Engineering Applications of Artificial Intelligence 26 (2013) 2581–2596

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

OBST-based segmentation approach to financial time series

Yain-Whar Si*, Jiangling Yin

Department of Computer and Information Science, University of Macau, Av. Padre Tomas Pereira, Taipa, Macau

ARTICLE INFO

Article history:
Received 2 February 2013
Received in revised form
26 June 2013
Accepted 29 August 2013
Available online 7 October 2013

Keywords:
Financial time series
Turning points
Segmentation
Trends
Optimal binary search tree

ABSTRACT

Financial time series data are large in size and dynamic and non-linear in nature. Segmentation is often performed as a pre-processing step for locating technical patterns in financial time series. In this paper, we propose a segmentation method based on Turning Points (TPs). The proposed method selects TPs from the financial time series in question based on their degree of importance. A TP's degree of importance is calculated on the basis of its contribution to the preservation of the trends and shape of the time series. Algorithms are also devised to store the selected TPs in an Optimal Binary Search Tree (OBST) and to reconstruct the reduced sample time series. Comparison with existing approaches show that the time series reconstructed by the proposed method is able to maintain the shape of the original time series very well and preserve more trends. Our approach also ensures that the average retrieval cost is kept at a minimum.

1. Introduction

Time series is a temporal data object which is used to store a collection of observations made chronology (Fu, 2011). Time series are usually generated from a wide range of scientific and financial applications. Examples of time series include historical price and trading volume obtained from financial stock markets, motion and speech data from computer vision applications, and electrocardiogram (ECG) data from medical diagnosis systems. Time series are often characterised as large in data size, high in dimensionality, and require continuous updating (Fu, 2011).

There are two common approaches for analysing financial markets; fundamental analysis and technical analysis. In fundamental analysis, factors such as profit and loss, performance data, and analysis for the corresponding industry are used to predict the movement of the price. Analysts may also take into account other factors such as government policies and political climate for the prediction. However, fundamental analysis often fails to predict price movement due to the sheer number of factors involved and the influence of the biases from the analysts' judgment. In technical analysis, time series data such as historical price, volume, and other statistical data are used to predict the future price. In this paper, we address the segmentation problem of financial time series.

Segmentation is often performed prior to locating technical patterns such as Head-and-Shoulder or Double Tops patterns in financial time series. In order to match with the patterns from the pattern template in different resolution, feature points need to be extracted during the segmentation process and matched with the pattern template (Fu et al., 2007; Zhang et al., 2010a). For instance, an extended version of Perceptually Important Points (PIP) algorithm is used to segment the time series for locating technical patterns such as Up-Trapeziform, Bottom-Trapeziform, Up-flag, and Down-flag from the bid and trade time series of Chicago stock market (Zhang et al., 2007). Segmentation method such a Piecewise Linear Regression method is also used to find the past trends in the stock data (Lavrenko et al., 2000). The trends identified in the segmentation step are then used in combination with the news stories for predicting the future trends. Piecewise Linear Representation method is also used as a pre-processing step to segment the time series before predicting the future stock price based on a multilayer feed forward artificial neural network (Kwon and Sun, 2011).

Various methods for the time series segmentation have been proposed, including Discrete Fourier Transform (DFT) (Agrawal et al., 1993; Chu and Wong, 1999), Discrete Wavelet Transform (DWT) (Chan and Fu, 1999; Kahveci and Singh, 2001), Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001a), Adaptive Piecewise Constant Approximation (APCA) (Chakrabarti et al., 2002), Piecewise Linear Representation (Piecewise Constant Approximation) (Keogh, 1997; Keogh and Pazzani, 2000), and Singular Value Decomposition (SVD) (Kanth et al., 1998). These segmentation approaches usually focus on the lower bound of the Euclidean distance, but smooth out the salient points of the original time series (Fu et al., 2008). Such smoothing effect inadvertently removes the important points which are crucial in identifying the change in trends and shapes. In this paper, we propose an approach for identifying and storing important points

* Corresponding author. Tel.: +853 83974923.
E-mail addresses: fstasp@umac.mo (Y.-W. Si), jyin@eecs.ucf.edu (J. Yin).

0952-1976/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2013.08.015

which can be used to preserve the overall shape and trend of the time series.

Uncovering financial market cycles and forecasting market based on trends has been extensively studied by analysts. One of the renowned studies on trend analysis is the Elliott Wave Principle (Frost and Prechter, 2001) which is used to describe movement of the stock based on five distinct waves on the upside and three distinct waves on the downside. The major and minor waves determine the major and minor market trends, respectively. Elliott Wave Principle becomes one of the important foundations in stock market analysis for identifying significant trends for forecasting price movement. When analysing or predicting the movement of financial time series, analysts often rely on the trends contained within the time series rather than on the exact value of each data point in that series (Atsalakis and Valavanis, 2009; Chung et al., 2002; Kamijo and Tanigawa, 1990). In addition, Dow Theory (Hamilton, 2010) also categorises stock movement into primary and secondary trends. Accordingly, as an important characteristic, these trends should be taken into consideration during time series segmentation. In this paper, we propose a segmentation method which is not only capable of segmenting the time series for storing and incremental retrieving of important data points but also capable of preserving as much trends as possible for further data mining tasks.

A time series comprises numerous data points. However, the fluctuations, trends and patterns within a time series are reflected by only a small number of local maxima or minima data points. These data points in time series are often called landmarks and can be defined by the points, times or events of greatest importance, with the points primarily part of the local maxima or minima. Perng et al. (2000) define a landmark model for performing similarity pattern searching in a database. Instead of using raw data, the model proposed in Perng et al. (2000) employs landmarks to improve the efficiency of series data processing. The perceptually important points (PIP) method proposed in Zhang et al. (2010a) also selects critical points called PIPs to preserve the general shape of the time series. The resulting PIPs are able to reflect any fluctuation from the original time series. Keogh et al. (2001b) use the end points of the fitted line segments to compress the time series, whereas Pratt (2001) also select part of the local maxima or minima to perform a similar search for time series. In this paper, we propose the use of the local minimum and maximum points on a time series, called Turning Points (TPs), for segmentation. TPs can be extremely useful in technical analysis because they can be used to preserve the overall shape and trend of the time series compared to other data points.

To allow the incremental reconstruction of reduced-sample time series from these TPs at a later stage, we propose an algorithm to calculate the degree of importance for each TP, which is a measure used to compare its contribution to the preservation of the trends and overall shape of the original time series. Using the degree of importance as a measure allows TPs to be ordered on the basis of their priority. An Optimal Binary Search Tree (OBST) is then used to store the TPs. The advantage of the proposed storage scheme is that it allows the more important points to be retrieved at an early stage to reconstruct the reduced-dimension time series. The use of an OBST also guarantees that the average search cost is kept to a minimum.

In addition, analysts can employ the proposed method to reconstruct reduced-dimension time series at different detail levels by retrieving TPs based on their degree of importance. Such a property allows the top-down analysis of the time series in which highly visible trends are first identified and allows the retrieval of more detailed segments to be postponed until later stages. Our experimental findings show the proposed method to achieve promising results in preserving a higher number of trends than existing methods.

The paper is an extension of our previous conference paper (Yin et al., 2011) and as such the background and definition of turning point are taken from this previous work. The remainder of the paper is organised into five sections. We discuss related work on financial time series segmentation in Section 2. The algorithms for calculating the degree of importance and storing the TPs into an OBST are provided in Section 3. In Section 4, we report details of the experimental results obtained from tests of our proposed approach with price data from the Hong Kong stock market, and in Section 5, we summarise our ideas and discuss directions for future research.

2. Related work

The multitude of time series data produced by a wide range of applications has been increased at an unprecedented pace in recent years. Application such as financial trading, medical diagnosis, computer vision, and speech recognition can produce a vast amount of time series data. The complexity and dimensionality of the time series data vary from one application to another. For instance, time series data from computer vision applications such as body pose tracking and action recognition are in high dimension and non-linear.

A number of techniques have been proposed by the scientific community to analyse these high dimensional time series data. Dimensionality reduction is often referred to as a feature extraction step in machine learning and statics. For instance, linear dimensionality reduction method such as Principal Component Analysis (PCA) (Pearson, 1901) transforms the data from high-dimensional space to low-dimensional space. PCA has been widely used in applications such as face recognition to reduce the number of variables.

Two main categories of techniques are available in non-linear dimensionality reduction methods; mapping-based and embedding-based methods (Lewandowski et al., 2010). Mapping-based approach such as Gaussian Process Latent Variable Mode (GP-LVM) (Lawrence, 2003), Back Constraint Gaussian Process Latent Variable Mode (BC-GPL-VM) (Lawrence and Quinonero-Candela, 2006), Gaussian Process Dynamical Model (GPDM) (Wang et al., 2006) comprise a mapping from a latent space to the data space. GP-LVM technique proposed by Lawrence (2003) is a non-linear generalisation of probabilistic PCA providing a smooth mapping from latent to data space. Embedded-based approaches such as Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2001), Temporal Laplacian Eigenmaps (TLE) (Lewandowski et al., 2010), Isomap (Tenenbaum et al., 2000), and ST_Isomap (Jenkins and Mataric, 2004) estimate the structure of the underlying manifold by approximating each data point according to their local neighbours on the manifold (Lewandowski et al., 2010).

Financial time series analysis and prediction constitute active research problems in the data mining area due to the attractive commercial applications and benefits that data mining offer (Kamijo and Tanigawa, 1990; Ralanamahatana et al., 2005). A number of algorithms, such as classification, clustering, segmentation, indexing and rule discovery (Atsalakis and Valavanis, 2009; Chung et al., 2002) from historical time series have been introduced in this area.

The segmentation of financial time series is a crucial pre-processing step in financial time series analysis. A number of approaches have been proposed for sampling and segmentation of financial time series: Discrete Fourier Transform (DFT) (Agrawal et al., 1993; Chu and Wong, 1999), Discrete Wavelet Transform (DWT) (Chan and Fu, 1999; Kahveci and Singh, 2001), the Piecewise Linear, and Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001a, 2001b), and Singular Value Decomposition (SVD)

(Kanth et al., 1998). In this section, we review three common segmentation methods for time series, namely, PLA, PIP, and PAA.

2.1. Piecewise Linear Approximation method

Piecewise Linear Approximation (PLA) is one of the most frequently used representations (Fuchs et al., 2010). PLA uses K straight lines to approximate a time series T with length n. Fig. 1 presents an example of the approximation of a data segment employing this method. By setting K smaller than n ($n=21$ and $k=8$ in this example), the PLA method can be used for segmentation. PLA is one of the methods to abstract key features of the data for higher-level representation. Such representation is useful since using original data for querying in time series databases is computationally expensive (Keogh and Pazzani, 1999). PLA has been used as a preprocessing step in a number of data mining applications including:

- Feature extraction, data compaction, and noise filtering of boundaries of regions of pictures and waveforms (Pavlidis and Horowitz, 1974),
- Relevance feedback retrieval of ECG data (Keogh and Pazzani, 1999),
- Pattern matching in time series sensor data from Space Shuttle Mission Data Archives (Keogh and Smyth, 1997), and
- Predicting trends in stock prices based on the content of news stories (Lavrenko et al., 2000).

