Copyright Undertaking

This work is protected by copyright, with all rights reserved.

**By reading and using the work, the reader understands and agrees to the following terms:**

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the work.

2. The reader will use the work for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.

3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

**IMPORTANT**

If you have reasons to believe that any materials in this work are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

Pao Yue-kong Library, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

http://www.lib.polyu.edu.hk
Perceptually Important Point Identification for Big Data Analytics
Performance Analysis and Applications

Hung Ying Kit

Master of Science in E-Commerce

The Hong Kong Polytechnic University

November 2017
Statement of Authorship

Except where reference is made in the text of this dissertation, this dissertation contains no material published elsewhere or extracted in whole or in part from a dissertation presented by me for another degree or diploma.

No other person’s work has been used without due acknowledgement in the main text of the dissertation.

This dissertation has not been submitted for the award of any other degree or diploma in any other tertiary institution.

________________________________________

Name: Hung Ying Kit

Dated: 03/11/2017
Acknowledgements

I am grateful to my supervisor, Dr. Chung, Fu-lai Korris, Associate Professor in the Computing Department of Hong Kong Polytechnic University offering me opportunities to take Dissertation to study related to data science and give me valuable advice and guidance.

I wish to express my sincere thanks to Dr. Fu Tak Chung, Senior Lecturer in Department of Information Technology, Hong Kong Institute of Vocational Education (Sha Tin), for authorizing me to use literature review of his papers and advise me a lot valuable suggestions and guidance on my work.

Last but not the least, I would like to thank my wife, Judy, for looking after my 2 sons along and supporting me spiritually throughout working on my dissertation.
List of Publications


Hung, Y.K., Fu, T.C. and Chung, F.L. Distributed Perceptually Important Point Identification for Time Series Data Mining. *20th International Conference on Data Mining (ICDM 2018)*, Accepted.

Abstract

The concept of the Perceptually Important Point (PIP) identification process is introduced in 2001 in the field of times data mining. This process is originally work for financial time series pattern matching and it is then found suitable for time series dimensionality reduction and representation. The PIP strength is that it is able to preserve the overall shape of the time series by identifying the salient points. With the rise of Big Data, time series data contributes a major proportion, especially on the data which generates by sensors in the Internet of Things (IoT) environment. According to the nature of PIP identification and the successful cases, it is worth to further explore the opportunity to apply PIP in time series “Big Data”. However, the performance of PIP identification is always considered as the limitation when dealing with “Big” time series data.

In this dissertation, performance of PIP identification, which always limits the usability of this process, is studied in detail. Improvement algorithms on both algorithmic level and distributed environment are proposed and evaluated. Significant improvement in terms of speed is obtained by these improvement algorithms. In addition, the applications of PIP are reviewed. Then, the usability of PIP for deep learning is evaluated.
# Table of Contents

Statement of Authorship ........................................................................................................... II
Acknowledgements .................................................................................................................. III
List of Publications .................................................................................................................. IV
Abstract..................................................................................................................................... V
Table of Contents ...................................................................................................................... VI
List of Figures .......................................................................................................................... VIII
List of Tables ............................................................................................................................ X
Chapter 1 Introduction .............................................................................................................. 1
Chapter 2 Literature Review .................................................................................................... 3
Chapter 3 Perceptually Important Point Identification ........................................................... 8
  3.1 PIP identification .................................................................................................................. 9
  3.2 Distance Measures in PIP Identification ............................................................................ 10
  3.3 Storing the PIP ................................................................................................................... 14
  3.4 Retrieving PIPs ................................................................................................................ 17
Chapter 4 Performance Analysis and Proposed Solutions .................................................... 20
  4.1 Complexity Analysis ......................................................................................................... 20
  4.2 The Proposed Improvement Algorithms ....................................................................... 21
  4.3 Experimental Results ...................................................................................................... 28
Chapter 5 Distributed PIP Identification ................................................................................ 32
  5.1 Distributed Distance Calculation Approach ................................................................. 32
  5.2 Distributed SB-Tree Building Approach ..................................................................... 34
  5.3 Experimental Results ..................................................................................................... 37
Chapter 6 Applications of PIP ............................................................................................... 44
  6.1 Dimensionality Reduction ............................................................................................. 44
  6.2 Similarity Measure and Pattern Matching ................................................................. 45
6.3 Segmentation ................................................................. 47
6.4 Pattern Discovery .......................................................... 48
6.5 Visualization ................................................................. 50
6.6 Rule Discovery and Prediction ........................................ 51
6.7 The Integration of PIP and Deep Learning ....................... 53
Chapter 7 Conclusions ....................................................... 73
References ........................................................................... 75
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A sample heartbeat pattern in ECG with salient identified</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Pseudo code for PIP identification process (Fu, T.C., 2008b)</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Data point importance identification with Euclidean distance (PIP-ED)</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>(Fu, T.C., 2008b)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Data point importance identification with perpendicular distance (PIP-PD)</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>(Fu, T.C., 2008b)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Data point importance identification with vertical distance (PIP-VD)</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>Identification of the 10 PIPs using PIP-VD</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>Pseudo code of building the SB-Tree recursively</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>SB-Tree building process</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Pseudo code of accessing SB-Tree</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>SB-Tree accessing process</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Detail pseudo code of the Caching method of the PIP identification process</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Conceptual design of the Splitting algorithm</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>Detail pseudo code of the Splitting method of the PIP identification process</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>An example of the splitting process</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>Processing time of the three PIP identification processes in financial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>domain (Stock time series)</td>
<td>29</td>
</tr>
<tr>
<td>16</td>
<td>Processing time of the three PIP identification processes in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>engineering domain (temperature detected by valve)</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>Processing time of the three PIP identification processes in medical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>domain (EEG time series)</td>
<td>30</td>
</tr>
<tr>
<td>18</td>
<td>Processing time of the three PIP identification processes in a long</td>
<td></td>
</tr>
<tr>
<td></td>
<td>time series (EEG time series)</td>
<td>31</td>
</tr>
<tr>
<td>19</td>
<td>Performance of the two improved methods (Caching and Splitting) in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>terms of the ratio of improvement comparing to the Naïve PIP</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>identification method (EEG time series)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Distributed Distance Calculation Approach (DDCA)</td>
<td>33</td>
</tr>
<tr>
<td>21</td>
<td>Architecture of the Distributed SB-Tree Building Approach (DSBA)</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Pseudo code of the DSBA in the Controller side</td>
<td>36</td>
</tr>
<tr>
<td>23</td>
<td>Pseudo code of the DSBA in the Server side</td>
<td>37</td>
</tr>
<tr>
<td>24</td>
<td>Experimental result on different thresholds MinLen setting for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DDCA</td>
<td>39</td>
</tr>
<tr>
<td>25</td>
<td>Experimental result on different thresholds SegmentLimit setting for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSBA</td>
<td>40</td>
</tr>
<tr>
<td>26</td>
<td>Experimental result on different thresholds of depth setting for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSBA</td>
<td>41</td>
</tr>
<tr>
<td>27</td>
<td>Comparison of the 3 PIP identification processes: Standalone, DDCA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and DSBA</td>
<td>42</td>
</tr>
<tr>
<td>28</td>
<td>Processing Time Comparison when using different numbers of servers</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>PIP integrated with Deep learning network architecture</td>
<td>55</td>
</tr>
</tbody>
</table>
Figure 30 PIP-VD Error example ......................................................... 56
Figure 31 A simple TensorFlow graphic design ............................................. 57
Figure 32 Binary image of Bird and Chicken dataset (From: http://timeseriesclassification.com) .................................................. 58
Figure 33 Bird Class Training set ................................................................. 60
Figure 34 Chicken Class Training Set .......................................................... 62
Figure 35 Error (i.e. PIP-VD) by representing the Bird and Chicken training dataset with different numbers of PIPs ........................................... 63
Figure 36 Accuracy of the classification result by using different numbers of PIPs and different numbers of hidden nodes ................................................. 63
Figure 37 Accuracy of the Bird and Chicken testing dataset classification using different numbers of hidden nodes and different input data ................. 64
Figure 38 Testing patterns that PIP and PAA recognize it correctly but not in original time series pattern (i.e. #4, #12 & #13) ........................................... 66
Figure 39 A testing pattern that only PIP recognizes it correctly (i.e. #5) .......... 67
Figure 40 Sample financial technical patterns generated based on the head-and-shoulder (H&S), double tops, triple tops, rounded top and spike top respectively ........................................................................................................ 68
Figure 41 Error (i.e. PIP-VD) by representing the synthetic financial training set with different numbers of PIPs ......................................................... 69
Figure 42 Accuracy of the synthetic financial testing dataset classification using different numbers of hidden nodes and different input data ................. 69
Figure 43 Speed for training the deep neural network by using the raw data, PIP and PAA representation ................................................................. 70
Figure 44 A synthetic rounded top pattern that PIP and PAA recognize it correctly but not in original time series pattern ............................................. 71
Figure 45 A synthetic triple tops pattern that raw data and PIP recognize it correctly but not in PAA representation ......................................................... 71
List of Tables

