Alignment of Time Series for Subsequence-to-Subsequence Time Series Matching

Vineetha Bettaiah and Heggere S. Ranganath

Abstract The success of time series data mining applications, such as query by content, clustering, and classification, is greatly determined by the performance of the algorithm used for the determination of similarity between two time series. The previous research on time series matching has mainly focused on whole sequence matching and to limited extent on sequence-to-subsequence matching. Relatively, very little work has been done on subsequence-to-subsequence matching, where two time series are considered similar if they contain similar subsequences or patterns in the same time order. This paper presents an effective approach capable of handling whole sequence, sequence-to-subsequence and subsequence-to-subsequence matching. The proposed approach derives its strength from the novel two stage segmentation algorithm, which facilitates the alignment of the two time series by retaining perceptually important points of the two time series as break points.

Keywords Data mining · Dimensionality reduction · Piecewise linear representation · Time series representation

1 Introduction

Many areas of science, engineering and business are generating, archiving and processing vast amounts of time series data. Mathematically, a time series T = {T[1], T[2], …, T[n]} is a sequence of n real numbers in the increasing order

V. Bettaiah (✉) · H.S. Ranganath
Computer Science Department, The University of Alabama in Huntsville,
301 Sparkman Dr NW, Huntsville, AL 35803, USA
e-mail: vineetha.bettaiah@gmail.com

H.S. Ranganath
e-mail: ranganat@uah.edu

of time, where each value has a time stamp. The ability to match two time series, T1 of length m, and T2 of length n, to determine their similarity, is a fundamental and critical step in most time series data mining applications including query by content, clustering and classification [1]. Time series T1 and T2 may differ in length, time scale, amplitude scale, time shift, and amplitude shift. There may be considerable distortion due to time warp and noise, and the elements of T1 and T2 may not align. In such cases, the matching should be established on general shapes and local trends of T1 and T2. There are three time series matching scenarios.

1. Sequence-to-sequence matching
2. Sequence-to-subsequence matching
3. Subsequence-to-subsequence matching

The subsequence-to-subsequence matching, which is the task of establishing similarity based on sufficiently long similar subsequences of T1 and T2, has received very little attention and remains an open research problem. In the simplest case, where T1 and T2 are of the same time scale and basic Euclidean distance is used as the similarity measure, the order of computation for subsequence to subsequence matching is O(n3m). As time series are high dimension data, the computation of dissimilarity between two time series in their raw form is very expensive. The situation becomes even worse, if the data points of the two time series do not align.

This paper presents an approach capable of handling sequence-to-sequence, sequence-to-subsequence, and subsequence-to-subsequence matching effectively and efficiently. A segmentation algorithm that selects perceptually important points as primary break points, a time series alignment algorithm that suggests sequence-to-sequence, sequence-to-subsequence, or subsequence-to-subsequence based on the relational analysis of primary break points, and a hierarchical representation that supports coarse-fine matching are the primary contribution of this paper.

The remainder of this paper is organized five sections. Section 2 reviews briefly the related work. The HPLA based time series matching approach is described in Sect. 3. The approach consists of three major steps: Identification of perceptually important primary breakpoints, alignment of the two time series to identify pairs of corresponding candidate matching segments, and matching of segments in each suggested pair using their HPLA representations. The experimental results using a variety of time series data are analyzed and discussed in Sect. 4. Finally, the conclusion is given in Sect. 5.

2 Related Work

Several distance measures have been developed for the computation of the dissimilarity between T_1 and T_2 [2]. These distance measured are grouped into 4 categories: lock-step, elastic, edit and threshold based distance measures. The computationally efficient lock-step distances are not capable of handling even the slightest misalignment between T_1 and T_2. The elastic measures, such as

Dynamic Time Warping (DTW), allow time series to be stretched or compressed as needed to achieve good matching [3]. As time series are high dimension data and DTW uses dynamic programming requiring $O(mn)$ time, matching time series in their raw form is computationally expensive. Researchers have embraced two approaches for improving computational efficiency. They have developed techniques to speed-up DTW and other time series matching algorithms, and to represent time series compactly while preserving salient attributes.

Sakoe and Chiba [4] improved the efficiency of the DTW algorithm by defining a warp window, and by comparing each data point of T2 (query sequence) with only the data points of T1 (sequence in the data base) that are inside the warp window. The Fast Time Series Evaluation algorithm (FTSE) maps data points of T1 into a grid based on their values. The data point of T1 that matches T2[i] is determined by comparing T2[i] with only the data points of T1 that reside in the same grid cell as T2[i] [5]. The Embedding-Based Subsequence Matching (EBSM) algorithm converts each subsequence of T1 into a k-dimensional vector, where the ith component of the vector is the DTW distance between the subsequence and the ith embedding sequence. Thus, each time series T1 in the database becomes a sequence of vectors. The query sequence T2 is also converted to a vector using the same embedding sequences, and vector matching techniques are used for retrieval. The experimental results in [6] indicate one to two orders of magnitude faster retrieval than the brute force method. Though time series are converted to sequence of vectors offline, the approach generates a large number of vectors with high computational cost. Many widely used methods including DTW are natural for only sequence-to-sequence matching. There are variants of DTW algorithm, which are developed for sequence-to-subsequence matching [7]. Some methods, in order to handle this problem, cut the long time series into non-overlapping short segments, and match each segment with the query sequence [8]. Such approaches cannot retrieve any subsequence other than the stored segments. Faloutsos et al. [9] use a sliding window of size w to convert each time series in the database to a trail in a low-dimensional feature space. The window is placed at all possible position, features are extracted for the subsequence inside the window and used to map the subsequence to a point in the feature space. The trail is partitioned into subtrails, and each sub-trail is enclosed in a minimum bounding rectangle for indexing purpose. Similarly, the query sequence T2 is mapped to the feature space to determine the sequences for retrieval.