Moreover, in financial time series analysis, PLA has been employed to perform a fast similarity search (Park et al., 2000), and it has also been applied to pattern matching (Wu et al., 2004; Zhang et al., 2010b) and trading point prediction (Chang et al., 2009).

PLA can be obtained through the sliding window, top-down, bottom-up, or B-spline wavelet methods (Keogh et al., 2001b), and segments can be recursively generated by applying certain actions (such as partitioning in the top-down method or merging in the bottom-up method) on the time series until a predefined stopping criterion is met.

The pseudocode for the top-down implementation of PLA (Keogh et al., 2001b) is described in Algorithm 1. First, the algorithm locates the best location for splitting the time series. Next, based on the error threshold defined by the user (max_error), the algorithm recursively continues to split the left and the right segments. The algorithm stops when the approximated error is less than the user defined threshold.

Algorithm 1. Pseudocode for generic Top-Down PLA (Keogh et al., 2001b).

```
Function Seg_TS=Top_Down(T, max_error)
  best_so_far=inf;
//Find best place to split the time series.
for i=2 to length(T)-2
    improvement_in_approximation
  =improvement_splitting_here(T,i);
     if improvement_in_approximation
  <best_so_far
          breakpoint=i;
              best_so_far=improvement_
  in_approximation;
     end;
  end;//Recursively split the left segment if
  necessary
if calculate_error(T[1:breakpoint])>max_error
     Seg_TS=Top_Down(T[1:breakpoint]);
end;
```

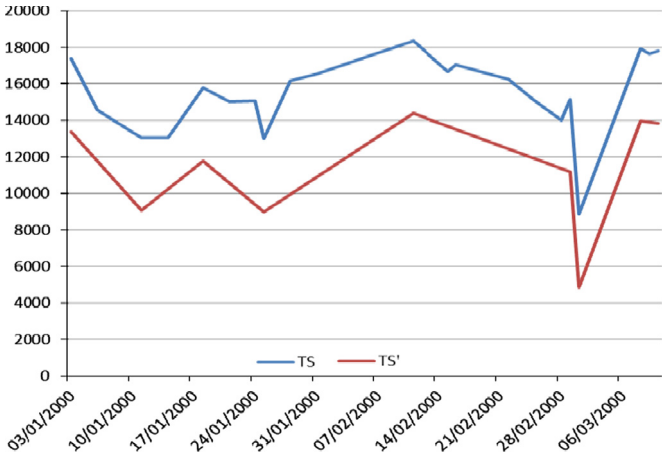

Fig. 1. Intuitive example of PLA method representation.

```
//Recursively split the right segment if necessary
if calculate_error(T[breakpoint+1:length(T)])
  >max_error
    Seg_TS=Top_Down(T[breakpoint+1:length(T)]);
end;
```

One difficulty in the PLA approach is the selection of an appropriate stopping condition (the 'max_error' in Algorithm 1), which has to be defined by the user. In general, the lengths of the segments extracted by the PLA approach will increase when higher threshold values are used. Keogh et al. also noted that the top-down and bottom-up versions of PLA can be modified to include the desired number of segments K. For a given time series with n data points, and the average segment length $L=n/K$, the complexity of top-down, bottom-up, and sliding window algorithms are $O(n^2K)$, $O(Ln)$, and $O(Ln)$ respectively (Keogh et al., 2001b). Among these algorithms, only sliding window version of PLA can segment continuous streams of time series. According to the experimental results (Keogh et al., 2001b), sliding-window and top-down algorithms do not scale well whereas bottom-up approach scales linearly with the size of the dataset. To address the shortcomings of these methods, Keogh et al. introduced a hybrid algorithm called Sliding Window and Bottom-up (SWAB). Experimental results (Keogh et al., 2001b) show that the SWB algorithm scales linearly with the size of the dataset and produces high quality approximations requiring only constant space. Analysts often employ different threshold values to decompose historical data based on the underlying characteristics of stocks. Several approaches have been proposed to choose the right threshold value. For instance, Chang et al. (2009) employ genetic algorithms for this purpose.

2.2. Perceptually Important Points method

Perceptually Important Points (PIP) (Chung et al., 2001; Fu et al., 2008) selects a group of critical points called PIPs to represent the original time series. The pseudocode for identifying PIPs is described in Algorithm 2. In the identification process, the first and last points of the time series are chosen as the PIPs. Next, from the remaining points on the time series, the algorithm selects that with the longest distance to the first two PIPs as the next PIP. In each subsequent step, it continues to select the point with the maximum distance to its two adjacent PIPs. The point selected may be located either between the first and second PIPs or the second and last PIPs. The process continues until all of the points in the time sequence are added to the list (the PIPList in the algorithm) or a stopping criterion is met (for example, a lower Euclidean distance). Eq. (1) is used to measure

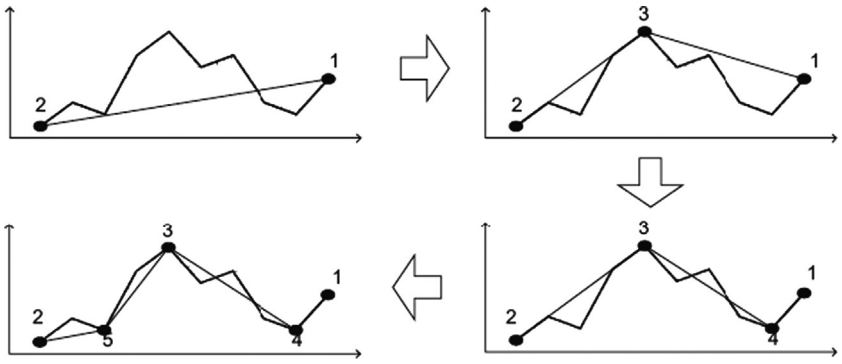

Fig. 2. Identification of the first five PIPs using the PIP method (Fu et al., 2008).

the distance from the next PIP (x_2, y_2) to its two adjacent PIPs ((x_1, y_1) and (x_3, y_3)). Fig. 2 illustrates the identification of the first five PIPs via the PIP method.

$$Dis = |\hat{y}_2 - y_2| = \left| y_1 + \frac{(y_3 - y_1)(x_2 - x_1)}{(x_3 - x_1)} - y_2 \right| \quad (1)$$

Algorithm 2. Pseudocode of the PIP identification (Fu et al., 2008).

```
Function PIP_Identification
Input: sequence P[1...m]
Output: PIPList L[1...m]
Begin
  Set L[1]=P[1], L[2]=P[m]
Repeat until L[1...m] all filled
Begin
  Select point P[j] with maximum distance to the
  adjacent points in  PIPList (L[1] and L[2]
  initially)
Append P[j] to L
End
Return L
```

This method focuses primarily on preserving the general shape of the time series. Because the algorithm results in no transformation of the original time series, and the PIPs are also selected from the original time series with no changes, it is very useful in helping stock market investors to understand data mining results. However, the selection process in the PIP approach is global (that is, the entire time series must be taken into account), and it may therefore be unsuitable for online applications in which new data are frequently added in real time. To alleviate this drawback, several localisation approaches have been proposed to apply PIPs in online applications or subsequence mining. For instance, the sliding window method was adopted for real-time pattern matching in Fu et al. (2007) and Zhang et al. (2010a).

2.3. Piecewise aggregate approximation and adaptive piecewise constant approximation method

PAA is designed to reduce a time series from n dimensions to N dimensions, and the original time series is divided into N equal-sized segments (Keogh et al., 2001a). The mean value of each segment is calculated to represent the corresponding segment. Two extreme cases using this method are as follows:

- When $N=n$, the transformed representation is identical to the original time series, as each point becomes one segment, and the mean value is exactly equal to that point value.

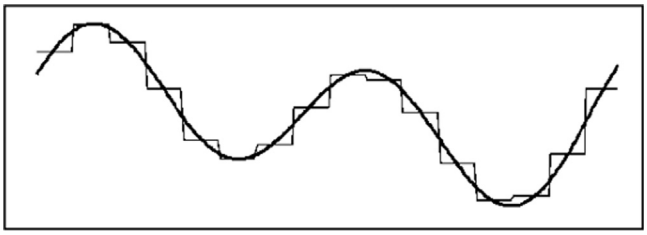

Fig. 3. Visualisation of the PAA method.

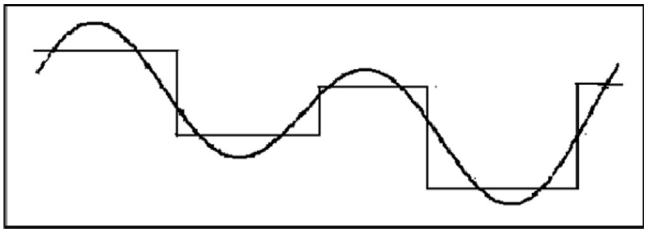

Fig. 4. Visualisation of the APCA method.

- When $N=1$, the transformed representation is the mean of the original sequence, as the entire time series belongs to only one segment. Fig. 3 presents a straight view of how PAA approximates a sequence with sub-piece segments.

Because the representation is based on piecewise segments, it can be used directly for the searching sequence with variable lengths or pattern matching from the local perspective. Another advantage of this method is that it is faster than others because it needs to scan the original data only once. Also, each segment represented by PAA has the same length, which makes the storage of representations in databases easier. A disadvantage of this method, however, is the need to determine a proper length or how many segments should be adopted.

Keogh et al. (2001a) introduced APCA as an extension to PAA representation. Instead of equal length division, APCA allows the segments to have arbitrary lengths. Fig. 4 presents an intuitive example of this method. In contrast to PAA, this type of representation requires two numbers to record each segment. The first number records the mean value of a segment, and the second records the length of each segment. As APCA approximates a time series by a set of constant value segments of varying lengths, it achieves better compression than PAA for time series with fewer fluctuations.

2.4. Comparison of PLA, PIP, and PAA approaches

In this section, we briefly compare the PLA, PIP, and PAA approaches based on three properties. The overview of the comparison is given in Table 1.

Online use/representation: Online use or representation concerns about how to deal with the new data. The algorithm of PAA depends on local calculation while PLA and PIP approaches need to compare all the data points from a global view. Thus PAA approach could be directly used for online representation while PLA and PIP approaches are unsuitable.

Representation interval: Representation interval denotes the length of a time series slice, which is determined by specific criteria such as breakpoints. PAA approach is based on segments with identical length calculation. Representational interval of PLA and PIP depends on the degree of fluctuation of the time series.

Complexity: PLA and PIP approaches takes $O(n^2)$ time since identifying a point during the segmentation process requires the comparison of the entire data points once (see Algorithms 1 and 2). PAA approach needs $O(n)$ because the algorithm scans the data only once.

Table 1
Comparison of PLA, PIP, and PAA approaches.