Table 1 Data point importance list using PIP-VD ........................................... 12
Table 2 Detail classification result of the Bird and Chicken testing dataset..... 65
Chapter 1 Introduction

Recent years, the rapid development of cloud computing, mobile technologies and Internet of Things (IoT) have led to the explosive growth of data. It is no doubt that Big Data Era is already here (Jin et al., 2015). Among the diversified data types (Variety), temporal data, in some time series data, is certainly one of the major contributions to this “Big” data (Yu et al., 2014). A time series is considered as a collection of observations made chronologically. The time series data has the features: large data size (Volume), high dimensionality (Variability) and update continuously (Velocity). Big data analytic can be an extension of data mining and machine learning research. Review on data mining for IoT can be found at (Chen et al., 2015), machine learning for big data at (Al-Jarrah et al., 2015) and time series data mining at (Fu, 2011).

Time series data generated continuously at almost every industry from geo, climate and environment (Rahman et al., 2016), bio, medicine and health (Xiao et al., 2017; Sakr and Elgammal, 2016), to business and finance (Zhou et al., 2015). Enormous time data is generating by IoT sensors and mobile devices all around the world in every moment (Ang and Seng, 2016; Papageorgiou et al., 2015b; Tortonesi et al., 2016). This creates a high dimensional space for further analysis. Moreover, time series data is considered as individual subject in numerical field. Therefore, representing the raw time series data in another form is always preferred before modeling and analysis.

Among different time series representation, Perceptually Important Point (PIP) is one of the approaches that is able to preserve the shape of the time series. In previous years, different applications of PIP are proposed, such as representing the movement shape of time series, using for data compression and modeling the time series data. In this research, we will conduct a detail study on the PIP identification process which include
In next Chapter, we first review different time series representation approaches. The conventional PIP identification process and the corresponding storage and retrieval approaches of the identified PIPs are reviewed in Chapter 3. In Chapter 4, we analyze on the performance and suggest improvement algorithms. Furthermore, we re-design the PIP identification process for the distributed environment in Chapter 5. Chapter 6 discusses on the common usages after representing time series by PIP and the existing applications. We also evaluate on the integration of PIP and deep learning in this Chapter. The final Chapter offers our conclusions.
Chapter 2 Literature Review

One of the major purposes of PIP is to reduce the dimension of a long data times series. The naive approach may be sampling (Astrom, 1969). That says a rate with \( n/m \), where \( m \) is the length of a time series \( P \) and \( n \) is the reduced dimension. However, the disadvantage of sampling method is distorting the shape of time series in case of too low sampling rate. Some information would be lost.

An improved method to reduce a long time series is by splitting it into \( n \) subset with same length and take the average (mean) value of each subset with a single value. That is a time series \( P = (p_1, \ldots, p_m) \) and \( \hat{P} = (\hat{p}_1, \ldots, \hat{p}_n) \) is the average value set that can be computed by:

\[
\hat{p}_k = \frac{1}{e_k - s_k + 1} \sum_{i=s_k}^{e_k} p_i
\]  

(1)

where \( s_k \) and \( e_k \) are the starting and ending data points of the \( k \)th segment in the time series \( P \), respectively. That is, a long time series is represented by its segmented average. (Yi and Faloutsos, 2000). This approach is also named Piecewise Aggregate Approximation (PAA) by Keogh et al. (2000) and Adaptive Piecewise Constant Approximation (APCA) (Keogh et al., 2001), study the Euclidean distance with lower bounding.

Another approach to dimension reduction is to use straight lines to represent a time series. The techniques are divided in two mean categories. The first category is linear interpolation. Piecewise Linear Representation (PLR) is a popular one (Keogh, 1997; Keogh and Smyth, 1997). The approximating line for the subset \( P(p_{i_1}, \ldots, p_{i_j}) \) is represented by the line connecting starting \( p_i \) and ending points \( p_j \). It tends to closely
align the endpoint of consecutive segments, giving the piecewise approximation with
connected lines.

Denoting symbolic form to represent a numeric time series is another time series
representation approach. The approach is to divide the time series into segments with
equal length. Each segment is then converted into a symbol (Yang and Zhao, 1998;
Yang et al., 1999; Motoyoshi, et al., 2002; Aref et al., 2004). Lin et al. (2003; 2007)
present a method called Symbolic Aggregate approxXimation (SAX) which convert
point in PAA to symbol string. It is to divide the distribution space (y-axis) into regions
with equal width. Assigning a symbol to a region, segment reside at it can be mapped
to a symbol. The transformed time series $\hat{P}$ is converted to a symbol string
$SS(s_1,\ldots,s_W)$.

The methods described before are directly to represent time series by time domain.
Another large family of approaches are to representing time series in the transformation
domain. One of the popular transformation techniques is Discrete Fourier Transforms
(DFT) (Agrawal et al., 1993). Research also uses wavelet transform (Struzik and
Siebes, 1998), Discrete Wavelet Transform (DWT) (Chan and Fu, 1999), Principal
Component Analysis (PCA) (Fukunaga, 1990) and Singular Value Decomposition
(SVD) (Korn et al., 1997).

The approaches mentioned above are that representing an original time series will
smooth out salient points which has counterproductive effect for some type of time
series data (Chung et al., 2001; Fu et al., 2008b; Park et al., 2010). Financial domain
is a typical example, identification of technical patterns relies on the data shape and
the importance of data points. For these purposes, it is very important to retain the
information of salient points as they are main construction of the shape of time series.
Chung et al. (2001) propose the Perceptually Important Point (PIP) identification. It is
used for matching technical patterns in financial domain. The similar concept is a technique reducing the number of point for representing a line by Douglas and Peucker (1973) (see also Hershberger and Snoeyink, 1992). A complete and comprehensive description of the algorithm can be found at (Fu et al., 2008b). PIP identification is considered as preprocessing step in financial application by Tsinaslanidis and Zapranis (2016).