The second approach obtains compact representations of the two time series, and matches them in the representation space. During the past two decades, several representations, such as Discrete Fourier Transform (DFT) [10], Discrete Wavelet Transform (DWT) [10], Singular Value Decomposition (SVD) [11], Piecewise Aggregation Approximation (PAA) [12], Adaptive Piecewise Constant Approximation (APCA) [13], Piecewise Linear Approximation (PLA) [14], Piecewise Polynomial Approximation (PPA) [16], Symbolic Representations (SAX) [15], etc. have been developed.

It is important to note that most of the time series matching research is in the context of query by content, where the focus is on whole sequence or sequence-to-subsequence matching. The indexing centered algorithms and representations are not very beneficial to many other applications of time series matching. The speed-up techniques beneficial to indexing, for example, are not beneficial to clustering, where pairwise comparison of all time series in the dataset is needed. It is fair to say that, relatively, very little work has been done on subsequence-to-subsequence matching of time series.

Any representation that is suitable for subsequence-to-subsequence matching must segment and represent the matching subsequences in the two time series similarly, even if the two time series differ in translation, time and amplitude scale. Ideally, the segmentation should ensure a one-to-one mapping of segments of the two matching subsequences, and the corresponding segments must be similar. One approach is to partition time series at perceptually important points, and build the representation for each segment independently. The local maxima and minima, and points at which the slope changes abruptly may be taken as the perceptually important points. Bettaiah and Ranganath [17] have clearly shown that the segmentation algorithm used for the generation of PAA, APCA, PLA, and SAX representations do not segment the two time series to meet the stated requirement. Thus, they are not able to support subsequence-to-subsequence matching [19]. The DFT being a global transform is also not able to handle subsequence-to-subsequence matching [17]. The PLA representation has the potential to support the development of algorithms for all matching scenarios if each time series is segmented into identifiable segments by placing breakpoints at the perceptually important local maxima and minima [18]. However, well-known and frequently used segmentation methods (sliding window, top-down and bottom-up) do not guarantee the identification of PIPs as breakpoints.

3 The HPLA Based Time Series Matching Approach

An efficient and effective approach for subsequence-to-subsequence matching is given in this section. The approach consists of three major steps: Identification of perceptually important primary breakpoints, alignment of the two time series to identify pairs of corresponding candidate matching segments, and matching of segments in each suggested pair using their HPLA representations.

3.1 Identification of Perceptually Important Primary Breakpoints

Usually, a time series has many local maxima and minima due to the presence of noise. The goal is to develop an algorithm that ignores minor fluctuations and identifies prominent peaks and valleys that define the general shape of the time series. The following algorithm selects such prominent peaks and valleys as primary breakpoints.

```
Algorithm 1. Determine_Primary_BreakPoints (T)
//Find average raise of local maxima and average fall of
local minima
(MaxMin, MaxMinIndex) = FindMax&Min (T);
//Find all local maxima and minima
(avgRaise, avgFall) = FindAverageRaise&Fall (MaxMin);
//Select candidates for prominent maxima and minima
for i=0 to length(MaxMin)-1
   if MaxMin(i) > avgRaise OR MaxMin(i) < avgFall
      Prominent_MaxMin_I.add(MaxMin(i));
      Prominent_MaxMin_Index_I.add(MaxMinIndex(i));
end

//Select final prominent maxima and minima as primary
breakpoints
i=0;
while i < length(Prominent_MaxMin_I) - 1
   if Prominent_MaxMin_I (i) > 0
      Add index of the global maximum between Promi
      nent_MaxMin_Index_I(i) and
      Prominent_MaxMin_Index_I(i+1) to Promi
      nent_MaxMin_Index_F
   if Prominent_MaxMin_I (i) < 0
      Add index of the global minimum between Promi
      nent_MaxMin_Index_I(i) and
      Prominent_MaxMin_Index_I(i+1) to Promi
      nent_MaxMin_Index_F
   i++;
return (Prominent_MaxMin_Index_F);
```

The algorithm first identifies every maximum (minimum) with a raise (fall) greater than the average raise (average fall) as an initial perceptually important maximum (minimum), and stores its value in Prominent_MaxMin_I. The time index of each entry in Prominent_MaxMin_I is recorded in Prominent_MaxMinIndex_I. The adja-cent elements of Prominent_MaxMin_I are examined to obtain Prominent_MaxMin_F, the final list of perceptually important points.

3.2 *Time Series Alignment Algorithm*

Let, $\{p_1, p_2, p_3, \ldots, p_N\}$ be the set of breakpoints of T_1 identified by **Determine_ Primary_BeakPoints**. Let, A be the $(N \times N \times N)$ relational array, where a_{ijk} is a r-dimensional vector that specifies the relationship between p_i, p_j, and p_k. In this paper, a 2-dimensional vector $a_{ijk} = \left[\left(t_j - t_i\right)/\left(t_k - t_j\right)\ \left(T_1\left[t_j\right] - T_1[t_i]\right)/(T_1[t_k] - T_1[t_j])\right]$