Properties	PLA	PIP	PAA
Online use/representation	Unsuitable	Unsuitable	Suitable
Representation interval	Variable	Variable	Identical length
Complexity	$O(n^2)$	$O(n^2)$	$O(n)$

3. Turning point selection, storage and retrieval

In this section, we first outline the properties of TPs. We then introduce an approach for the storage and retrieval of TPs based on their degree of importance. In this method, TPs are prioritised according to their contribution to the preservation of the trends and shape of the time series. Fig. 5 presents an overview of the proposed method, in which there are four steps. First, TPs are identified from the original time series. Second, the selected TPs are evaluated and stored into a stack according to their degree of importance, which is calculated on the basis of their effectiveness in preserving the trends and shape of the time series. Third, an OBST is built to store the TPs. Finally, a reduced-sample time series can be reconstructed based on user criteria.

3.1. Turning points

PAA models and the DFT and PLA methods usually smooth out some of the data points when sampling is performed on a time series. These methods involve a certain degree of information loss due to the smoothing effect. However, financial analysts often depend on the local maximum and minimum points to identify technical patterns or transaction periods. For example, such frequently appearing technical patterns as Head-and-shoulder patterns, which consist of a head point, two shoulder points and a pair of neck points, can typically be characterised by a few local maximum and minimum points. Fig. 6 shows all seven points of the technical pattern formed by the local maximum and minimum points on a time series. These points are called TPs.

TPs can be used to represent stock movement trends, and they can also be employed to identify the beginning or end of a crucial transaction period. For instance, in a stock market scenario, a trader may buy shares at the minimum price and sell them at the maximum price. These buying and selling actions define the beginning and end of the transaction period. The difference between the two extreme prices can be considered the profit. Bao and Yang (2008) proposed an intelligent stock trading system that operates by evaluating TPs and employing probabilistic reasoning to forecast future price movements. We denote a point a_i on a time series as a TP

1. if the stock price ends the increasing trend at a_i and starts a decreasing period or
2. ends the decreasing trend at a_i and starts an increasing period.

In general, we can define the TPs as the maximum or minimum points during a given period. The TPs in Fig. 7 are depicted as filled-in circles.

Due to the frequent fluctuations in the stock market, a large number of TPs can be found in a time series. For instance, using the foregoing criteria, we can extract 1284 TPs from the 2590 data points in the Hang Seng Index (HSI) over the past 10 years. In the following section, we detail an algorithm for the identification and evaluation of TPs from a financial time series.

3.2. Identification and evaluation

The proposed method comprises two phases: an identification phase and an evaluation phase. In the identification phase, we select all minimum or maximum points on the time series and label them as TPs. In the evaluation phase, TPs that contribute less to the overall shape of the time series are assigned a lower degree of importance and stored into a stack early in the process.

Identification phase: All of the minimum or maximum points from the time series are first marked. For our purposes, these points are denoted as $TP[1\ldots m]$. Let $f(x)$ be the value of a point x in a time series.

1. For point p_i and neighbouring points p_{i-1} and p_{i+1}, if $f(p_i) < f(p_{i-1})$ and $f(p_i) < f(p_{i+1})$, then p_i is a minimum point.
2. For point p_i and neighbouring points p_{i-1} and p_{i+1}, if $f(p_i) > f(p_{i-1})$ and $f(p_i) > f(p_{i+1})$, then p_i is a maximum point.

Evaluation phase: The less importance points from $TP[1\ldots m]$ are pushed earlier into a stack 'S'. A point's importance is calculated using the importance evaluation method, which is described in the next section. After a point is pushed into the stack, the status of the remaining points may change, and some may no longer be a minimum or maximum point in the resulting sequence. If the prior pushing action alters the status of a point $p[i]$ in the series $TP[1\ldots m]$, then $p[i]$ will be pushed into 'S' because it no longer makes a contribution to preserving the trends. In the resulting time series, these consecutive points are considered to be within a single uptrend or downtrend if they are monotonically increasing or decreasing. Fig. 8 demonstrates such a scenario. When $p2$ is pushed into 'S', it is no longer the minimum or maximum and the points p_1, p_2 and p_4 are in a single uptrend. The evaluation process continues until all TPs are pushed into the stack. The pseudocode for both phases is described in Algorithm 3. Fig. 9 illustrates a series of TPs being pushed into a stack 'S'.

Algorithm 3. Pseudocode for identifying TPs by their degree of importance.

```
Function TPs_identification_phase
        Input:T[1…n]
        Output:TP[1…m]
  For each point i in T[1…n]
        If (T[i−1] < T[i] and T[i] > T[i+1])
              put T[i] into TP
      Or (T[i−1] > T[i] and T[i] < T[i+1])
              put T[i] into TP
  End

Function TPs_evaluation_phase
```

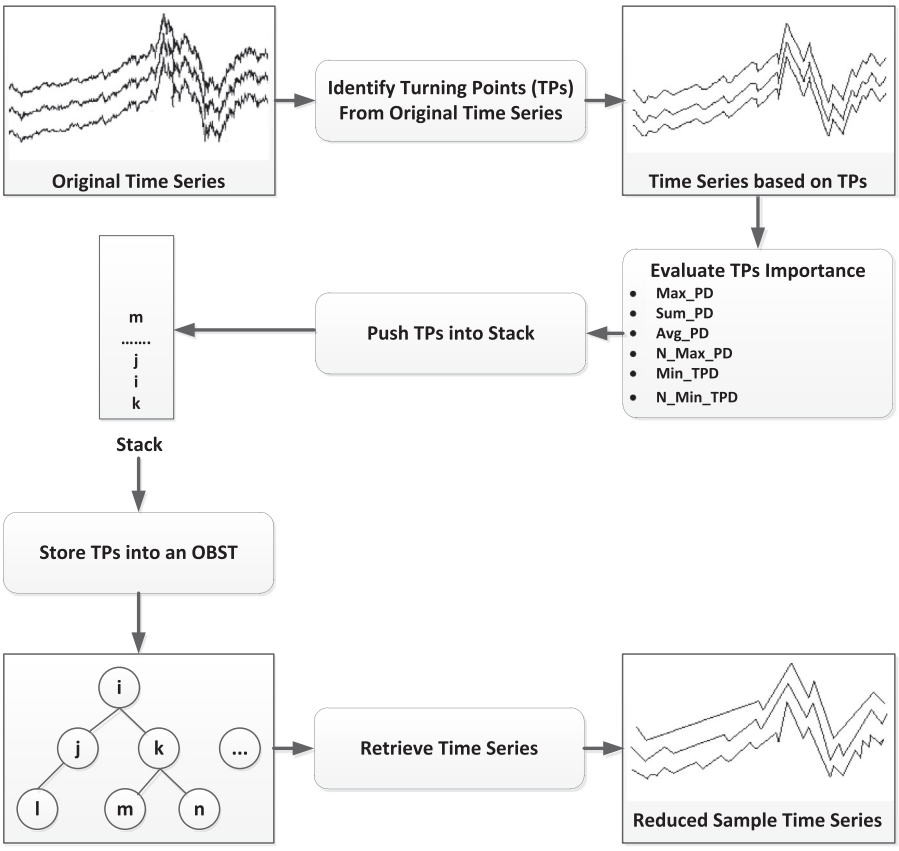

Fig. 5. Overview of the proposed method.

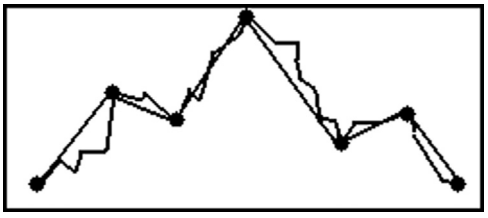

Fig. 6. Technical pattern determined by seven maximum and minimum points.

```
      Input: Sequence TP[1…m]
      Output: TPStack S[1…m]
Calculate each TP's importance value.
  Begin
      Repeat until all point in TPs are pushed into S
      Begin
Push the TP with lowest importance value into the
  Stack 'S'.
          If a point being no longer minimum or
maximum in TP[1…m] found
                Push it into S
      Update the neighbour TPs' importance
        End
  Return S
  End
```

3.2.1. TP importance evaluation

The importance of a $TP(p_i)$ can be calculated on the basis of the difference between the original series and the resulting new series if point p_i is ignored. A visual depiction of the effect of ignoring a point (e.g. point number 8) is depicted in the second step of Fig. 9.

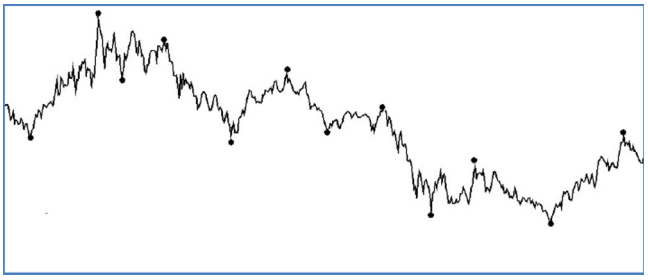

Fig. 7. Turning points in a time series.

We employ Eq. (1) as the basic distance function. To calculate a TP's degree of importance, we use its left and right neighbouring TPs from $TP[1\ldots m]$, which is obtained during the identification phase. Suppose that three consecutive TPs are $p_1(x_1, y_1)$, $p_2(x_2, y_2)$ and $p_3(x_3, y_3)$.

The first evaluation method in Eq. (2) is based on Fu et al. (2008) and is illustrated in Part 1 of Fig. 10. Suppose that $\hat{y}_i$ is a point on an approximation line that connects the maximum and minimum points among three points (p_1, p_2 and p_3). In this example, p_1 and p_3 are the maximum and minimum points, and (x_i, y_i) is the coordinate of the ith point on the original time series. The result of this calculation is the degree of importance for the median point of p_1, p_2 and p_3 (p_2 in this example). *Max_PD* (the maximum point distance) from Eq. (2) is biased towards retaining the shape of the time series over the period. Fu et al. (2008) employed this evaluation method to select important points to represent the overall shape of a time series.

$$Max\_PD = \max(|y_i - \hat{y}_i|) \tag{2}$$

The second evaluation method in Eq. (3) is based on the PLA approach in Heckbert and Garland (1997) and Keogh et al. (2001b). The difference between the first and second evaluation methods is

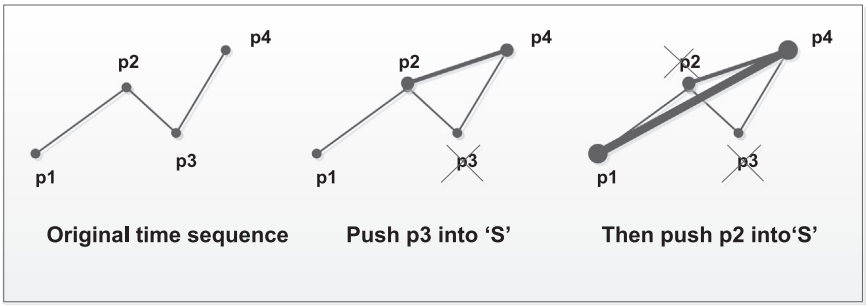

Fig. 8. Illustration of pushing less importance points into a stack.