Similar work has also been found in (Perng et al., 2000, Pratt & Fink, 2002 and Fink et al., 2003). Perng et al. (2000) use a landmark model for similarity measurement by identification of the important points. Pratt and Fink (2002) and Fink et al. (2003) denote minima and maxima as extrema in a time series and select certain important extrema and discard other points. The main concept is to drop minor fluctuations and retain major minima and maxima. It can be considered as compression process which define a compression ratio with a parameter $R$. $R$ is greater than one; an increase in $R$ leads to decrease points selected. That is, given a series which has starting point $i$ and ending point $j$ where $i \leq x \leq j$, a point $p_x$ is an important minimum (maximum) if $p_x$ is the minimum (maximum) in $p_i,...,p_j$, and $p_i/p_x \geq R$ and $p_j/p_x \geq R$. This algorithm found out the important points with their values and indices, as well as the first and last points. The computation time is linear, and memory used is constant. This algorithm can process new points at arriving but not need to store the original series. The identification of important points is worked on local information of each segment. It does not need to consider the points outside of each segment. Fink and Gandhi (2010) also proposed the idea to identify the importance levels of extrema. Identifying the local maximum and minimum points to be the turning points is proposed in (Yin et al., 2011; Si and Yin, 2013). All points identified are stored in an optimal binary search tree (OBST) ordered by their importance. Similarly, Vo. et al. (2013) also propose on dimensionality reduction by the turning points identification.
One of the major differences between the original PIP identification process and the approaches discussed so far is that they only consider the local information of a segment rather than the shape of the whole time series. It is because of the performance consideration especially when focusing on streaming data.

Similar to the PIP identification, Pheking et al. (2008) propose to identify two most important points in each iteration being the most peak (MP) point and the most dip (MD) point. A TS-binary search tree is used to store identified points. Similarly, Wu and Huang (2009) represent time series with the maximum points and the minimum points (which called Extreme Points) in different ranges and gets the important points in different resolutions.

Bao (2008) propose a Critical Point Model (CPM) and Bao and Yang, 2008 also propose a high-level representation built on a sequence of critical points for financial data analysis. Wang et al. (2013) term the PIP identified as feature points. Yu et al. (2017) propose the key turning points (KTPs) which are determined by PLR. These points are representing the changing points of correlation directions among process variables. Idea behind is also to identify point with the maximum distance to a line connecting the first and last points of the time series which has the greatest influence on the correlation direction.

Several research groups proposed different extensions/variants of PIP to support different usages. For example, Mojsilovic (2007) extends the PIP identification process to support multivariate time series data.

Phetking et al. (2009) and Phetchanchai et al. (2010) propose a method of important points identification build on zigzag trends behavior. It is called Zigzag based PIP (ZIP) identification method. They use the method to identify the bending of the financial time series. Furthermore, the identified PIPs are indexed based on a Zigzag
based Multiway Search Tree (ZM)-Tree.

Son and Anh (2010) propose an Improvement of PIP (IPIP) identification method and a multi-dimensional index structure based on the Skyline-Index (Li et al., 2004) which provides tighter lower bounds.

Based on ZIP, Chi and Jiang (2012) propose a Feature Zigzag based Perceptually Important Points (FZPIP) feature identification method by introducing the definition of futures trend characteristics. They present a Feature Index Structure (FIS) index mechanism based on the binary search tree and define the futures time series is composed of a series of price data which is observed from futures trading software in chronological order. In this method, VD is adopted to measure the importance and sign “+” and “-” are kept forming the Sign Vertical Distance (SVD). Then, FIS is proposed to store the PIPs.

The research of Charbonnier et al. (2013) studies the detection of quenches occurring on the LHC super-conducting magnets based on the trend analysis of the first derivative of pressure. Instead of calculating the first order derivative using calculus, a modified version of PIP by adding a post evaluation process of the PIPs is adopted to general the episodes.

Instead of using on time series data, Song and Lee (2013) adopt PIP to find major inflection points of atypia-amplitude signature. They further introduce a post processing step to eliminate unnecessary PIPs. In next chapter, the conventional PIP identification process will be reviewed.
Chapter 3 Perceptually Important Point Identification

A time series is sequence of data points. Each data point’s amplitude has different influence on the shape of the time series. In other words, each data point has own importance to the time series but not equal. Some data point may be considered as main contribution to the whole series, but some may be considerate without any influence. The less influence may be discarded. For example, a heartbeat pattern in an electrocardiography (ECG) can be identified by a few salient points which play an important position. As shown in Figure 1, the human can easily identify the highlighted important points of a heartbeat visually. Then these points are more important than others. In this Chapter, we review the conventional PIP identification approach (Fu et al., 2008b; Seunghye, 2017). After that, we review the three methods of evaluating PIPs in a time series. They are: Euclidean distance (PIP-ED), perpendicular distance (PIP-PD) and vertical distance (PIP-VD). At the end, we present the storage and retrieval methods of the PIP.

![Figure 1 A sample heartbeat pattern in ECG with salient identified](image-url)
3.1 PIP identification

PIP Identification is to figure out the influence of a data point to the shape of the time series. A data point having a greater influence to the time series shape is considered as more important. The identification process of PIPs is as follows: Given a time series $P(p_1, \cdots, p_m)$ will go through the PIP identification process as described in Figure 2. The first and last points of $P$ would be the first two PIPs. The next PIP that is found will be the point in $P$ with the maximum distance to the first two PIPs. The fourth PIP will be found that has greatest distance to its two adjacent PIPs, either between the first and second PIPs or between the second and the last PIPs. The identification process of the PIPs continues until all the points in $P$ are located or a required PIP number are identified.

```
1 Function PIP_Identification (P)
2     Input: Time Series P[1..m]
3     Output: PIP PIP[1..m]
4     Begin
6         Repeat until PIP[1..m] all filled
7             Begin
8                 MaxDist = -1
9                 For each i in P and not in L
10                    Begin
11                        Calculate the distance Dist of i to the 2 adjacent points in PIP
12                        If Dist > MaxDist
13                            Begin
14                                MaxDist = Dist
15                                j = i
16                            End
17                        End
18                        Append P[j] TO PIP
19                     End
20     Return PIP
21 End
```

Figure 2 Pseudo code for PIP identification process (Fu, T.C., 2008b)
3.2 Distance Measures in PIP Identification

Three evaluation methods are previously proposed to measure this distance to the two adjacent PIPs. They will be introduced in the next 3 sections.

3.2.1 Euclidean Distance

The first one is using the Euclidean Distance (ED) (Fu, T.C., 2008b) to represent the importance of points lay between 2 adjacent points. As illustrated in Figure 3, this measurement computes the sum of the ED of the test point \( p_3 = (x_3, y_3) \) to its adjacent PIPs \( p_1 = (x_1, y_1) \) and \( p_2 = (x_2, y_2) \), i.e.,

\[
ED(p_1, p_2, p_3) = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} + \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}
\] (2)

This important point will be most found at the middle part between \( p_1 \) and \( p_2 \).

![Figure 3 Data point importance identification with Euclidean distance (PIP-ED) (Fu, T.C., 2008b)](image)

3.2.2 Perpendicular Distance

The second identification method is based on Perpendicular Distance (PD) (Fu, T.C., 2008b). This measurement is the PD between the line connecting 2 adjacent PIPs and test point \( p_3 \) as shown in Figure 4, i.e.,

\[
Slope(P_1, P_2) = S = \frac{y_2 - y_1}{x_2 - x_1}
\] (3)
\[ x_c = \frac{x_2 + (s \cdot y_3) - (s \cdot y_2) + (s^2 \cdot x_2)}{1 + s^2} \]  
(4)

\[ y_c = (s \cdot x_c) - (s \cdot x_2) + y_2 \]  
(5)

\[ PD(P_3, P_c) = d = \sqrt{(y_c - y_3)^2 + (x_c - x_3)^2} \]  
(6)

![Figure 4 Data point importance identification with perpendicular distance (PIP-PD) (Fu, T.C., 2008b)](image)

### 3.2.3 Vertical Distance

The last data point identification method is by Vertical Distance (VD) (Fu, T.C., 2008b) as shown in **Figure 5**, computes VD between the line connecting 2 adjacent PIPs (p1 and p2) and test point p3, i.e.,

\[ VD(p_3, p_c) = \left| y_3 - y_c \right| = \left| y_1 + (y_2 - y_1) \cdot \frac{x_c - x_1}{x_2 - x_1} - y_3 \right| \]  
(7)

where \( x_c = x_3 \). The points in PIPs are ordered by their fluctuation ratio. The most fluctuated sequence and highly fluctuated points are taken out as most front of PIPs.
Figure 5 Data point importance identification with vertical distance (PIP-VD)  
(Fu, T.C., 2008b)

An example of the PIP identification process is shown in Figure 6. 10 PIPs in the time series is identified based on PIP-VD. If a PIP is identified earlier, it will be more important. The point with smaller number is more important (e.g. PIP 3 is more important than PIP 4).