is used to specify the relationship among p_i, p_j, and p_k. Note, a_{ijk} is computed for all (i, j, k), where $1 \leq i \leq (N-2)$, $i+1 \leq j \leq (N-1)$, and $j+1 \leq k \leq N$. Similarly, B is a $(M \times M \times M)$ relational array, where b_{lmn} specifies the relationship among q_l, q_m, and q_n, and is computed as $[(t_m - t_l)/(t_n - t_m)\ (T_2[t_m] - T_2[t_l])/(T_2[t_n] - T_2[t_m])]$. Note that A and B are invariant to translation, time scale, amplitude shift, and amplitude scale. The primary breakpoint mapping matrix C is computed by matching elements of A and B as follows.

```
Algorithm 2. Align_TimeSeries ($T_1$, $T_2$)
($p_1$, $p_2$, $p_3$, . . ., $p_N$) = Determine_Primary_BreakPoints
($T_1$);
($q_1$, $q_2$, $q_3$, . . ., $q_M$) = Determine_Primary_BreakPoints
($T_2$);
A = RelationalArray ($p_1$, $p_2$, $p_3$, . . ., $p_N$);
B = RelationalArray ($q_1$, $q_2$, $q_3$, . . ., $q_M$);
$c_{ij}$ = 0, for 1 ≤ i ≤ N and 1 ≤ j ≤ M
for i = 1 to N-2
  for j = i+1 to N-1
    for k = j+1 to N
      for l = 1 to M-2
        for m = l+1 to M-1
          for n = m+1 to M
            if (1 - $\varepsilon_1$) $a_{ijk}$[r] = $b_{lmn}$[r]) = (1 + $\varepsilon_1$) $a_{ijk}$[r],
                                                    for r = 1, 2
               $c_{il}$++; $c_{jm}$++; $c_{kn}$++;
List_of_Matching_Primary_Breakpoints =
Align_Primary_BreakPoints (C);
return (List_of_Matching_Primary_Breakpoints);
```

The contents of matrix C suggest possible correspondences between the breakpoints of T_1 and T_2. For example, a high value of c_{il} suggests that p_i in T_1 is very likely to correspond to q_l in T_2. If c_{jm} is zero or close to zero then p_j is unlikely to correspond to q_m. The algorithm **Align_Primary_BreakPoints** identifies likely correspondences between the primary breakpoints of T_1 and T_2 by analyzing C. The following example illustrates the use of the above algorithm for aligning two time series.

The two time series T1 and T2 to be aligned are given in Figs. 1 and 2, respectively. The time scale of T1 and the time scale of T2 differ by a factor of 2, and the subsequence T1[100:524] is identical to T2 in shape. In other words, T1 and T2 differ in translation, time scale and length. The algorithm Determine_Primary_BreakPoints partitions time series T1 into 6 primary segments by identifying 7 primary breakpoints (including end points) labeled p1 through p7. The primary breakpoints of T1 are {(1, 0), (52, 0.1735), (191, −0.1979), (263, 0.2072), (431, 0.16), (504, 0.211), (635, −0.1375)}. The five 2-dimensional arrays that make $7 \times 7 \times 7$ relational array A are shown in Fig. 3.

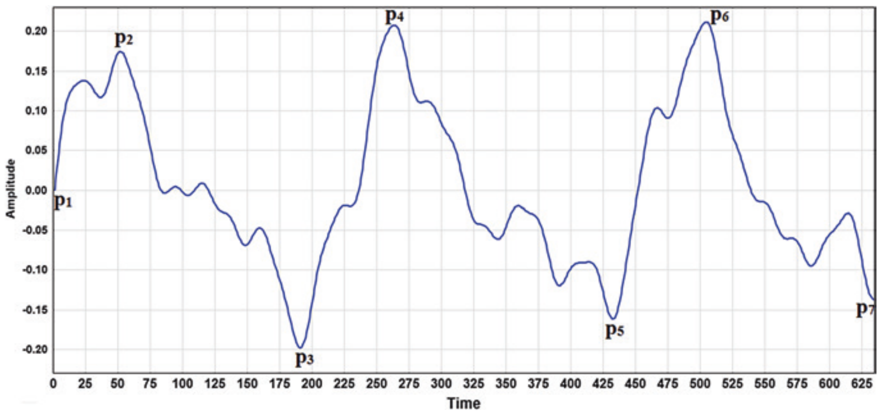
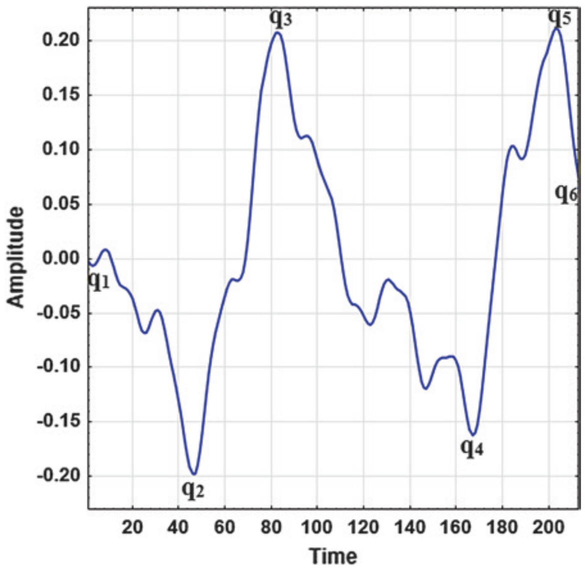