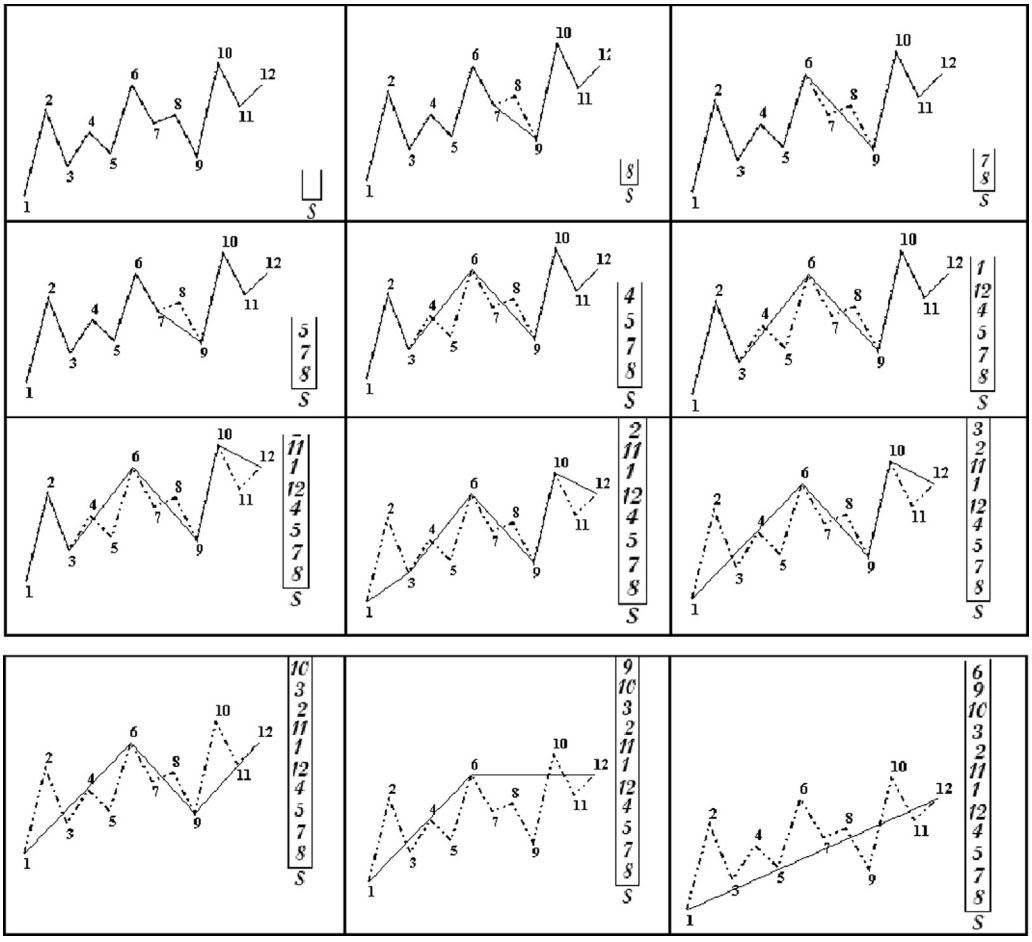

Fig. 9. A series of turning points being pushed into a stack 'S'.

that the sum of the distance is used as the degree of importance in the latter. We illustrate this evaluation method in Part 2 of Fig. 10. *Sum_PD* (the sum points distance) from Eq. (3) is biased towards retaining the least degree of error between the original and resultant time series.

$$Sum\_PD = \sum(|y_i - \hat{y}_i|) \tag{3}$$

The third evaluation method in Eq. (4) is based on Jiang et al. (2007) and is illustrated in Part 3 of Fig. 10. In this method, the *Sum_PD* from the previous method is multiplied by the inverse of the duration of segment $(x_3 - x_1)$.The result of this calculation, called *Avg_PD* (average points distance), is the degree of importance of the median point of p_1, p_2 and p_3 (p_2 in the example). This method is biased towards keeping the resulting time series as smooth as possible. The equation is used to merge the over-fitting segments in the PIP approach (Jiang et al., 2007).

$$Avg\_PD = \frac{1}{(x_3 - x_1)}\sum(|y_i - \hat{y}_i|) \tag{4}$$

The fourth evaluation method is given in Eq. (5) and illustrated in Part 4 of Fig. 10. In this evaluation, we multiply the duration of segment $(x_3 - x_1)$ with *Max_PD* from Eq. (2). The calculation result is the degree of importance of the median point of p_1, p_2 and p_3 (p_2 in the example). *N_Max_PD* in Eq. (5) considers not only the shape of the time series but also the duration.

$$N\_Max\_PD = (x_3 - x_1) \times \max(|y_i - \hat{y}_i|) \tag{5}$$

The fifth evaluation method is given in Eq. (6) and illustrated in Part 5 of Fig. 10. y_k ($k \in \{1, 2, 3\}$) is the actual y value of the three consecutive TPs (p_1, p_2 and p_3), and $\hat{y}_k$ is the y value of the point on the approximation line defined by the other two TPs. The result of

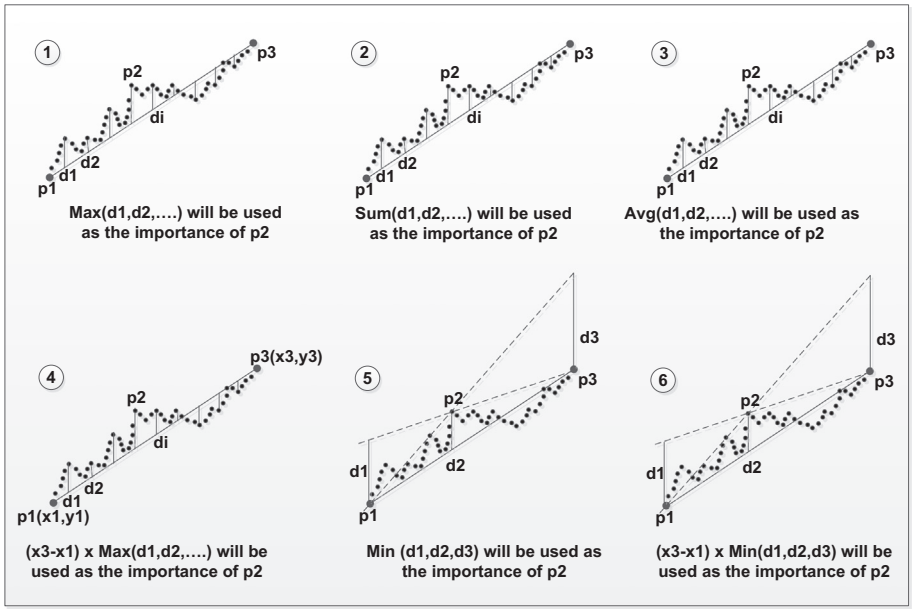

Fig. 10. Importance evaluation for Eqs. (2)–(7) corresponding to Parts 1–6, where $p_1 < p_2 < p_3$.

this calculation is the degree of importance of p_2. *Min_TPD* (minimum TP distance) in Eq. (6) minimises the fluctuation changes.

$$Min\_TPD = \min(|y_k - \hat{y}_k|) \tag{6}$$

The sixth evaluation method is given in Eq. (7) and illustrated in Part 6 of Fig. 10. We multiply the duration of segment $(x_3 - x_1)$ with *Min_TPD* from Eq. (6). The result of this calculation is the degree of importance of p_2. *N_Min_TPD* from Eq. (7) takes into account both the effect of the fluctuation and the duration of the segment:

$$N\_Min\_TPD = (x_3 - x_1) \times \min(|y_k - \hat{y}_k|) \tag{7}$$

3.3. Storage and retrieval

In this section, we describe an approach for the storage of TPs with respect to their degree of importance in an OBST (the third part of Fig. 5). An algorithm is also devised to reconstruct the reduced-sample time series from this tree (the final part of Fig. 5).

3.3.1. Optimal binary search tree

A binary tree with the smallest average retrieval cost is called an OBST (Cormen et al., 2009). Consider, for instance, four data points, p_1, p_2, p_3 and p_4, whose corresponding weights are 0.1, 0.2, 0.3 and 0.4. Suppose that two different tree structures (see Fig. 11) are used to store these points. The calculation of average retrieval cost, Eq. (8), is defined (with w_i and d_i are the weight and the depth of the element), and the average retrieval cost of the first tree is $(0.1*1+0.2*2+0.3*3+0.4*4)/(0.1+0.2+0.3+0.4)=3.0$ and that of the second is $(0.1*2+0.2*1+0.3*2+0.4*3)/(0.1+0.2+0.3+0.4)=2.2$. Because the average cost of the second tree is smaller than that of the first, the second tree is considered to be more efficient in organising these four points and thus is the OBST.

$$Cost = \sum_{1}^{n} w_i \times d_i \tag{8}$$

Any subtree of an OBST is also proved to be optimal (Cormen et al., 2009). To find the solution for an OBST, the optimal solution for its subtrees must first be guaranteed. To do so, we can find the value of an optimal solution in a recursive manner. Suppose that

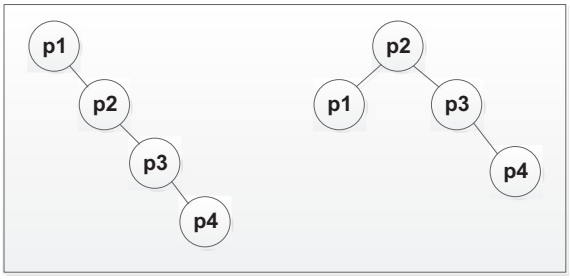

Fig. 11. Two different tree structures.

Table 2
Table for building OBST (Cormen et al., 2009).

	0	1				J	n
1	0	w_1					$C[1, n]$
		0	w_2				
i			0	w_3		$C[i, j]$	
				0			
					0		
						0	w_n
$n+1$							0

picking a sub-problem domain is the same as finding an OBST containing elements $p_i, \ldots, p_k \ldots, p_j$ with $i \le k \le j$. If p_k is the root of the subtree, then the left subtree will contain $p_i \ldots p_{k-1}$, and the right subtree will contain $p_{i+1} \ldots p_j$. Thus, the overall goal is now divided into two sub-problems. The base situation is $i=k=j$, which is only one element in the group, and the root is itself.

In practice, Table 2 can be used to calculate the smallest average retrieval cost for a group of elements. $C[i, i-1]$ and $C[i, i]$ are initially assigned 0 and w_i (element weights), respectively (the base situation). Then, $C[i, j]$ (the sub-problem) is recursively calculated through Eq. (9). The value in $C[1, n]$ is the smallest average retrieval cost and the OBST can be achieved by tracking the calculation process activities in the table. The detailed pseudocode is illustrated in the next section, and the proof of the

Table 3
TP weights for the example in Fig. 9.