Based on the PIP identification process, Fu et al. (2008b) further process the concept of data point importance and the corresponding data point importance list. Table 1 shows the data point importance lists built using different PIP-VD.

<table>
<thead>
<tr>
<th>Importance</th>
<th>x</th>
<th>Y</th>
<th>Distance Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.2</td>
<td>n/a</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.9</td>
<td>n/a</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0.1</td>
<td>0.49</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.7</td>
<td>0.56</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0.1</td>
<td>0.43</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0.6</td>
<td>0.30</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>0.4</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0.2</td>
<td>0.20</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>0.6</td>
<td>0.20</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.7</td>
<td>0.10</td>
</tr>
</tbody>
</table>
Figure 6 Identification of the 10 PIPs using PIP-VD
3.3 Storing the PIP

To better store the identified PIP based on their corresponding data point importance, a tree structure called Specialized Binary (SB) Tree proposed by Fu et al. (2004a). It is built on binary tree structure. Searching started from the most important data point is fast. When new time data points are coming continuously and frequently, it is necessary to build/rebuild a SB-Tree that needs a lot of time, Fu et al. (2005d) propose three updating methods. They are periodic rebuild, batch update and point-by-point update mechanisms.

Intuitively, a SB-Tree arrange the data points in a hierarchy structure. The following descript SB-Tree structure in details:

- A node has the x- and y- coordinates of PIP which is identified in PIP identification process.
- The path between nodes stores the distance measured between parent and child note

The PIP identification process is applied on the whole time series to create a SB-Tree. The first and last data points in the original time series are taken as the first two nodes of the SB-Tree. The last point is the root of the tree while first point is the child on the left-hand-side of the root. The third PIP identified would be the child on the right-hand-side of the node. We customize the tree building approach with recursive process as shown in Figure 7. As a simple example, Figure 8 shows the steps of creating a SB-Tree, based on the sample time series given in Figure 6. In Figure 8, the number of the node is the cnode.x value of the data point while the number of the path before each node shows the distance measured, cnode.dist.

```
1  Function Main (p)
2    Input: TimeSeries p[1..m]
3    Output: SBTree root
```
Begin
// Last data point
Create root
root.x = m
root.y = p[m]
root.left = NULL
root.right = NULL
root.dist = NULL
// First data point
Create node
node.x = 1
node.y = p[1]
node.dist = NULL
node.left = NULL
node.right = NULL
root.left = node
node.right = Build_SB_Tree(p)
Return root
End

Function Build_SB_Tree (q)
Input: Segment q[starts..ends]
Output: Node_Pointer node_ptr
Begin
Calculate the distance of each data point in
q[starts+1..ends-1] with line draws from starts to ends
Select point j in q with maximum distance dist calculated in
step 31
Create new_node
new_node.x = j
new_node.y = q[j]
new_node.dist = dist
new_node.left = NULL
new_node.right = NULL
node_ptr = new_node
If (starts+1 < j) Then
new_node.left = Build_SB_Tree(q[starts..j])
End If
If (j < ends-1) Then
new_node.right = Build_SB_Tree(q[j..ends])
End If
Return node
End

Figure 7 Pseudo code of building the SB-Tree recursively
Figure 8. SB-Tree building process
3.4 Retrieving PIPs

In this section, we discuss the retrieving of PIPs after a SB-Tree is built. PIP are retrieving starting from the root one by one sequentially according to the importance recursively.

- The root represents first PIP. A sorted heap stores the child for the next retrieval process.
- The tree representation is accessed from the root and each accessible node in each path of the tree is checked. An accessible node is defined as the first node of a path that is not retrieved yet. All these nodes will be put into a heap.
- By sorting the distances among all the accessible nodes in the heap, the first one (the one with maximum distance, VD) is selected as the next PIP. Again, this node will be removed from the heap.
- This process continues until all the nodes in the tree are processed and retrieved.

*Figure 9* shows the pseudo code for accessing an SB-Tree. As an example, the sample time series is retrieved from the SB-Tree created in section 3.1. Starting from the root of the SB-Tree in Figure 10, the third PIP (the starting and ending points of the time series are the first two PIPs) is the third node (i.e. point 5, *Figure 10a*). This node is marked as USED (in black in the diagram). To identify the next PIP, all the accessible nodes in the tree are first identified (bold circles). Then, the node with greatest distance stored is identified as the next PIP, for example, point 9 in *Figure 10b*. This process continues until all the nodes in the tree are marked and the whole time series is retrieved according to the order of the data point importance in Table 1.
<table>
<thead>
<tr>
<th>Function Access_Tree(root)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Tree root</td>
</tr>
<tr>
<td><strong>Output:</strong> PIPList L</td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>If root.left &lt;&gt; NULL Then</td>
</tr>
<tr>
<td>Put root.left in Heap</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
<tr>
<td>Repeat until Heap is Empty</td>
</tr>
<tr>
<td>// L[1..m] all filled</td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>Remove a node from Heap</td>
</tr>
<tr>
<td>where node.dist is Maximum</td>
</tr>
<tr>
<td>Append node.x TO L</td>
</tr>
<tr>
<td>/* Other information like node.y can also be appended to L */</td>
</tr>
<tr>
<td>If node.left &lt;&gt; NULL Then</td>
</tr>
<tr>
<td>Put node.left in Heap</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
<tr>
<td>If node.right &lt;&gt; NULL Then</td>
</tr>
<tr>
<td>Put node.right in Heap</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
<tr>
<td><strong>End</strong> Return L</td>
</tr>
</tbody>
</table>

**Figure 9 Pseudo code of accessing SB-Tree**

**Figure 10 SB-Tree accessing process**
Comparing with traditional approach, the suggested approach has advantage that sequentially retrieves the time series from the begin to the end even though a few PIP taken, the overall shape can be preserved. In addition, Fu et al. (2006a) further describe how to search for a segment of time series (i.e. subsequence) in the tree and Fu et al. (2006d) design a query language for the SB-Tree. On the other hand, by putting a set of SB-Trees together, a time series data store is formed and the concept of Specialized Binary Tree-Forest (i.e. SBT-Forest) is proposed by Fu et al. (2005c). This is considered as an indexing approach based on transforming the SB-Trees to symbol strings first and then indexing the symbol strings by a trie data structure. After providing the background algorithm, we will study the performance of the PIP identification process and offer our proposed improvement algorithms in the next chapters.
Chapter 4 Performance Analysis and Proposed Solutions

Preliminary evaluation of the original work of PIP identification process can be found at (Chung et al., 2001) and a series of evaluation can be viewed at (Fu et al., 2008b). It evaluated the error by representing time series with various numbers of PIP and different distance measures, the visualization effect, the ability of different updating approaches by using the SB-Tree, and its accuracy and processing time. Wan et al. (2016) further evaluate the performance of different time series representation methods in terms of accuracy, precision and recall. They conclude that PIP identification method can perform well to preserve the overall shape of the sequence. Gong et al. (2016) also compare different pattern matching methods. However, there is lack of detail investigation on the performance of the PIP identification process on Big time series data in the literature. Therefore, we investigate the performance of the PIP identification process in detail in this Chapter first. Then, possible improvement algorithms will be proposed and evaluated.