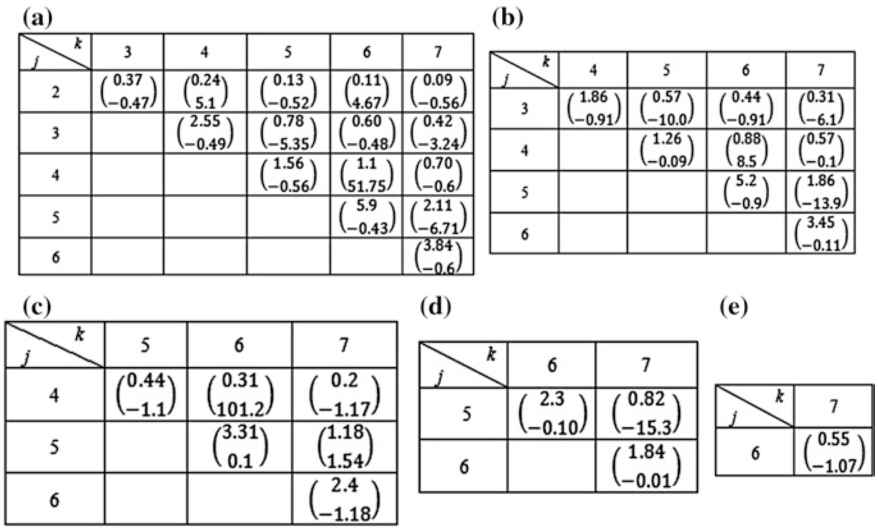
Fig. 1 The time series T_1 of length 635 with 7 primary breakpoints

Fig. 2 The time series T_2 of length 212 with 6 primary breakpoints



The time series T_2 is partitioned into 5 primary segments and the 6 primary break-points are labeled q_1 through q_7. The primary breakpoints of T_2 are {(1, −0.0023), (45, −0.1969), (82, 0.2061), (167,−0.1622), (203, 0.2112), (213, 0.075)}. The four 2-dimensional arrays that make 6 × 6 × 6 relational array B are shown in Fig. 4.

The vector aijk is taken as a match to blmn if corresponding values are within 15 % of each other. That means ε1 is set to 0.15. The 6 × 7 breakpoint mapping matrix C, where rows represent break points of T2 (q1 through q6), and columns



(a)

j \ k	3	4	5	6	7
2	$\begin{pmatrix}0.37\\-0.47\end{pmatrix}$	$\begin{pmatrix}0.24\\5.1\end{pmatrix}$	$\begin{pmatrix}0.13\\-0.52\end{pmatrix}$	$\begin{pmatrix}0.11\\4.67\end{pmatrix}$	$\begin{pmatrix}0.09\\-0.56\end{pmatrix}$
3		$\begin{pmatrix}2.55\\-0.49\end{pmatrix}$	$\begin{pmatrix}0.78\\-5.35\end{pmatrix}$	$\begin{pmatrix}0.60\\-0.48\end{pmatrix}$	$\begin{pmatrix}0.42\\-3.24\end{pmatrix}$
4			$\begin{pmatrix}1.56\\-0.56\end{pmatrix}$	$\begin{pmatrix}1.1\\51.75\end{pmatrix}$	$\begin{pmatrix}0.70\\-0.6\end{pmatrix}$
5				$\begin{pmatrix}5.9\\-0.43\end{pmatrix}$	$\begin{pmatrix}2.11\\-6.71\end{pmatrix}$
6					$\begin{pmatrix}3.84\\-0.6\end{pmatrix}$

(b)

j \ k	4	5	6	7
3	$\begin{pmatrix}1.86\\-0.91\end{pmatrix}$	$\begin{pmatrix}0.57\\-10.0\end{pmatrix}$	$\begin{pmatrix}0.44\\-0.91\end{pmatrix}$	$\begin{pmatrix}0.31\\-6.1\end{pmatrix}$
4		$\begin{pmatrix}1.26\\-0.09\end{pmatrix}$	$\begin{pmatrix}0.88\\8.5\end{pmatrix}$	$\begin{pmatrix}0.57\\-0.1\end{pmatrix}$
5			$\begin{pmatrix}5.2\\-0.9\end{pmatrix}$	$\begin{pmatrix}1.86\\-13.9\end{pmatrix}$
6				$\begin{pmatrix}3.45\\-0.11\end{pmatrix}$

(c)

j \ k	5	6	7
4	$\begin{pmatrix}0.44\\-1.1\end{pmatrix}$	$\begin{pmatrix}0.31\\101.2\end{pmatrix}$	$\begin{pmatrix}0.2\\-1.17\end{pmatrix}$
5		$\begin{pmatrix}3.31\\0.1\end{pmatrix}$	$\begin{pmatrix}1.18\\1.54\end{pmatrix}$
6			$\begin{pmatrix}2.4\\-1.18\end{pmatrix}$

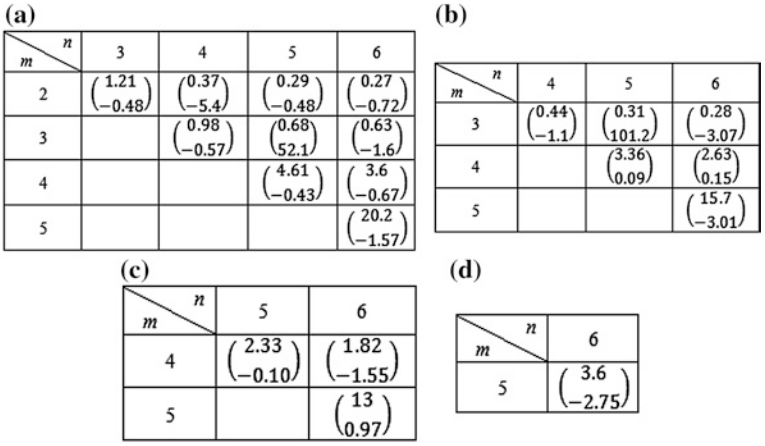
(d)

j \ k	6	7
5	$\begin{pmatrix}2.3\\-0.10\end{pmatrix}$	$\begin{pmatrix}0.82\\-15.3\end{pmatrix}$
6		$\begin{pmatrix}1.84\\-0.01\end{pmatrix}$

