたときウィンドウサイズ w を変化させたときの再現率をそれぞれ図 12 及び図 13 に示す．Sakoe-Chiba band の大きさを変えたいずれの場合も，$w = 40$ の w が小さい場合では再現率が低く，$W = 80, 100$ で高くなった．この結果から，BsecH における適切な部分時系列長は Sakoe-Chiba band のパラメータに依存しないと言える．

6　結　　論

本論文では，データベースから，クエリ時系列データとの DTW 距離が小さい時系列データを求める問題に対しての，ハッシュを用いた近似高速計算手法である SSH について，データの特徴表現であるスケッチと類似検索アルゴリズムであるハッシュの 2 つの問題点を指摘した．1 つは n-gram が時系列データの距離から計算される DTW との相関が弱いこと，及びランダムに選択されたフィルタベクトルがデータベースに適応的でないことであり，もう 1 つはハッシュ値を時系列全体から 1 つ生成するため，大きく離れた位置同士でのマッチングや，複数のハッシュ値を比較する時に時間順序が逆転したマッチングが起こり，DTW に忠実でないことである．そして，これら 2 つの問題点を改善した計算手法である BSecH を提案した．新たなスケッチ生成方法では，ユークリッド距離ベースで部分時系列を量子化し，DTW との相関が強いスケッチを生成できるようにした．新たなハッシュ計算方法である区間 Min Hash では，スケッチから部分的に区間を定めて取り出し，ハッシュ値を生成することで，DTW が許容する位置ずれに対応しつつハッシュ値の一致と DTW との相関が強いハッシュ値を生成できるようにした．実データを用いた実験により，BSecH が SSH より高い精度を得られることを示した．また，区間幅を DTW のワーピング制約のサイズより少し大きくすることで，高い性能を得られることも分かった．今後は，Sakoe-Chiba band の大きさによらずパラメータの調節が必要としない手法の検討を行う．

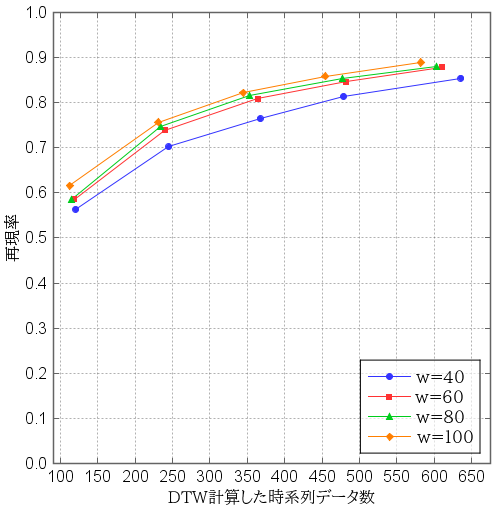

図 13　BSecH における Sakoe-Chiba band $b = 50$ の場合のウィンドウサイズ w を変化させたときの再現率

謝　　辞

本研究は JSPS 科研費 JP21K11901 の助成を受けたものである．

文　　献

[1] Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the 17th International Conference on Data Engineering*, pp. 607–614, USA, 2001. IEEE Computer Society.

[2] Eamonn Keogh, Li Wei, Xiaopeng Xi, Michail Vlachos, Sang-Hee Lee, and Pavlos Protopapas. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures. *The VLDB Journal*, Vol. 18, No. 3, pp. 611–630, oct 2008.

[3] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 26–270, 2012.

[4] Zhengxin Li, Jiansheng Guo, Hailin Li, Tao Wu, Sheng Mao, and Feiping Nie. Speed up similarity search of time series under dynamic time warping. *IEEE Access*, Vol. 7, pp. 163644–163653, 2019.

[5] Wonyoung Choi, Jaechan Cho, Seongjoo Lee, and Yunho Jung. Fast constrained dynamic time warping for similarity measure of time series data. *IEEE Access*, Vol. 8, pp. 222841–222858, 2020.

[6] Chenyun Yu, Lintong Luo, Leanne Lai-Hang Chan, Thanawin Rakthanmanon, and Sarana Nutanong. A fast lsh-based similarity search method for multivariate time series. *Information Sciences*, Vol. 476, pp. 337–356, 2019.

[7] Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, pp. 279–288, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Chen Luo and Anshumali Shrivastava. SSH (sketch, shingle, & hash) for indexing massive-scale time series. In *Proceedings of the NIPS 2016 Time Series Workshop, co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, Vol. 55, pp. 38–58, 2016.

[9] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 380–388, New York, NY, USA, 2002. Association for Computing Machinery.

[10] Mark Manasse, Frank McSherry, and Kunal Talwar. Consistent weighted sampling. Technical Report MSR-TR-2010-73, June 2010.

[11] Fuentes-Pineda Gibran, Koga Hisashi and Watanabe Toshinori. Scalable Object Discovery: A Hash-Based Approach to Clustering Co-occurring Visual Words. IEICE Transactions. 94-D. 2024-2035. 10.1587/transinf.E94.D.2024, 2011.

[12] Chen Luo. ssh, 2017. `https://github.com/rackingroll/ssh`.