4.1 Complexity Analysis

One of the major considerations when adopting PIP identification process is its complexity. In the study of Park et al. (2010), PIP takes a longer run time than SAX because PIP identification process requires more scans of time series than SAX. Todorov et al. (2015) specify that one of the bottlenecks in their study is the PIP identification process when analyzing a large amount of data. Son and Anh (2011a) also comment that their proposed variation of PIP, i.e. IPIP, suffers the computational complexity which is higher than that of PAA. Phetking and Selamat (2008) suggest that reducing the computational expensiveness of PIP identification process can support the user to estimate his predictive performance of his proposed approach.
The purpose of the PIP identification process is to identify the point that contributes most to the overall shape of a time series in each iteration. Therefore, the original algorithm needs to evaluate every point in a time series for every PIP identification and the worst case complexity is $O(n^2)$ (Todorov et al., 2015; Jugel et al., 2014; 2016). However, such worst case will be happened in very rare shapes of the time series like nearly a straight line or regular zip-zap patterns. The estimated average case complexity of PIP identification is $O(n \log n)$.

To minimize the re-calculation issue when dealing with time series streaming data, a SB-Tree structure is proposed with the support of different updating approaches as mentioned in Section 2.3. In the case of Zhou and Hu (2009), it is focusing on the dynamic recognition system. They propose a dynamic algorithm based on PIP identification. Similar to the SB-Tree, a binary tree structure is first used to organize the identified PIP. Then, a dynamic PIP identification algorithm is proposed for updating the binary tree. The proposed algorithm avoids computation expense and reduces the computation cost to 20% of the original PIP. In another case from Si and Yin (2013), as a trade-off for the online representation and real-time applications, instead of focusing on the overall shape, their proposed method is based on local comparison for preserving as many of trends in a time series as possible. Retaining more monotonous trends is the major advantage due to its identification ability of turning points which represent trend fluctuations in the time series.

### 4.2 The Proposed Improvement Algorithms

As shown in Figure 2, to identify the importance of all the data points in a time series with length $m$, it is necessary to conduct $m$ iterations. In each iteration, besides those points already identified as PIPs, the distances between all the remaining data points and the shape formed by the PIPs already identified in the previous iterations need to
be calculated.

4.2.1 Caching Algorithm

Papageorgiou et al. (2015b) focus on the streaming of IoT data. As the nature of the original PIP identification process is based on time series data already collected and it is claimed as resource intensive to perform the process for each incoming point, they propose to handle the incoming data by maintaining the size of the cache at a level that does not cause significant delays in the pre-processing and forwarding of items. They called this step as cache reduction and it is performed to reduce the point number analyzed in each step in a way that, it does not harm the quality of the analysis but makes the delay insignificant compared to the transmission delay.

Here, we also propose to cache the previous work done in the PIP identification process (Figure 11). We call it as Caching algorithm. In each iteration, as the PIP identified in the previous iteration only changes the shape of a segment instead of the whole time series, the distances computed in the previous iteration for many data points, which are not located in this segment, are remain unchanged. By caching these results, it is expected the speed of the whole process can be speed up.

The algorithm works on the following steps:

- Set an array (CacheResult) to store PIP value laid between CachPreDonePoint and CachNextDonePoint.
- Take CachPreDonePoint and CachNextDonePoint which in between has CacheResult with MaxDist,
- All points laid between CachPreDonePoint and CachNextDonePoint are set -1 to CacheResult.
- In the next step, it only need to re-calculate the CacheResult with -1.
Function PIP_Identification_Caching (P)

Input: Time Series P[1..m]
Output: PIP I[1..m]

Begin
Set I[1] = P[1], I[2] = P[m]
Float CachePreDonePoint, CacheNextDonePoint
Float CacheResult[1..m]
Set all CacheResult[1..m] = -1
Repeat until I[1..m] filled
Begin
MaxDist = -1
For each i in P and not in L
Begin
If CacheResult[i] = -1 Then
Begin
Calculate the distance Dist of i to the adjacent points in L
CacheResult[i] = Dist
End
If Dist > MaxDist Then
Begin
MaxDist = Dist
j = i
CachePreDonePoint = smaller point of the adjacent points in L
CacheNextDonePoint = larger point of the adjacent points in L
End
End
Append P[j] TO L
Set all points CacheResult[i] = -1 where i lay between CachePreDonePoint and CacheNextDonePoint
End
Return L
End

Figure 11. Detail pseudo code of the Caching method of the PIP identification process

4.2.2 Splitting Algorithm

We further purpose another algorithm to speed up the performance which does not cache the distance but caches the maximum distance and maximum point between two adjacent points in L. We called it as Splitting Algorithm. Figure 12 illustrates the general idea of the conceptual design.
First, a sorted segment list is maintained. Each segment in the sorted list has BEGIN POINT, END POINT, MAX POINT and MAX VALUE (i.e. Max Perpendicular Distance). BEGIN POINT and END POINT have already been in the PIP list but the points lay between BEGIN and END Points are not in the PIP list. MAX POINT is the point laid between BEGIN and END with MAX VALUE. The segments are sorted with MAX VALUE in descending order.

The algorithm works on the following steps for assigning a point to the PIP list:

- Remove the HEAD segment from the sorted list.
● Put the MAX POINT and MAX VALUE of the HEAD segment to PIP list.
● Split the HEAD segment into 2 segments.
● Set one segment’s BEGIN POINT to be BEGIN POINT of HEAD node and END POINT to be the MAX POINT.
● Set the other segment’s BEGIN POINT to be the MAX POINT of HEAD node and END POINT to be the END POINT of HEAD NODE.
● Calculate the MAX VALUE and MAX POINT in the 2 nodes corresponding to their BEGIN and END POINTS.
● Insert the 2 split segments into the sorted list at the right position provided that the sorted track list is in descending order.
● Repeat the above steps until all the points are filled in PIP list.

This algorithm does not repeat any calculation of the distance of each point except changing its adjacent point in PIP. It does not need to consider all MAX POINT in each step but need to insert the 2 segments into the sorted list. It is expected the PIP identification process can speed up a lot by reducing most of the repetitive calculation. **Figure 13** shows the detail procedure. **Figure 14** demonstrates how the splitting algorithm works by an example.
Function PIP_Identification_Splitting (P)
Input: time series P[1..m]
Output: PIP I[1..m]
Begin
Set I[1] = P[1], I[2] = P[m]
DataStruct TrackList
Begin
Int Begin
Int End
Float Max
Int MaxPoint
Float MaxValue
TrackList * PrePointer
TrackList * NextPointer
End
TrackList * ListHead // the head of the sorted list
TrackList * MaxSegment
Listhead->Begin = 1
Listhead->End = m
Calculate the MaxValue and MaxPoint in ListHead with ListHead-
>Begin and ListHead->End as the adjacent points in PIP
Repeat until I[1..m] all filled
Begin
MaxNode = ListHead
ListHead = Listhead->NextPointer
Append MaxNode->MaxPoint TO I
Split MaxNode into 2 segments HeadSegment and TailSegment
Begin
HeadSegment->Begin = MaxSegment->Begin
HeadSegment->End = MaxSegment->MaxPoint
TailSegment->Begin = MaxSegment->MaxPoint
TailSegment->End = MaxSegment->End
End
Calculate the MaxValue and MaxPoint in HeadSegment with
HeadSegment->Begin and HeadSegment->End as the adjacent points in PIP
Calculate MaxValue and MaxPoint in TailSegment with
TailNode->Begin and TailNode->End as the adjacent points in PIP
Insert HeadSegment and TailSegment into sorted list
beginning with ListHead
End
Return I
End

Figure 13 Detail pseudo code of the Splitting method of the PIP identification process
Figure 14 An example of the splitting process
4.3 Experimental Results

We evaluate the two proposed improvement algorithms in this section on different dimensions, which include their ability on different datasets and the effect when increasing the length of time series. The experiments are implemented with Visual C++ programming language performed on a standalone PC with configuration: Microsoft Windows 10 Professional edition, Intel ® Core™ i5-4440CPU @3.10GHZ, 8 GB RAM.