(e)

j \ k	7
6	$\begin{pmatrix}0.55\\-1.07\end{pmatrix}$

Fig. 3 The relational array for the time series in Fig. 1



(a)

m \ n	3	4	5	6
2	$\begin{pmatrix}1.21\\-0.48\end{pmatrix}$	$\begin{pmatrix}0.37\\-5.4\end{pmatrix}$	$\begin{pmatrix}0.29\\-0.48\end{pmatrix}$	$\begin{pmatrix}0.27\\-0.72\end{pmatrix}$
3		$\begin{pmatrix}0.98\\-0.57\end{pmatrix}$	$\begin{pmatrix}0.68\\52.1\end{pmatrix}$	$\begin{pmatrix}0.63\\-1.6\end{pmatrix}$
4			$\begin{pmatrix}4.61\\-0.43\end{pmatrix}$	$\begin{pmatrix}3.6\\-0.67\end{pmatrix}$
5				$\begin{pmatrix}20.2\\-1.57\end{pmatrix}$

(b)

m \ n	4	5	6
3	$\begin{pmatrix}0.44\\-1.1\end{pmatrix}$	$\begin{pmatrix}0.31\\101.2\end{pmatrix}$	$\begin{pmatrix}0.28\\-3.07\end{pmatrix}$
4		$\begin{pmatrix}3.36\\0.09\end{pmatrix}$	$\begin{pmatrix}2.63\\0.15\end{pmatrix}$
5			$\begin{pmatrix}15.7\\-3.01\end{pmatrix}$

(c)

m \ n	5	6
4	$\begin{pmatrix}2.33\\-0.10\end{pmatrix}$	$\begin{pmatrix}1.82\\-1.55\end{pmatrix}$
5		$\begin{pmatrix}13\\0.97\end{pmatrix}$

(d)

m \ n	6
5	$\begin{pmatrix}3.6\\-2.75\end{pmatrix}$

Fig. 4 The relational array for the time series in Fig. 2

represent breakpoints of T1 (p1 through p7) is given Fig. 5a. If the tolerance is increased to 20 % (ε1 = 0.2), the C matrix changes as shown in Fig. 5b.

In both cases, it is clear that breakpoints p_3, p_4, p_5, and p_6 align with q_2, q_3, q_4, and q_5, respectively. Once again, with only one external input (tolerance ε_1), even when the two time series differ in scale and translation, the algorithm automatically suggested subsequence-to-subsequence matching, and correctly identified likely correspondence between the primary breakpoints of T_1 and T_2. The pairs

Fig. 5 Breakpoint mapping matrices for tolerances of 10 and 20 %

$$C=\begin{bmatrix}0&0&0&0&0&0&0\\0&0&3&0&0&0&0\\0&0&0&3&0&0&0\\0&0&0&0&3&0&0\\0&0&0&0&0&3&0\\0&0&0&0&0&0&0\end{bmatrix}\quad C=\begin{bmatrix}1&1&0&0&0&0&0\\0&1&4&0&1&0&0\\0&0&1&3&0&1&0\\0&0&0&0&3&2&2\\0&0&0&0&0&3&0\\0&0&0&0&0&0&1\end{bmatrix}$$

of corresponding candidate matching segments suggested by **Align_Primary_ Breakpoints** for further matching are (p_3p_4, q_2q_3), (p_4p_5, q_3q_4), (p_5p_6, q_4q_5).

3.3 Matching of Segments Using the HPLA Representations

The alignment algorithm gives a list of pairs of segments to be matched to determine the similarity between the two given time series. It does not consider the shape of segments in the decision making process. In order to ascertain that the suggested pairs of segments indeed match, further processing is necessary. Depending on the application, one of the following two approaches may be taken.

1. Euclidean distance or DTW may be used to compute the similarity between the corresponding segments identified by the alignment algorithm. This method is suitable if the application requires the comparison of a specific time series with many time series in a dataset. As each time series in the dataset participates only in one comparison, there is no incentive to develop compact representations of segments for the purpose of matching.
2. The Hierarchical Piecewise Linear Approximation (HPLA), which supports coarse-fine matching of time series, may be used for the determination of similarity between two segments. In HPLA, any segment between adjacent primary breakpoints is called a primary segment. Each primary segment is partitioned recursively into two segments at the optimal point (secondary breakpoint) until the desired level of representation accuracy is achieved. A binary tree is used for recording the segmentation hierarchy of each primary segment (subsequence). The structure of the binary tree is simple. Each non-leaf node represents a subsequence $T[i{:}j]$, its left child represents $T[i{:}p]$, and right child represents $T[p{:}j]$, where p is the optimal break point that splits $T[i{:}j]$ into two sub-segments. Each non-leaf node includes a feature vector, components of which relate attributes of the two sub-segments represented by its two child nodes. The components of the feature vector are $(p-i)/(j-p)$ and $(T[p]-T[i])/(T[j]-T[p])$. The root node represents the subsequence by two line segments. The two non-leaf nodes in level-1 represent the subsequence by 4 line segments, and so on. The representation accuracy increases with the increasing level. Each leaf node specifies the starting point of the segment represented by the node. The HPLA representation of a time series is the time ordered sequence of binary trees of its primary segments.