1	2	3	4	5	6	7	8	9	10	11	12
0.46	0.62	0.69	0.31	0.23	0.92	0.15	0.08	0.85	0.77	0.54	0.38

optimum nature of this type of tree can be found in Cormen et al. (2009).

$$C[i,j] = \min_{1 \le k \le j}\{C[i,k-1], C[k+1,j]\} \sum_{s=i}^{j} w_s \quad (9)$$

The structure for storing the TPs of time series should satisfy two requirements: (1) the points can be retrieved more easily if they make a greater contribution to preserving the shape and trends of the time series, and (2) the structure should not alter the original time order of the TPs. In an OBST, the elements with larger weights are usually found near the top of the tree and thus are retrieved first. The OBST renders tree traversal, including pre-order, in-order and breadth first traversal, more flexible. Moreover, an OBST can also keep the average retrieval cost low, and its subtrees also maintain these properties. Thus, an OBST is efficient at storing the TPs of a time series.

3.3.2. Storing Turning Points in an OBST

A weight should first be assigned to each TP based on its contribution to preserving the shape of the time series. Intuitively, the TPs near the top of the stack generated in the evaluation phase will yield larger weights than those near the bottom, as the former are of greater importance. Suppose that the weights of the TPs identified in Fig. 9 are $w_1 \ldots w_m$ and are calculated on the basis of Eq. (10), where x is the distance from w_i to the top of the stack and m is the stack size. Table 3 presents the weights of the TPs in the stack in Fig. 9.

$$w_i = \frac{m-x}{m+1} \quad (10)$$

To calculate an optimum retrieval cost for $C[i, j]$ using Eq. (9), the cost value is obtained by taking the minimum value from i to j (see $\min_{i \le k \le j}$) in the equation. Each time that we obtain the minimum cost, we record the index (for instance 'k') in a root table $R[i, j]$. The pseudocode in Algorithm 4 takes the weights $w[1 \ldots m]$ as the input, and the output is the root table 'R' that records the tree structure.

An OBST can be constructed by tracking the index in 'R'. Suppose that all of the nodes in this tree have the same size: the basic information of the x- and y-coordinates of the TP and the two pointers to the left and right child. If element $x = R[i, j]$ is a node of the OBST, then the left node is $x' = R[i, x-1]$ and the right node is $x' = R[x+1, j]$. At the beginning, the root node content x is $R[1, n]$. The pseudocode is demonstrated in Algorithm 5, through which Table 4 (the root table) can be generated and the OBST obtained for the example in Fig. 9.

Algorithm 4. Pseudocode for obtaining a root table (Cormen et al., 2009).

```
Function OPTIMAL-BST
    Input: w1…wm
    Output: R
For i=1 to n//do initialisation
    C[i, i−1] = 0;
    C[i, i] = w[i];
    R[i, i] = i;
    For d=1 to n-1//diagonal calculation
        For i=1 to n−d
            j=i+d
              For k=i to j
                If {C[i, k−1], C[k+1, j]} + Σ_{s=i}^{j} w_s is minimum
                  C[i, j] = {C[i, k−1], C[k+1, j]} + Σ_{s=i}^{j} w_s
                  R[i, j] = k;
            End
        End
    End
End
Return R
```

Table 4
Root table for TPs in the example in Fig. 9.

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	2	2	3	3	3	3	6	6	6	6
2	0	2	3	3	3	3	3	3	6	6	6	6
3		0	3	3	3	6	6	6	6	6	6	6
4			0	4	4	6	6	6	6	9	9	9
5				0	5	6	6	6	6	9	9	9
6					0	6	6	6	6	9	9	9
7						0	7	7	9	9	10	10
8							0	8	9	9	10	10
9								0	9	9	10	10
10									0	10	10	11
11										0	11	11
12											0	12
13												0

Algorithm 5. Pseudocode for creating OBST.

```
Function Create_Tree(T)
        Input: root table R, T[1…m];
        Output: Tree root
    Initialisation:
Create cnode
    index=R[1][n]
    cnode.value=T[index]
    root=cnode
    pnode=root
Sub-tree-construction (pnode, R[1][n])

Function Sub-tree-construction(pnode, R[i][j])
k=R[i][j]
if (R[i][k-1] is an index of time series sequence)
Create cnode
    index=R[i][k-1]
    cnode.value=T[index]
    pnode→left=cnode
    Sub-tree-construction(cnode, R[i][k-1])
Else if (R[k+1][j] is an index of time series
  sequence)
Create cnode
    index=R[k+1][j]
    cnode.value=T[index]
    pnode→right=cnode
    Sub-tree-construction(cnode, R[k+1][j])
Else
    Return
```

From Fig. 12, we can show that an OBST holds two important properties: (1) the turning points can be retrieved earlier from OBST if they make a greater contribution to preserving the shape and trends of the time series, (2) the OBST can still maintain the original chronological order of the turning points if we use depth first traversal strategy for retrieval. If we compare Figs. 9 and 12, we can clearly see the advantage of the time order kept by the OBST. For instance, retrieving points 1–6 only requires visiting half of the tree from Fig. 12, whereas in Fig. 9, we need to pop off nearly the entire stack.

3.4. Example: Technical pattern matching in reduced-sample time series

Predicting future stock movement based on technical patterns is one of the most common techniques used by financial analysts. The width of a technical pattern can be defined as the total number of days between the leftmost and the rightmost data points of the pattern. In this section, we describe how a technical pattern with different widths (durations) can be found in the reduced-sample time series. This capability is very useful since a financial analyst can quickly locate technical patterns of desired size by selecting appropriate reduce-sample time series. For instance, a time series with low sample rate can be used to find longer duration technical patterns and vice versa, a time series with high sample rate can be used to find shorter duration technical patterns.

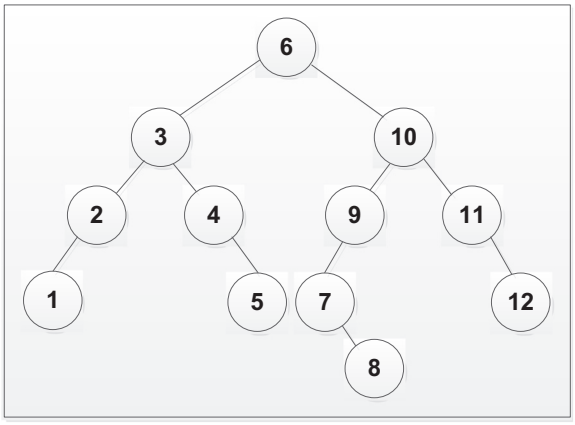

Fig. 12. OBST building for the example in Fig. 9.

We select the Hang Seng Index (HSI) from 3 January 2000 to 25 May 2010 for our example. Our dataset for the example is taken from the Yahoo (2013) Web site. First, 2586 turning points are extracted from the original time series and stored in a stack in the order of decreasing importance based on *Max_PD* evaluation method. Next, we prepare three reduced-sample time series by popping different numbers of points for each time series from top of the stack. By controlling the number of points to be popped from the stack, user can easily control the level of sample rate for the time series to be reconstructed. For our case, 324, 654, and 1035 turning points are popped out from the same stack to reconstruct three reduced-sample time series. The resulted time series contain only 75%, 50%, and 25% data points compared to the total number of turning points in the original time series.

Next, we use a modified version of rule-based and template-based pattern matching methods proposed by Fu (Fu et al., 2007) to locate Head-and-Shoulder technical patterns in these time series. Our revised algorithm performs following steps:

1. Reconstruct the reduced-sample time series based on the turning points from the stack.
2. Set the size of Sliding Window (SW) to n (e.g. $n=7$ for the current example).
3. SW is gradually shifted from the first to the last point in the reduced sample time series. This step will generate all subsequences of size equal to n.
4. Normalise each subsequence and the Head-and-Shoulder technical pattern.
5. Use Template-based and Rule-based method to check if the subsequence is similar to the technical pattern.
6. Return the patterns which are recognised by both rule-based and template-based pattern matching methods.

Table 5
Seven Head-and-Shoulder patterns found the reduced-sample time series.

	Pattern 1			**Pattern 2**			**Pattern 3**		
	Point ID	Date	Stock price	Point ID	Date	Stock price	Point ID	Date	Stock price
Reduced-sample time series with 75% sample rate	142	31/07/2000	16,840.98	1710	28/11/2006	18,639.53	2145	22/08/2008	20,392.06
	144	02/08/2000	17,277.39	1712	30/11/2006	18,960.48	2146	25/08/2008	21,104.79
	145	03/08/2000	17,274.28	1713	01/12/2006	18,690.82	2147	26/08/2008	21,056.66
	147	07/08/2000	17,727.25	1716	06/12/2006	19,026.36	2148	27/08/2008	21,464.72
	149	09/08/2000	17,181.99	1718	08/12/2006	18,739.99	2149	28/08/2008	20,972.29
	150	10/08/2000	17,333.21	1719	11/12/2006	18,924.66	2150	29/08/2008	21,261.89
	152	14/08/2000	16,998.06	1721	13/12/2006	18,718.19	2151	01/09/2008	20,906.31
	Pattern 4								
	Point ID	Date	Stock Price						
Reduced-sample time series with 50% sample rate	316	17/04/2001	12,606.45						
	325	02/05/2001	13,814.24						
	334	15/05/2001	13,250.09						
	339	22/05/2001	13,877.95						
	347	01/06/2001	13,141.38						
	352	08/06/2001	13,808.89						
	360	20/06/2001	12,918.71						
	Pattern 5			**Pattern 6**			**Pattern 7**		
	Point ID	Date	Stock Price	Point ID	Date	Stock Price	Point ID	Date	Stock Price
Reduced-sample time series with 25% sample rate	248	03/01/2001	14,589.58	457	12/11/2001	10,592.45	2373	22/07/2009	19,248.17
	252	09/01/2001	15,500.59	467	26/11/2001	11,391.96	2377	28/07/2009	20,624.54
	254	11/01/2001	15,090.77	469	28/11/2001	11,066.19	2378	29/07/2009	20,135.5
	266	01/02/2001	16,163.99	494	07/01/2002	11,892.64	2387	11/08/2009	21,074.21
	281	22/02/2001	15,098.64	498	11/01/2002	11,166.46	2393	19/08/2009	19,954.23
	282	23/02/2001	15,280.56	499	14/01/2002	11,209.43	2396	24/08/2009	20,535.94
	287	02/03/2001	13,966.43	517	07/02/2002	10,409.68	2402	01/09/2009	19,872.3

The difference between our algorithm and Fu's approach (Fu et al., 2007) is that, in our algorithm, segmentation (approximation) based on turning points is performed before a sliding window is used to find the subsequences. Whereas in Fu's approach, original time series is used for finding subsequences by a sliding window and PIP (Chung et al., 2001; Fu et al., 2008) approach is then used to segment the resulting subsequences. Seven Head-and-Shoulder patterns found by our revised algorithm are listed in Table 5.

For illustration purposes, we select one pattern from each time series. The visualisations of selected Pattern 2, 4, and 5 are shown in Figs. 13–15. Since the window size is set to 7 in our algorithm, all the subsequences found contain exactly 7 turning points. Both the original time series from Hang Seng Index (HSI) and the Head-and-Shoulder patterns formed by turning points are plotted in Figs. 13–15. The total number of actual trading days (width) of patterns 2, 4, and 5 are 12, 44, and 43 respectively. Note that the width of pattern 5 is similar to that of pattern 4 since Stock Market in Hong Kong is closed during Lunar New Year holiday's period.