4.3.1 Experiments on Different Application Domains

To evaluate the performance of the two improvement algorithms in different application domains, 720 experiments were conducted. The dataset for the simulation tests include three application domains which are financial (stock data), engineering (temperature detected by a Valve) and medical (Electroencephalography, EEG) domains. Each domain has 10 time series and each time series has 4,000 data points. The average processing time among the 10 time series were presented.

*Figure 15 to Figure 17* show the time performance on the corresponding three application domains. The results are very close to each other. It helps to improve the conviction that the PIP identification process works well for the time series from different application domains.

The Naïve method performs the worst and it is served as the base for the comparisons of the two improvement methods. The Caching method kept running around 25% of Naïve and the Splitting method kept 3~4% of Naïve method.
Figure 15 Processing time of the three PIP identification processes in financial domain (Stock time series)

Figure 16 Processing time of the three PIP identification processes in engineering domain (temperature detected by valve)
4.4.2 Experiment on Big Time Series Data

To figure out whether the improved performance can be sustained when the time series data is increasing, an EEG time series with 200,000 data points was adopted in this test. Figure 18 shows the processing time of the three PIP identification processes and Figure 19 shows the performance of the two improved methods (Caching and Splitting) in terms of the ratio of improvement comparing to the Naïve PIP identification method. The Caching method can keep speed ratio 25% of Naïve method even the length of time series increased. On the other hand, the improvement of the Splitting method drops gradually when the data points increased. It is because the Splitting method needs to maintain the sorted list and every step needs to insert the two new nodes to the sorted list. If the sorted list is very long, the time to process will be increased a lot. However, the Splitting method is still the fastest methods among the three.
Figure 18 Processing time of the three PIP identification processes in a long time series (EEG time series)

Figure 19 Performance of the two improved methods (Caching and Splitting) in terms of the ratio of improvement comparing to the Naïve PIP identification method (EEG time series)
Chapter 5 Distributed PIP Identification

To further improve the performance of the PIP identification process, re-design the algorithm for the distributed environment is a possible direction. One of the popular distributed programming frameworks is dispy for Python. It is an independent framework for distributed and network programming which can support asynchronous and concurrent execution. It uses compute clusters to execute computations in parallel across multiple processors among many computers in a cluster, grid or cloud.

Adopting parallel computation in the field time series data mining is not the main stream of research study. The current work that can be found in the literature includes: based on hardware architecture consideration, Movchan and Zymbler (2015) and Narang and Bhattacherjee (2010) propose customized parallel algorithms for time series subsequence search and motif discovery respectively. On the other hand, Kholod et al. (2017) focus on the data and proposes to perform distributed analysis on the time series data by only focusing on a subset of data in most of the analysis tasks to reduce the network traffic.

In this Chapter, two distributed approaches of the PIP identification process are proposed which are based on the concept of the Specialized Binary (SB) Tree. The two proposed distributed approaches are introduced in Section 5.1 and 5.2. The experimental results are evaluated in Section 5.3.

5.1 Distributed Distance Calculation Approach

The first proposed distributed approach is simply that distributes the distance calculation of data points in a segment to a set of servers. That is, instead of calculating the distance of each data point one-by-one in a standalone computer, the data points
which need to compute the distance in Step 31 of *Figure 7* can be distributed to a set of servers. In another view, each server is assigned with equal number of data points to be computed. We called this approach as Distributed Distance Calculation Approach (DDCA).

This proposed DDCA is a synchronous parallel process. Step 31 of *Figure 7* will be waited until all the involved servers completed the calculation and returned their results to the main process. *Figure 20* shows the distributed part of DDCA. It may suffer from the problem that if any server returns result lately no matter due to the network problem or lower computation capability, the whole process needs to wait until all servers finished their computation.

![Figure 20 Distributed Distance Calculation Approach (DDCA)](image-url)
5.2 Distributed SB-Tree Building Approach

As shown in Section 3.3, the building of each branch in the SB-Tree is an independent process and the sub-tree under each branch can be built in parallel. Furthermore, building of the sub-tree in each level is a recursive process. Based on these natures, we further propose to transform the SB-Tree building process to a distribution process. We called it as Distributed SB-Tree Building Approach (DSBA).

In DSBA, each segment being processed will be distributed to a server to build a sub-tree recursively. A Controller is used to monitor the whole process and all Servers are responsible for building the sub-tree of the assigned segment. The algorithm is as below:

1. Controller keeps a Segment Queue and a Job List
2. Controller has Distribution Function and Call Back Function which can work concurrently
3. Distribution Function pops first Segment from the Segment Queue and distribute it to a free Server to build sub-tree. It adds a Job ID together with the segment to the Job List. If no Segment in Segment Queue, it waits
4. Assigned Server receives the Segment and calculate the PIP in this Segment and put each newly formed segment in a list
5. Server calls the Call Back Function at the Controller and return the list of newly formed segments with Job ID
6. Call Back Function scans the Job ID in the Job List and remove it from Job List
7. Call Back Function rebuilds the subtree and adds it to the SB-Tree and appends all the newly returned leave segments to the Segment Queue
8. Repeat Step 3 to 7 until both Segment Queue and Job List are empty

It is necessary to rebuild the sub-tree when server return result. The parent node in
binary structure points to child’s address and this address is the server address. This address is not existed in controller. The server only return a list a segment so the controller need to rebuild it according to the max point.

The Servers work asynchronously and they do not need to wait for each other to finish the computation. Distribution Function and Call Back Function in the Controller work independently except that Distribution Function needs to wait when the Segment Queue is empty. DSBA optimally use the Server time even different servers have different computation time. **Figure 21** shows the overall architecture of the system based on DSBA. **Figure 22** and **Figure 23** are the detail algorithms in the Controller and Server sides respectively.

![Figure 21 Architecture of the Distributed SB-Tree Building Approach (DSBA)](image)

```
1 Function Controller (p)
2   Input: Time series p[1..n]
3   Output: SBTree root
4   Begin
5     Create Global Segment_Queue = Empty
6     Create Global Job List = Empty
```
// Last data point
Create root
root.x = n
root.y = p[n]
root.left = NULL
root.right = NULL
root.dist = NULL

// First data point
Create node
node.x = 1
node.y = p[1]
node.dist = NULL
node.left = NULL
node.right = NULL
root.left = node

PUSH p to Segment_Queue

Repeat
   Call Distribution ()
Until Segment_Queue = Empty and
   Job_list = Empty
Return root

End

Function Distribution ()
Begin
   Wait Until
   Identify available Server S
   POP Segment seg from Segment_Queue
   Generate Job_ID JID
   ADD (JID, seg) to Job_List
   Call Asyn.Build_SB_Tree(JID, seg) at server S
End

Function Call_Back (JID, node_ptr, seg_list)
Input: Job_ID JID
Input: Node Pointer node_ptr
Input: Segment_List seg_list
Begin
   Remove the JID from Job_List and get seg segment
   Rebuild seg_list to a subtree
   Put subtree to the tree under seg
   For Each Leave in the subtree
      Begin
         PUSH Leave to Segment_Queue
      End For
End

Figure 22 Pseudo code of the DSBA in the Controller side
1 Function Build_SB_Tree (JID, q)
2   Input: Job_ID JID
3   Input: Segment q[starts..ends]
4
5 Begin
6   Calculate the distance of each data point in q[starts+1..ends-1] with line draws from starts to ends
7   Select point j in q with maximum distance dist calculated in step 6
8
9   Create new_node
10  new_node.x = j
11  new_node.y = q[j]
12  new_node.dist = dist
13  new_node.left = NULL
14  new_node.right = NULL
15
16  Create Segment_List seq_list = Empty
17  Add q[starts..j] to seq_list
18  Add q[j..ends] to seq_list
19
20  Call Controller.CallBack (JID, new_node, seg_list)
21 End

Figure 23 Pseudo code of the DSBA in the Server side

5.3 Experimental Results

In this section, we evaluate the two proposed distributed approaches: Distributed Distance Calculation Approach (DDCA) and Distributed SB-Tree Building Approach (DSBA) for SB-Tree building (i.e. PIP identification). The parameters of the two approaches are first proposed and evaluated. Then, the performance of them based on the optimized parameters is evaluated.