The two segments in each suggested pair of candidate segments are matched by matching their binary trees. Two primary segments are considered similar if the feature vectors of the corresponding nodes of their binary trees are similar. The time series T_1 and T_2 are considered similar, if a sufficiently long continuous sequence of binary trees in the HPLA representation of T_1 matches a sequence of binary trees in the HPLA representation of T_2.

Therefore, for the example being considered, the six primary segments p3p4, p4p5, and p5p6 of T1, and their corresponding segments q2q3, q3q4, and q4q5 of T2 are further segmented to obtain their HPLA representations. As the components of the feature vectors of the corresponding non-leaf nodes of the corresponding segments are within 15 % of each other, the segments in all three suggested pairs are taken as similar. Therefore, T1 and T2 are taken as similar time series.

4 Experimental Results

Two datasets from UCR (Mallat and OliveOil) [20] and Pseudo Periodic Synthetic Time Series from UC Irvine archive (http://kdd.ics.uci.edu) are used as base time series to create a search set and a query set of time series. The three base time series are shown in Fig. 6. From each base time series, several translated, time scaled, amplitude scaled, and amplitude shifted versions are created. Some of these are corrupted with random Gaussian noise. Finally, the created time series are partitioned into two sets, a search set of 212 time series and query set of 40 time series. The search set includes 84 time series created from Mallat, 44 time series created from OliveOil, and 84 time series created from Pseudo Synthetic data. The query set includes 16 time series created from Mallat, 8 time series created from OliveOil, and 16 time series created from Pseudo Synthetic data. The data is carefully chosen to include several cases of sequence-to-sequence, sequence-to-subsequence, and sub-sequence-to-subsequence matching scenarios.

4.1 Matching Approach and Simulation Results

Each query time series Q in $\{Q_1, Q_2, \ldots, Q_{40}\}$, is matched with all 212 time series in the search set $S = \{T_1, T_2, \ldots, T_{212}\}$, and time series similar to the query time series are identified. Ideally, the matching algorithm should identify all time series in the search set that are created from the same base time series as the query time series, and others should not be selected. The details of the two stage matching approach used are given below.

1. As the 2×1 feature vectors used in the HPLA representation are invariant to amplitude scale and amplitude shift, the time series are not normalized. In fact, normalization is meaningful only in the case of whole sequence matching for

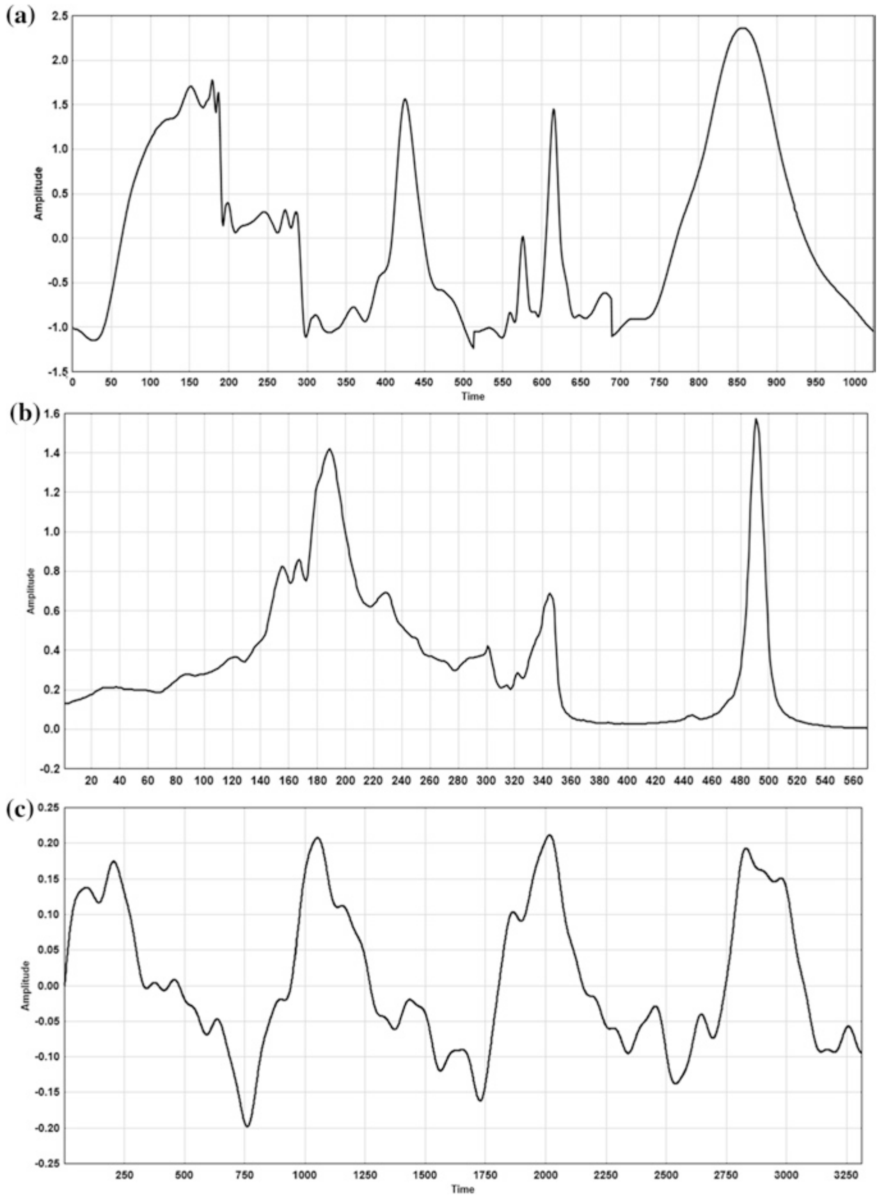


Fig. 6 The three time series used for the creation of the search and query sets. **a** Mallat time series of length 1,024. **b** OliveOil time series of length 570. **c** Pseudo Synthetic time series of length 3,313

representations that are not invariant to amplitude shift and amplitude scale. As the mean and standard deviation of a time series and its sub-series are usually not equal, the normalization is more likely to hurt the matching process than help when the similarity is based on sequence-to-subsequence or subsequence-to-subsequence matching. The optional smoothing step is also not used.