4. Experimental results

In our experiment testing the proposed approach, we evaluate the performance of six equations for measuring the TPs of a time series –– *Max_PD*, *Min_TPD*, *Sum_PD*, *Avg_PD*, *N_Max_PD* and *N_Min_TPD* – and compare them with those in two other representation approaches, PIP and PLA. We revise the PLA approach by storing the selected points in a stack. We first demonstrate the visualisation of the time series, and then compare the performance of the three approaches in terms of accuracy and trend preservation.

Our experimental dataset is taken from the Yahoo Finance (Yahoo, 2013) Web site. We select the Hang Seng Index (HSI) from 30 March 2000 to 6 June 2010 for evaluation. In Figs. 16–18 we plot the time series using only 2%, 3% and 5% of the data points, respectively.

In Fig. 19, we compare the reconstruction error using different numbers of points. The error is also defined as the sum of the distance for every point between the original time series and the series generated by the TP, PIP and PLA approaches, and is calculated using Eq. (11). Fig. 19 shows that the value of the reconstruction error generated by any of these methods decreases sharply at the beginning. All three generate a similar error-decreasing curve.

In Fig. 20, we compare the number of up and down trends preserved by the proposed approach and the PIP and PLA approaches with different numbers of points. The number of trends is equivalent to the total number of local minimum/maximum points in the time series. We find the six aforementioned importance evaluation methods (*Max_PD*, *Min_TPD*, *Sum_PD*, *Avg_PD*, *N_Max_PD* and *N_Min_TPD*) based on turning points (TP) to achieve a very similar reconstruction error and trend performance. In Fig. 20, we can also observe that PLA and PIP approaches maintain less trends compared to the six importance evaluation methods from the TP approach.

For illustration purposes, we also compare the reconstruction error of *Sum_PD* in our approach with that in the PIP and PLA approaches in Figs. 21 and 22, respectively. We find the PLA approach achieve lower reconstruction error compared to TP approach with *Sum_PD* importance evaluation method and PIP approach. In Fig. 22, we can also observe that TP approach with *Sum_PD* importance evaluation method achieves highest

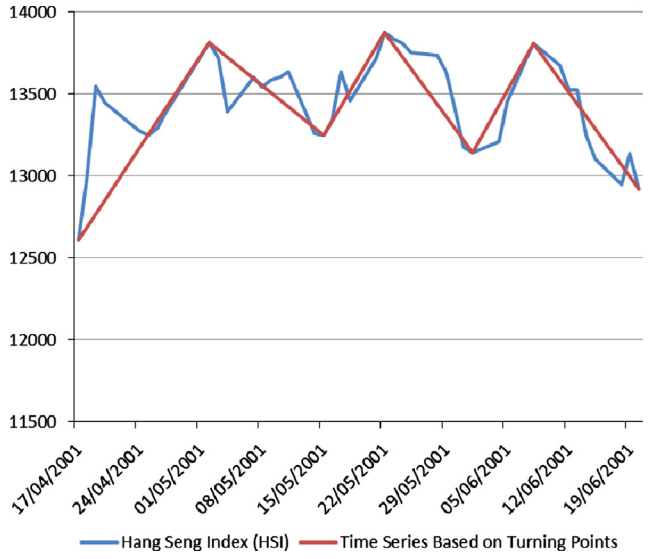

Fig. 14. Pattern 4 found from the reduced-sample time series with 50% sample rate.

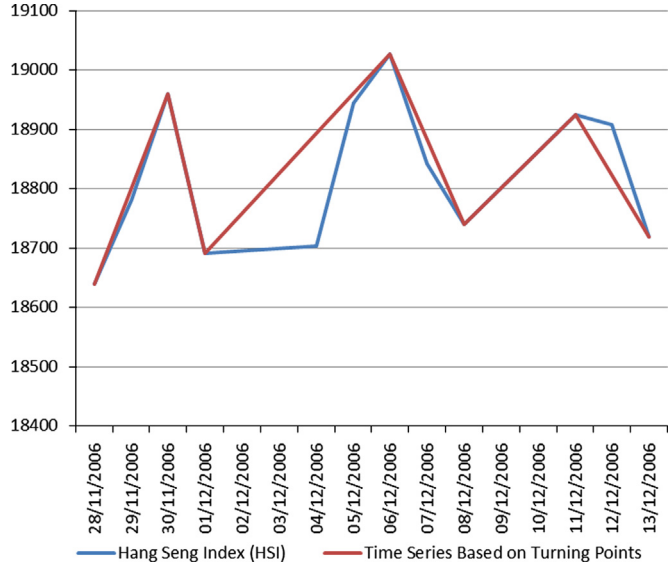

Fig. 13. Pattern 2 found from the reduced-sample time series with 75% sample rate.

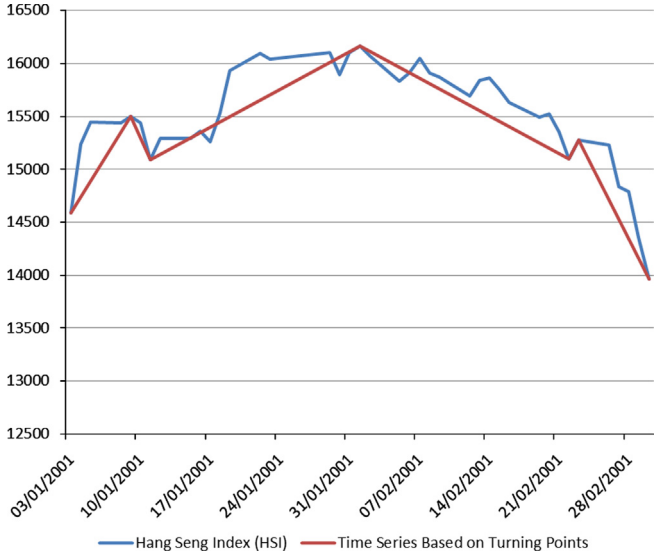

Fig. 15. Pattern 5 found from the reduced-sample time series with 25% sample rate.

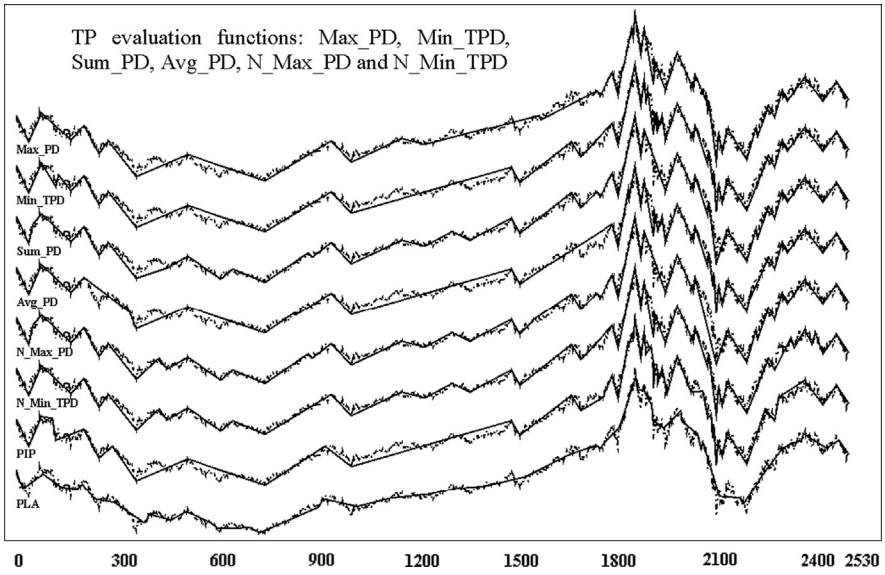

Fig. 16. Representing Hang Seng Index with 50 points (2%): (TP, PIP and PLA methods).

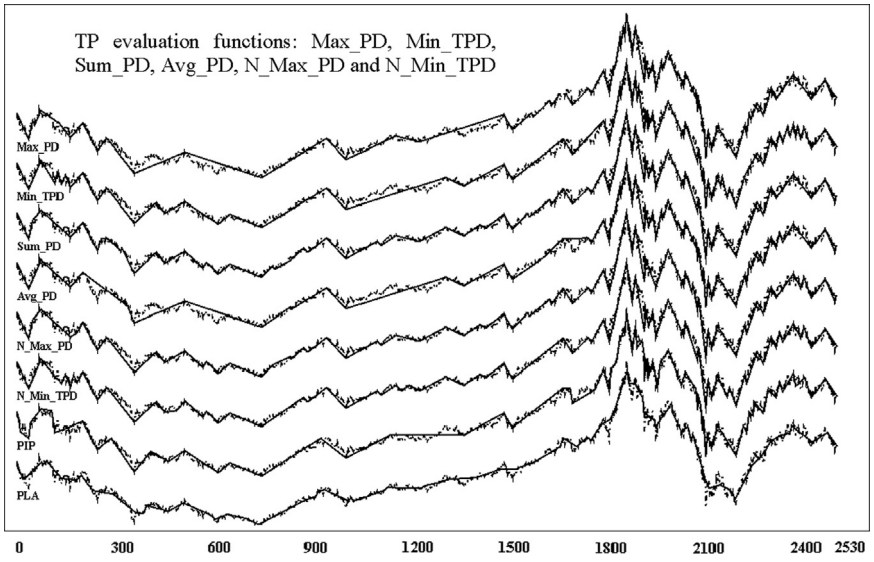

Fig. 17. Representing Hang Seng Index with 76 points (3%): top (TP method); bottom (PIP and PLA methods).

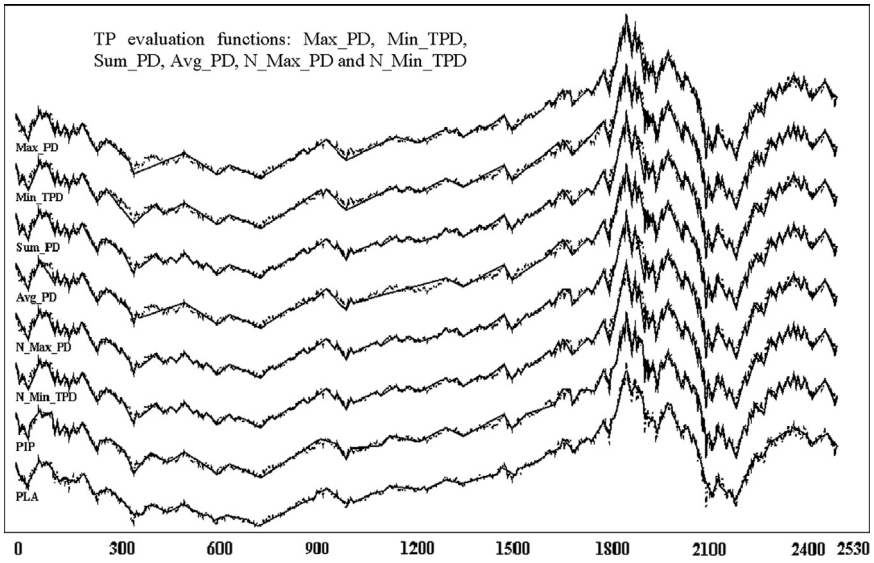

Fig. 18. Representing the Hang Seng Index 127 points (5%): top (TP method); bottom (PIP and PLA methods).