The algorithms were implemented with dispy build on Python. The main program (i.e. Controller) was performed on a PC with configuration: Microsoft Windows 10 Professional edition, Intel ® Core™ i5-4440CPU @3.10GHZ, 20 GB RAM. To simulate the distributed environment of the Servers, 6 Virtual Machines were run on VMware Workstation which the physical environment is Microsoft Windows 10 Professional edition, Intel ® Core™, i7-7700 CPU @3.60GHz, 32GB RAM (config of the VMs). Each Virtual Machine is configured as 1 CPU with 4 GB RAM.
The dataset for the experiments include two long time series from different application domains, one from engineering domain (temperature detected by a Valve) and another one from medical (Electroencephalography, EEG) domain. Each time series has 100,000 data points. To minimize the unstable effect from the environment, we run 5 times for each experiment and the best result is reported.

5.3.1 Parameters Setup

Based on the initial testing, it is found that it is necessary to further introduce parameters for both approaches for better performance. For the DDCA, it shows that it is not worth to distribute the distance calculation of the data points to other servers once the length of the segment is too short. It is because the time for scheduling the task will be longer than the distance calculation process. In such case, a threshold $MinLen$ is proposed to stop the distribution process based on the length of the segments. The segment will be calculated locally once the segment length is smaller than $MinLen$. We tested $MinLen$ from 3,000 to 40,000. As shown in Figure 24, the results of $MinLen$ with 20,000 to 40,000 are very close and the segment length of 30,000 got the best results in most situations. Therefore, this setup will be adopted onwards.
For DSBA, the approach proposed in Section III only identifies one PIP for each \texttt{Build_SB_Tree} process in a Server and will call the \texttt{Call_Back} function immediately afterwards. Similar to DDCA, it is not worth to distribute a short segment to servers to calculate which will increase the number of communication processes between the Controller and the Servers. Therefore, we proposed a threshold called \textit{SegmentLimit} to control if the length of a segment is less than \textit{SegmentLimit}, then the building of the subtree will be calculated in local controller.

Different settings of \textit{SegmentLimit} was evaluated and it is found that the setting of 2,000 got the best result and it will be adopted in the following experiments (Figure 25).
The setting of the depth of a sub-tree to be considered for processing in a server is also a determine factor on effecting the processing speed. If less shallow depth, controller needs more time on scheduling process and communication with servers. Deeper depth can have good utilization of servers and less scheduling time, but time on rebuilding the tree is increased exponentially with depth increased. The time complexity of rebuilding a sub-tree is $O(d \times 2^d)$ where $d$ is the depth of the sub-tree. Giving $d$ as the depth, the time complexity going through the tree is $(d-1)/2$ and the number of sub-tree is $n = \sum_{i=0}^{d-1} 2^i$. The time complexity for rebuilding the sub-tree is $O(d \times n) = O(d \times 2^d)$. Figure 26 shows the experimental result on different depth numbers to be considered in a server. Depth being 5 has best result compared with Depth being 1, 9, 13. It proved that Depth can’t be too small or large.
Figure 26 Experimental result on different thresholds of depth setting for DSBA
5.3.2 Performance of the Proposed Distributed Approaches

In this part, we compare the performance of the two distributed approaches, DDCA and DSBA. The performance of the standalone approach is also reported to serve as the baseline for comparison.

As shown in Figure 27, the processing time in the length range from 10,000 to 40,000 points is almost the same as they are almost computed in a standalone mode. After the length of 40,000, performance starts to improve. DSBA always performs the best and when the length reaches 100,000 points, DSBA reduced 25% of the processing time.

![Figure 27 Comparison of the 3 PIP identification processes: Standalone, DDCA and DSBA](image)

The effect of using different number of servers is also tested. Figure 28 shows the processing time when processing time series with 600,000 data points using different numbers of server. It has a significant speed improvement when the number of server is...
increased from 2 to 3. When considering the data size in this test, further increase the number of server cannot get much benefit especially more coordination work is needed between the controller and the servers.

![Figure 28 Processing Time Comparison when using different numbers of servers](image)

Figure 28 Processing Time Comparison when using different numbers of servers
Chapter 6 Applications of PIP

In this chapter, the applications of PIP identification process on different time series analysis tasks are first reviewed. The applications include dimensionality reduction, similarity measure and pattern matching, segmentation, pattern discovery, time series data visualization and also rule discovery and prediction. Then, we study the integration of PIP with deep learning.

6.1 Dimensionality Reduction

By identifying the PIPs, reduction of the dimension reduction of time series is possible while retaining its shape and salient points. It can be considered as a compression mechanism and the compression ratio can be calculated by:

$$CR = \frac{\text{Data point number of the original time series}}{\text{PIP number}}$$

Jiang et al. (2007b) improve the results of Principle Component Analysis (PCA) by using PIP as the dimensionality reduction method to preprocess the time series. The detail mechanism for dimensionality reduction of the proposed SB-Tree representation by tree pruning can be found at Fu et al. (2008a). Feller et al. (2011) and Todorov et al. (2015) employ PIP to serve as the multidimensional compression technique. A streamification version of PIP identification method by using cache projection/reduction is proposed by Papageorgiou et al. (2015a) to serve as network-edge data reduction for IoT systems. In addition, Wan and Si (2017a; 2017b) adopt PIP identification to decrease the numbers of points in a time series their proposed template-based and rule-based pattern matching.

Song et al. (2015) work on the linkage between fine particulate matter and health. They generalize the PM$_{2.5}$ concentration series in each Pollution Episode using PIP, and
classify all the PM$_{2.5}$ Pollution Episodes (PPEs) into one of categories according to its evolution modes. Jimenez et al. (2016) focus on the users’ mobility patterns study of bike sharing system (BSS). They use PIP identified to represent and index different time series from three ratios of each bike station. Then, rule set is used to classify the data. Katircioglu-Ozturk et al. (2017) adopt PIP together with Discrete Cosine Transform (DCT) to serve as the core part of the proposed window-based time series feature extraction method (WTC). DCT is used as a heuristic to feed a cut-off percentage to select the PIPs from the class representative average time series.

### 6.2 Similarity Measure and Pattern Matching

Similarity measure is a basic analytic task for time series data. In traditional data analysis, similarity measure is based on exact matching of 2 time series. However, owing to the numerical and continuous nature, typically similarity measure is done approximately. Also, the matching process can be categorized to whole time series pattern matching and subsequence matching, which the later one means to identify pattern of a segment in a long time series.

The original design of PIP is also focused on pattern matching (Chung et al., 2001) especially when the salient points of the patterns are the important criteria for matching process like many applications in financial domain. It is suitable for stock data analysis. Jiang et al. (2007) add the maximum distance $D$ as another coefficient factor to enhance it to be more appropriately for stock pattern matching. PIP is adopted by Chen and Chen (2016) to identify bull-flag pattern in the stock time series data.

The original design of PIP pattern matching is based on Euclidean distance to measure the similarity between the PIP identified and the pattern template. This is considered as the amplitude distance (AD) for the template-based matching approach in (Fu et al., 2007a). Template distance (TD) is also introduced in this paper to control the distortion
of a pattern. These two distance measures are suggested to be weighted at half/half in the pattern matching task.

Based on the stock time series data nature, Zhang et al. (2007a) consider PIP as visually important and further introduced the concept of practically important points which the points that closer to the transaction time are more important. They called this pattern matching scheme as Visually and Practically Important Point (VPIP).

Based on IPIP, Son and Anh (2011b) combine the PIP identification method and clipping technique for subsequence matching. The proposed approach provides a lower bounding condition for time series dimensionality reduction and a bit level representation for time series that allows the user to choose compression ratio.