2. For each time series T in S, primary breakpoints are identified using the algorithm **Determine_Primary_Breakpoints**.
3. Each primary segment is partitioned recursively at the optimal point to obtain its binary tree representation [19]. For the sake of uniformity, each primary segment is represented by a complete binary tree with 8 leaf nodes (4 levels). The feature vector of each non-leaf node is computed.

Each query time series *Q* in the query set is matched with each *T* in *S* in two stages as described below.

Stage 1: *Selection of candidate time series from the search set*

The goal is to select a candidate subset of S for further matching in Stage 2. Ideally, the candidate set should include all time series in S that are similar to Q and none of the time series that are not similar to Q. In reality, the set will include a few time series not similar to Q (false positives) and not include a few time series similar to Q (false negatives).

In order to determine the candidate set, the primary breakpoints and the HPLA representation of Q are determined first. Algorithm Align_TimeSeries is used to obtain the list of pairs of potential matching primary breakpoints, which suggest pairs of candidate segments from Q and T for further matching. The value of the parameter ε1 used for comparing the corresponding elements of the relational arrays of T and Q is set at 0.15. If more than 50 % of the primary breakpoints of Q align with the primary breakpoints of T in the same time order, T is selected for further matching in Stage 2. Otherwise, T is not a candidate for further matching.

Stage 2: *Filtering the false positives*

The goal is to filter as many false positives as possible by matching the HPLA representations of Q and each T. The two segments (one of Q and one of T) in each suggested pair are matched by comparing their HPLA representations to determine if they are similar. The matching of the two binary trees begins with the matching of the feature vectors stored in their root nodes. If the two feature vectors are not similar, the two segments are considered as dissimilar. If the two feature vectors are similar, the matching is continued using non-leaf nodes in the next level. This process terminates when all corresponding non-leaf nodes are exhausted. The user may limit matching to root nodes, or consider the non-leaf nodes in other levels, depending on the level of accuracy desired. The time series T is considered similar to Q, if 75 % of the suggested segment pairs of T and Q are found similar. The simulation results, evaluated and discussed in the next section are tabulated in Table 1. Because of space constraints, results for 12 out of 40 query time series are shown.

Table 1 Simulation results of the HPLA representation based time series matching

Query time series	Number of candidates from stage 1	Number of time series selected in stage 2	Number of false positives	Number of false negatives	Recall	Precision	Pruning Power
Q_1	95	85	3	2	0.9761	0.9647	0.0859
Q_2	101	84	0	0	1	1	0.1328
Q_3	90	84	2	2	0.9761	0.9761	0.0468
Q_4	92	84	0	0	1	1	0.0625
Q_{17}	49	44	0	0	1	1	0.0297
Q_{18}	51	44	0	0	1	1	0.0416
Q_{19}	50	44	0	0	1	1	0.0357
Q_{20}	48	46	2	0	1	0.9565	0.0238
Q_{25}	94	85	1	0	1	0.98	0.0781
Q_{26}	90	84	0	0	1	1	0.0468
Q_{27}	89	85	1	0	1	0.98	0.0390
Q_{28}	88	84	0	0	1	1	0.0312

4.2 Evaluation of Simulation Results

The two metrics, recall and precision, commonly used for evaluating the performance of database retrieval methods are used for the evaluation of the performance of the HPLA based time series matching approach. Recall is defined as the ratio of the number of truly matching time series retrieved to the total number of matching time series in the search set. The value of recall is in the range [0, 1], where higher value indicates better performance. A recall value of 1 indicates that all matching time series in the database are retrieved without any false negatives. Precision is defined as the ratio of the truly matching time series retrieved to the total number of time series retrieved from the search set. The value of precision is also in the range [0, 1]. A value of 1 indicates that there are no false positives.

Pruning power is another frequently used metric which specifies the number of time series considered for matching. For the current situation, as an HPLA based indexing method is not developed, the number of time series ruled as non-matching by Align_TimeSeries may be used as a measure of the pruning power. In this paper, pruning power is defined as the ratio of the number of time series selected for matching by Align_TimeSeries minus the number of time series similar to Q in S to the total number of time series in S minus the number of time series similar to Q in S.

The simulation results in Table 1 (shown only for 12 query time series), which lists number of false positives, number of false negatives, recall, precision, and pruning power are very impressive. The important observations about the HPLA based matching approach, and results are discussed below.