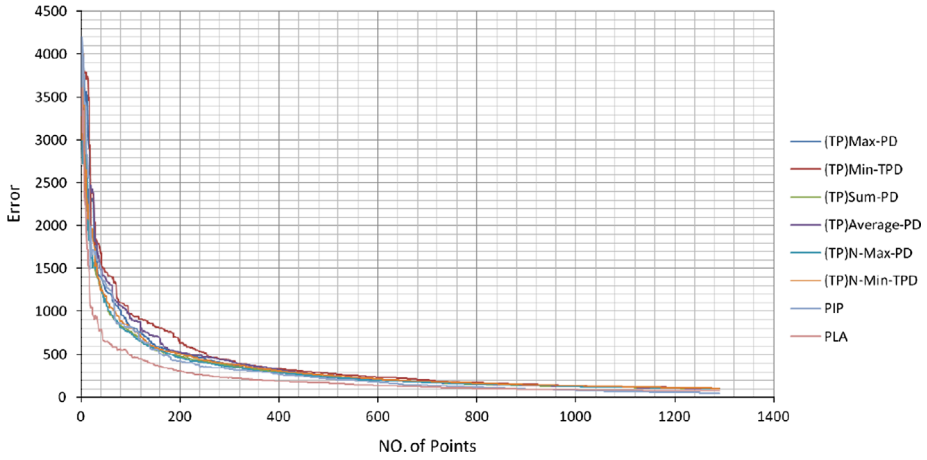

Fig. 19. Error with comparison to number of points.

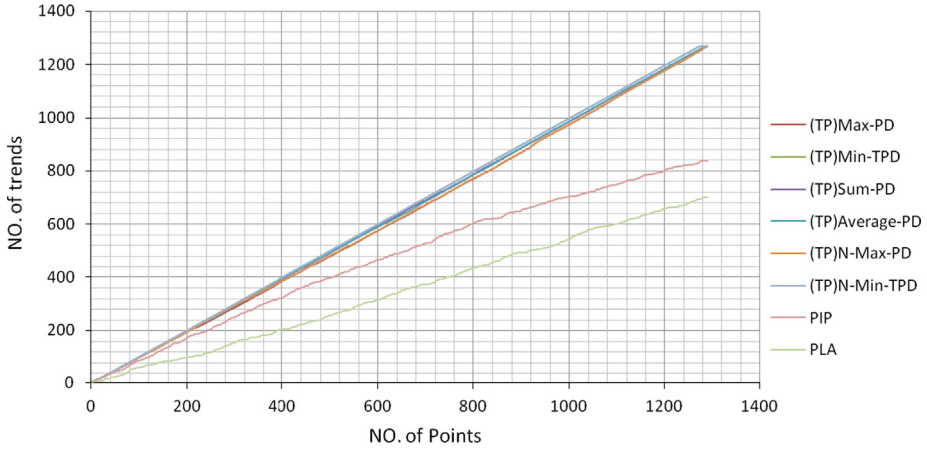

Fig. 20. Trends with comparison to number of points.

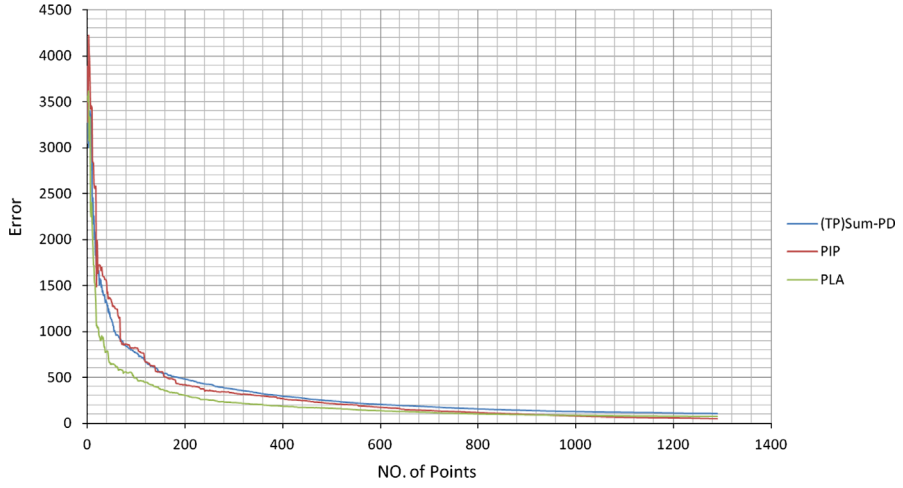

Fig. 21. Error with comparison to number of points.

performance in preserving trends.

$$error = \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{11}$$

In summary, comparing the error and trends produced by the three methods, we find the PLA approach to produce the least amount of error and the fewest trends. The proposed TP approach is able to preserve more trends than either the PLA or PIP approach. These results are unsurprising, as the PIP approach is designed to retain the overall shape of the time series, whereas one of the main objectives of the PLA approach is to represent the time series while minimising the error. In contrast, the objective of our approach is not only to keep the overall movement of the time series but also to preserve as many of its trends as possible.

With regard to the complexity of our method, in the identification phase we scan only the time series data from the beginning to the end. Its complexity is thus $O(n)$. In the evaluation phase, the algorithm is a bottom-up process, and the worst case is $O(n^2)$

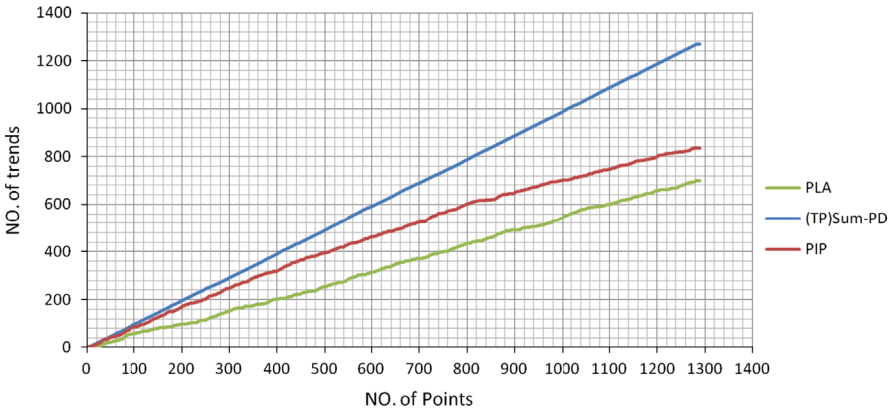

Fig. 22. Trends with comparison to number of points.

(refer to Algorithm 3). The method's overall complexity in identifying TPs by their degree of importance is $O(n^2)$.

However, because we initially reduce the time points at the beginning by selecting the TPs from the original time series, the data space is considerably reduced in the later stage. Financial analysts may employ the extracted turning points and trends to locate technical patterns such as head-and-shoulder or double tops to predict stock movements. Several applications which make use of extracted important points and segmentation process is demonstrated in a number of articles: feature points extraction and technical pattern matching (Fu et al., 2007; Zhang et al., 2010a), locating Up-Trapeziform, Bottom-Trapeziform, Up-flag, and Down-flag patterns from the bid and trade time series of Chicago stock market (Zhang et al., 2007), locating past trends in the stock data (Lavrenko et al., 2000), and predicting the future stock price (Kwon and Sun, 2011). In these applications, segmentation and important points extraction is performed as a preprocessing step.

Our experiment was conducted on a computer with Intel® Core™ 2 Duo CPU, T6500 at 2.10 GHz and 2.69 GB Memory. Our prototype programme is developed in jdk1.6.0 and MySQL Server 5.1 is installed to provide database service.

We also compare the processing time for *Sum_PD* in our approach with that in the PIP and PLA. Specifically, the execution time of *Sum_PD* approach is 406 ms, PIP approach is 1329 ms, and PLA is 1515 ms. The processing time of *Sum_PD* is much smaller than that of other two approaches. The reason behind the faster processing time is that, *Sum_PD* in our approach only evaluates the importance of identified TPs in the first phase, while PIP and PLA take into account all points.

5. Conclusion

In this paper, we investigate the properties of TPs in preserving the trends and fluctuation of time series. However, as TPs do not contribute equally to the overall shape of the time series, we introduce a novel method for selecting TPs based on their degree of importance, which is calculated according to their contribution to preserving the trends and shape of that series. We also store TPs in an OBST, which allows the more important ones to be retrieved first, thereby reducing the average retrieval cost. Such a capability is extremely useful for trend analysis and for analysing stock data in a top-down fashion.

In analysing the performance of our method, we primarily use the PIP approach as a benchmark because it also directly selects points from the original time series for representation. We find the time series generated by the proposed method to maintain the shape of the original time series very well. Its major advantage is that it is able to preserve more monotonous trends due to the TPs' ability to identify trend changes in the time series. In addition, the proposed method is based on local comparison, which can be used directly for online representation and real-time applications, such as pattern detection and the search for subsequences. Furthermore, TPs render the trend changes in time series very easy for users to understand.

TPs are important features of a time series and it would also be possible to use them for similar sequence searching, trend comparisons, and online pattern detection. As for the future work, we are planning to use the similarity of the segments generated through these TPs to predict the direction of stock movements based on probabilistic reasoning. We are also planning to apply data mining algorithms on the represented time series, such as extracting similar segments from historical price data, to train a neural network model to predict a potential trading point for a buy/sell action.

Acknowledgement

This research is funded by the Research Committee of University of Macau, under grant MYRG041(Y1-L1)-FST13-SYW. Authors thank Gong Xueyuan for the implementation of rule-based and template-based technical pattern matching programs for the article.

Appendix

The sliding window algorithm (Keogh et al., 2001b) for segmentation with PLA is described in Algorithm 6. Beginning with the first data point from the time series, an anchor point is used to denote the left end point of a potential segment. The segment is gradually extended to the right until the error of the potential segment is greater than the error threshold (*max_error*) defined by the user. Once the error threshold is reached, two data points from the time series at anchor and $i-1$ are connected to become a subsequence. Once a subsequence is found, the anchor point is shifted to the point *i*. The process continues until the input time series has been segmented.

Algorithm 6. The generic sliding window algorithm (Keogh et al., 2001b).

```
Function Seg_TS=Sliding_Window(T, max_error)
    anchor=1;
// Find best place to split the time series.
While not finished segmenting time series
i=2;
```

```
    while calculate_error(T[anchor: anchor+i])
  <max_error
        i=i+1;
      end;
     Seg_TS=concat(Seg_TS, create_segment(T
  [anchor: anchor+(i-1)]);
      Anchor=anchor+I;
end;
```

The bottom-up algorithm for PLA (Keogh et al., 2001b) is described in Algorithm 7. The function *create_segment* in Algorithm 7 generates a linear segment approximation of the input time series. The first *for* loop uses $n/2$ segments to approximate the input time series of length n. The next *for* loop is used to calculate the cost of merging each pair of adjacent segments from the previous step. The *calculate_error* function in Algorithm 7 returns the approximation error of the linear segment approximation of the input time series. The next *while* loop in the algorithm merges the lowest cost pairs until the cost of merging exceeds the error threshold defined by the user. The costs of the segments affected by the merging process are also updated subsequently.

Algorithm 7. The generic bottom-up algorithm for PLA (Keogh et al., 2001b).

```
Function Seg_TS=Bottom-UP(T, max_error)
for i=1: 2: length(T) // Create initial fine
  approximation.
Seg_TS=concat(Seg_TS, create_segment(T[i:
  i+1]));
end;
// Find cost of merging each pair of segment.
for i=1: length(Seg_TS)-1
    merge_cost(i)=calculate_error([merge(Seg_TS
  (i), Seg_TS(i+1))]);
end;
while min(merge_cost)<max_eror
index=min(merge cost); //Find cheapest pair to
  merge
Seg_TS(index)=merge(Seg_TS(index), Seg_TS
  (index+1))); //Merge
delete(Seg_TS(index+1)); //Update records.
merge_cost(index)=calculate_error(merge(Seg_TS
  (index), Seg_TS(index+1)));
merge_cost(index-1)=calculate_error(merge
  (Seg_TS(index-1), Seg_TS(index+1)));
end;
```

References

Agrawal, R., Faloutsos, C., Swami, A.N., 1993. Efficient similarity search in sequence databases. In: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, pp. 69–84.

Atsalakis, G.S., Valavanis, K.P., 2009. Surveying stock market forecasting techniques – Part II: soft computing methods. Expert Systems with Applications 36 (3), 5932–5941.

Bao, D., Yang, Z., 2008. Intelligent stock trading system by turning point confirming and probabilistic reasoning. Expert Systems with Applications 34 (1), 620–627.

Belkin, M., Niyogi, P., 2001. Laplacian Eigenmaps and spectral techniques for embedding and clustering. In: Proceedings of the Neural Information Processing Systems. MIT Press, Vancouver, British Columbia, Canada, pp. 586–691.

Chakrabarti, K., Keogh, E., Mehrotra, S., Pazzani, M., 2002. Locally adaptive dimensionality reduction for indexing large time series databases. ACM Transaction on Database Systems 27 (2), 188–228.

Chan, K.-P., Fu, A.W.-C., 1999. Efficient time series matching by wavelets. In: Proceedings of the 15th International Conference on Data Engineering, pp. 126–134.

Chang, P.-C., Fan, C.-Y., Liu, C.-H., 2009. Integrating a piecewise linear representation method and a neural network model for stock trading points prediction. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications 39 (1), 80–92.

Chu, K.K.W., Wong, M.H., 1999. Fast time-series searching with scaling and shifting. In: Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 237–248.

Chung, F.-l., Fu, T.-c., Luk, R., Ng, V., 2002. Evolutionary time series segmentation for stock data mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, pp. 83–90.

Chung, F.L., Fu, T.C., Luk, R., Ng, V., 2001. Flexible time series pattern matching based on perceptually important points. In: Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Learning from Temporal and Spatial Data, pp. 1–7.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to Algorithms, hird ed. The MIT Press, Cambridge, MA.

Frost, A.J., Prechter, J., Robert R., 2001. Elliott Wave Principle: Key to Market Behavior, first ed. Wiley, Chichester, England.

Fu, T.-c., 2011. A review on time series data mining. Engineering Applications of Artificial Intelligence 24 (1), 164–181.

Fu, T.-c., Chung, F.-l., Luk, R., Ng, C.-m., 2007. Stock time series pattern matching: template-based vs. rule-based approaches. Engineering Applications of Artificial Intelligence 20 (3), 347–364.

Fu, T.-c., Chung, F.-l., Luk, R., Ng, C.-m., 2008. Representing financial time series based on data point importance. Engineering Applications of Artificial Intelligence 21 (2), 277–300.

Fuchs, E., Gruber, T., Nitschke, J., Sick, B., 2010. Online segmentation of time series based on polynomial least-squares approximations. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (12), 2232–2245.

Hamilton, W.P., 2010. The Stock Market Barometer: A Study of its Forecast Value Based on Charles H. Dow's Theory of the Price Movement. With an Analysis of the Market and its History Since 1897, first ed. Nabu Press, New York.

Heckbert, P.S., Garland, M., 1997 Survey of polygonal surface simplification algorithms. In: Multiresolution Surface Modeling Course SIGGRAPH '97. Carnegie Mellon University. ⟨http://ftp.cs.cmu.edu/afs/cs/project/anim/ph/paper/multi97/release/heckbert/simp.pdf⟩ (last accessed 18.06.2013).

Jenkins, O.C., Mataric, M.J., 2004. A spatio-temporal extension to Isomap nonlinear dimension reduction. In: Proceedings of the 21st International Conference on Machine Learning. ACM, New York, NY, pp. 441–448.

Jiang, J., Zhang, Z., Wang, H., 2007. A new segmentation algorithm to stock time series based on PIP approach. In: Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, pp. 5609–5612.

Kahveci, T., Singh, A.K., 2001. Variable length queries for time series data. In: Proceedings of the 17th International Conference on Data Engineering, pp. 273–282.

Kamijo, K., Tanigawa, T., 1990. Stock price pattern recognition-a recurrent neural network approach. In: Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks, Neural Networks, pp. 215–221.

Kanth, K.V.R., Agrawal, D., Singh, A., 1998. Dimensionality reduction for similarity searching in dynamic databases. In: Proceedings of 1998 ACM SIGMOD International Conference on Management of Data, pp. 166–176.

Keogh, E., 1997. Fast similarity search in the presence of longitudinal scaling in time series databases. In: Proceedings of the 9th International Conference on Tools with Artificial Intelligence, pp. 578–584.

Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S., 2001a. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems 3 (3), 263–286.

Keogh, E., Smyth, P., 1997. A probabilistic approach to fast pattern matching in time series databases. In: Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining, pp. 20–24.

Keogh, E.J., Chu, S., Hart, D., Pazzani, M.J., 2001b. An online algorithm for segmenting time series. In: Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society, pp. 289–296.

Keogh, E.J., Pazzani, M.J., 1999. Relevance feedback retrieval of time series data. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, San Jose, CA, pp. 183–190.

Keogh, E.J., Pazzani, M.J., 2000. A simple dimensionality reduction technique for fast similarity search in sarge time series databases. In: Terano, T., Liu, H., Chen, A.P. (Eds.), Knowledge Discovery and Data Mining. Current Issues and New Applications. Springer, Berlin, Heidelberg, pp. 122–133.

Kwon, Y.-K., Sun, H.-D., 2011. A hybrid system integrating a piecewise linear representation and a neural network for stock prediction. In: 2011 6th International Forum on Strategic Technology (IFOST), pp. 796–799.

Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., Allan, J., 2000. Mining of concurrent text and time series. In: Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining, pp. 37–44.

Lawrence, N.D., 2003. Gaussian process latent variable models for visualisation of high dimensional data. In: Thrun, S., Saul, L.K., Scholkopf, B. (Eds.), Advances in Neural Information Processing Systems. MIT Press, Vancouver and Whistler, British Columbia, Canada.

Lawrence, N.D., Quinonero-Candela, J., 2006. Local distance preservation in the GP-LVM through back constraints. In: Proceedings of the 23rd International Conference on Machine learning, pp. 513–520.

Lewandowski, M., Martinez-del-Rincon, J., Makris, D., Nebel, J.-C., 2010. Temporal extension of Laplacian Eigenmaps for unsupervised dimensionality reduction of time series. In: Proceedings of the 2010 20th International Conference on Pattern Recognition, pp. 161–164.

Park, S., Chu, W.W., Yoon, J., Hsu, C., 2000. Efficient searches for similar subsequences of different lengths in sequence databases. In: Proceedings of the 16th International Conference on Data Engineering, pp. 23–32.

Pavlidis, T., Horowitz, S.L., 1974. Segmentation of plane curves. IEEE Transactions on Computers C 23 (8), 860–870.

Pearson, K., 1901. On lines and planes of closet fit to systems of points in space. Philosophical Magazine, 559–572.

Perng, C.S., Wang, H., Zhang, S.R., Parker, D.S., 2000. Landmarks: a new model for similarity-based pattern querying in time series databases. In: Proceedings of 16th International Conference on Data Engineering, pp. 33–42.

Pratt, K.B., 2001. Locating Patterns in Discrete Time-Series (Master thesis). Department of Computer Science and Engineering, University of South Florida, Tempa, FL. ⟨http://www.cs.cmu.edu/∼eugene/research/full/kevin-thesis.pdf⟩ (last accessed 18.06.2013).

Ralanamahatana, C.A., Jessica Lin, D.G., Keogh, E., Vlachos, M., Das, G., 2005. Mining time series data. In: Maimon, O., Rokach, L. (Eds.), Data Mining and Knowledge Discovery Handbook. Springer, US, pp. 1069–1103.

Tenenbaum, J.B., Silva, V.d., Langford, J.C., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500), 2319–2323.

Wang, J.M., Fleet, D.J., Hertzmann, A., 2006. Gaussian process dynamical models. In: Weiss, Y., Schölkopf, B., Platt, J. (Eds.), Advances in Neural Information Processing Systems 18. MIT Press, pp. 1443–1450.

Wu, H., Salzberg, B., Zhang, D., 2004. Online event-driven subsequence matching over financial data streams. In: Proceedings of 2004 ACM SIGMOD International Conference on Management of Data, pp. 23–34.

Yahoo, 2013. Yahoo! Finance. ⟨http://finance.yahoo.com/⟩ (last accessed 18.06.2013).

Yin, J., Si, Y.-W., Gong, Z., 2011. Financial time series segmentation based on Turning Points. In: Proceedings of the 2011 International Conference on System Science and Engineering (ICSSE), pp. 394–399.

Zhang, Z., Jiang, J., Liu, X., Lau, R., Wang, H., Zhang, R., 2010a. A real time hybrid pattern matching scheme for stock time series. In: Proceedings of the Twenty-First Australasian Conference on Database Technologies, vol. 104, pp. 161–170.

Zhang, Z., Jiang, J., Liu, X., Lau, W.C., Wang, H., Wang, S., Song, X., Xu, D., 2007. Pattern recognition in stock data based on a new segmentation algorithm. In: Proceedings of the 2nd International Conference on Knowledge Science, Engineering and Management, pp. 520–525.

Zhang, Z., Siekmann, J., Zhang, Z., Jiang, J., Liu, X., Lau, W., Wang, H., Wang, S., Song, X., Xu, D., 2010b. Pattern Recognition in Stock Data Based on a New Segmentation Algorithm. Knowledge Science, Engineering and Management. Springer, Berlin, Heidelberg, pp. 520–525.