1. Each time series in the query set is similar to a subset of time series in the search set. The similarity may be based on sequence-to-sequence, sequence-to-subsequence or

Table 2 The average pruning power, recall and precision for Mallat, OliveOil, and Pseudo Synthetic query sets

Query set	Average pruning power	Average recall	Average precision
Mallat	0.0469	0.9925	0.9921
OliveOil	0.0394	0.9943	0.9836
Pseudo Synthetic	0.0423	0.9937	0.9925

subsequence-to-subsequence matching. The matching scenario that establishes similarity between query and search time series is not known in advance. For example, consider Q_2 of length 512, which is created from Mallat of length 1,024 by adding 1.85 to each time scaled (factor of 2) value. In the search set there are 23 time series similar to Q_2 based on whole sequence matching, 31 time series similar to Q_2 based on sequence-to-subsequence matching, and 30 time series similar to Q_2 based on sub-sequence-to-subsequence matching. A total of 84 time series in S are similar to Q_2, and the remaining 128 time series in S are not similar. In Stage 1, the algorithm **Align_TimeSeries** has identified the correct matching scenario for Q_2, and for all other 39 query sequences.

2. The analysis of relative positions of primary breakpoints is able to prune most of the non-matching time series in S from further consideration. For example, in the worst case, 101 candidates are selected by **Align_TimeSeries** for further matching with Q_2 in Stage 2. As there are 84 time series in S that are similar to Q_2, even in the worst case, only 17 additional time series are selected for matching in Stage 2. On the average, for the Mallat query set, only 90 time series are selected as candidates for matching in Stage 2, giving an average pruning power of 0.0469 ((90 − 84)/(212 − 84)). The average pruning powers are also given for OliveOil and Pseudo Synthetic query sets in Table 2.
3. The segment by segment matching of the HPLA representations of Q and T has filtered most of the false positives selected in Stage 1. For example, 101 candidates selected by **Align_TimeSeries** for further matching in Stage 2 include all 84 time series that are similar to Q_2, and 17 time series that are not similar to Q_2 (false positives). The HPLA based matching in Stage 2 filters all the false positives, and retains all 84 time series that are similar to Q_2. The ability of the method in eliminating or significantly reducing the number of false positives is consistently good for all 40 cases.
4. The effectiveness of the HPLA based time series matching is obvious from high recall and precision values, which are close to the ideal value of 1. The low values of pruning power (close to the ideal value of 0) indicate the potential for building an efficient HPLA based indexing approach. The average recall and precision for the three groups of query sequences (Mallat, OliveOil, and Pseudo Synthetic) are given in Table 2.
5. The approach requires the specification of only two tolerance parameters ε_1 and ε_2. The value of ε_1 directly affects the number of false negatives and false positives in Stage 1. As ε_1 increases, the number of false positives increases and the

number of false negatives decreases. In order to avoid the risk of losing matching time series in Stage 1, the value of ε_1 should be relatively high. Similarly, the value of ε_2 also affects the number of false positives and false negatives in Stage 2. As the value of ε_2 increases, the number of false positives increases and the number of false negatives decreases.

5 Conclusion

In summary, the HPLA based time series approach described in this paper handles all three matching scenarios uniformly by identifying the appropriate scenario to be used through the analysis of the relative positions of the perceptually important primary breakpoints in query and search sequences. There is no need for the user to specify the type of matching scenario needed. The approach identifies the matching scenario automatically, and also prunes most of the non-matching time series in the search set from further consideration. The HPLA based matching algorithm filters most of the false positive, and achieves high precision and recall. The approach is invariant to time and amplitude translation and scale differences between the two time series matched, and requires the specification of two simple tolerance parameters as external input.

References

1. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. **45**, 34 (2012)
2. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. Very Large Databases **1**, 1542–1552 (2008)
3. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD Workshop, pp. 359–370. AAAI Press, Palo Alto (1994)
4. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Sig. Proc. **26**(1), 43–49 (1978)
5. Morse, M.D., Patel, J.M.: An efficient and accurate method for evaluating time series similarity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 569–580 (2007)
6. Athitsos, V., Papapetrou, P., Potamias, M., Kollios, G., Gunopulos, D.: Approximate embedding-based subsequence matching of time series. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 365–378 (2008)
7. Park, S., Kim, S., Chu, W.W.: Segment-based approach for subsequence searches in sequence databases. In: Symposium on Applied Computing, pp. 248–252 (2001)
8. Zhu, Y., Shasha, D.: Warping indexes with envelope transforms for query by humming. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 181–192 (2003)
9. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Snodgrass, R.T., Winslett, M. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 19–429 (1994)

10. Wu, Y., Agrawal, D., Abbadi, A.E.: A comparison of DFT and DWT based similarity search in time-series databases. In: Proceedings of the 9th International Conference on Information and Knowledge Management, pp. 488–495 (2000)
11. Hung, N.Q., Anh, D.T.: An improvement of PAA for dimensionality reduction in large time series databases. In: Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence, pp. 698–707 (2008)
12. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. **3**, 263–286 (2001)
13. Chakrabarti K, Keogh E, Mehrotra S, Pazzani M.: Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans. Database Syst. **27**, 118–228
14. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: 4th International Conference on Knowledge Discovery and Data Mining, pp. 239–241 (1998)
15. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Discov. **15**, 107–144 (2007)
16. Lemire, D.: A better alternative to piecewise linear time series segmentation. In: SIAM Data Mining (2007)
17. Bettaiah, V., Ranganath, H.S.: An analysis of time series representation methods. In: Proceedings of the 2014 ACM Southeast Regional Conference, Article 16, p 6 (2014)
18. Bettaiah, V., Ranganath, H.S: Two stage segmentation for efficient time series matching. In: 2nd International Conference on Research in Science, Engineering and Technology, pp. 29–35 (2014)
19. Bettaiah, V., Ranganath, H.S: An effective subsequence-to-subsequence time series matching approach. In: Science and Information (SAI) Conference, pp. 112–122 (2014)
20. The UCR Time Series Classification/Clustering. http://www.cs.ucr.edu/eamonn/time_series_data/. Retrieved Aug 2013