Besides the template-based pattern matching approach, Fu et al. (2007a) also propose to identify the query pattern by defining rules among the salient points of a pattern and the PIPs identified from the time series are evaluated according to these rules. This is called the rule-based approach which provides further ability to describe the query patterns and is constrainable on the shape of the query patterns. Salekin et al. (2012b) combine the template-based and rule-based approaches for composite pattern matching on variable subsequence lengths according to the PIP identified. They also propose to combine neural network with the rule-based approach for the same context (Salekin et al., 2012a).

Zaib et al. (2004) develop a system which adopts PIP identification for pattern recognition to emulate the human visual cognition process. A Pattern Definition Language (PDL) is then proposed to define patterns in time series by using a declarative programming paradigm.

(Berndt and Clifford, 1994) propose Dynamic Time Warping (DTW) which is one of
the most popular and field-tested similarity measures. By DTW, the proposed method in predefines some patterns to serve as templates for pattern detection.

### 6.3 Segmentation

Time series segmentation can be used as a preprocessing process for various time series analysis, a trend analysis technique, discretization of a long time series. The character of time series data their numerical and continuous nature but the transactional databases is in discrete manner. Obviously, the identified PIPs can be considered as the time (cutting) points for the segmentation problem (Fu et al., 2006c).

Jiang et al. (2007a), Gong and Si (2013) and Gong et al. (2016) adopt PIP for subsequence segmentation and then use different pattern matching methods, like PCA, templated, rule-based approach, for subsequence pattern matching. PIP is used by Wan and Si (2017c) to serve as the segmentation method to preprocess the sequence for chart pattern matching in financial data. Similarly, Tsinalanidis and Kugiumtzis (2014) use PIP to segment the time series into subsequence. Then, DTW is adopted for finding similar subsequences for prediction. Tao et al. (2016) also adopt PIP to segment the time series into subsequences for codebook generation. The purpose of this work is to improve the classification performance of Piecewise Vector Quantized Approximation (PVQA).

For the PIP segmentation, post-processing step is proposed by Zhang et al. (2007b) and Jiang et al. (2007a) to further merge the segments for reserving the principle trend.

The approaches, we discuss above usually can identify a pattern from a time series but they do not consider the identification of a suitable set of time points to compare with a given pattern templates, taking the technical patterns like double top and H&S as examples for stock analysis. Furthermore, a variety of patterns (e.g. in different
resolutions) must be identified to form a versatile mining space. The segmentation task mentioned before can be considered as an optimization problem and Fu et al. (2001) use evolutionary computation to develop a solution. Fu et al. (2002) and Chung et al. (2004) further fill up the detail of the proposed pattern-based time series segmentation approach. A similar evolutionary architecture is proposed by Yu et al. (2010) and Chen et al. (2013). Unlike their previous work in (Tseng et al., 2008), they also adopt PIP, instead of DWT, to effectively adjust the length of subsequences for finding appropriate segments and patterns to avoid the previous problems.

Yu et al. (2006; 2007) also adopt the evolutionary approach for time series segmentation. They propose to improve the distance measure based on pattern distance. The series of PIPs identified are converted into piecewise trend sequence for the matching process to minimize the PIPs added and lost problems. A pattern-based time series segmentation method is proposed by Deng et al. (2011). The matching process of this approach integrating PIP identification and DTW similarity measure.

Instead of proposing new approach, Gensler and Sick (2014) concentrate on the evaluation of the performance in time series segmentation techniques which based on classification related measures and segmentation zone measures.

6.4 Pattern Discovery

It is a non-trivial task to identify specify patterns and discover frequently appearing patterns from time series data. Given a set of time series pattern templates, it is a common task to identify these templates in a long time series. They may appear in different resolutions within a time series (i.e. time series subsequence). For the problem of time series pattern discovery, a common group of techniques being employed is clustering. Review on time series clustering can be found at (Liao, 2005; Aghabozorgi et al., 2015).
While patterns can be directly discovered from time series, a major problem is that time series data mostly increase linearly with time. This will cause the storage needs to increase rapidly and slow down the pattern discovery process exponentially. Therefore, an effective mechanism for compressing the huge amount of time series data, especially historical data, is needed. This not only reduces the size of storage, but also maintains an acceptable level of information for the discovery process. Fu et al. (2001) propose to adopt PIP identification to solve these problems. Then, the identified PIPs of each subsequence are fitted to the Self-Organizing Maps (SOM) for pattern discovery. In (Fu et al., 2004b), k-means clustering algorithm is adopted to cluster the PIPs of a set of time series data for fast indexing purpose. Categorization of time series subsequences, which after PIP identification, into different classes is also discussed in it. A two-step clustering process are further proposed by Fu et al. (2006e). The number of PIP identified which appropriate to represent the time series is considered as the first step to divide the time series patterns into different groups. Then, clustering with different input vectors are adopted for these different groups of time series for pattern discovery. PIP is adopted to reduce the dimension for time series clustering by Park et al. (2010). PIP identification is used in this research to preserve the salient points in a time series when comparing to SAX.

Zhou and Hu (2009) propose a dynamic PIP identification method to avoid the computation expense problem. The general idea is adopting binary tree to organize the identified PIP in a time series. Then, a three-layer neural network (NN) approach is involved for pattern recognition and window length identification. Similarly, Markowska-Kaczmar and Dziedzic (2008) also adopt NN to recognize technical analysis patterns in financial time series.

However, applying clustering approaches to discover frequently appearing patterns is claimed to be meaningless when focusing on time series subsequence (Keogh et al.,
2003). It is because when using a sliding window to discretize the long time series into subsequences in a fixed window size, patterns, which are derivations from sine curve, are always resulted no matter how the shape of the given time series is. Fu et al. (2005b) purpose an intermediate subsequences filtering process by detecting the change of PIP before the clustering process to solve this problem.

6.5 Visualization

Visualization of time series data attempts to improve the utility of common graph plotting algorithms, by using techniques such as increasing data density or polar-coordinate displays that emphasize the serial periodic nature of the data set, or by distorting the time axis to realize denser information displays. PIP identification for time series data supports multi-resolution visualization. It is extremely suitable for displaying time series on small screen like the mobile environment (Fu et al., 2004c; 2005a). In order to reduce the dimension of the time series data for real-time processing and to represent the main characteristics of time series graphs, Ziegler et al. (2010) also adopt PIP identification as a preprocessing step.

On the other hand, a time series visualization tool called VizTree is proposed by Lin et al. (2004a; 2004b; 2005). This approach firstly converts each numeric time series to a symbol string based on the SAX and a set of substrings (with the same number of symbol) extracts from the symbol string is encoded by a modified suffix tree to visualize the frequency of patterns. That is, the SAX discretizes the original time series into fixed length subsequences, converts each subsequence to a symbol and the symbols obtained are concatenated to form a symbol string. Next, a suffix tree will be constructed. The length of substring is reflected by the depth of the tree. Each branch of the tree represents a pattern. The frequency of the pattern is represented by the thickness of each branch. Different applications of VizTree are suggested by the authors including subsequence matching, frequently appearing pattern discovery and
surprising pattern discovery. Fu et al. (2008a) extend the work to the discovery of interesting patterns across different resolutions by adopting the symbolic representation of PIP instead of SAX in the VizTree. The advantages of this extension are discovering both frequently appearing and surprising patterns across different resolutions, and at the same time, preserving the overall shape of the time series patterns even they are warped and salient points will not be smoothed out.

A system is proposed by Fu et al. (2007b) which performs the analysis and visualization of the emerging Consumer Generated Media (CGM) posts and online news archives in a more user-friendly way. In order to overcome the heavy time complexity incurred, an approach to extract only the useful data from the CGM by PIP is employed. By correlating the sorted out time series data with the online texts, further analysis could be done in a more effective and efficient way. Similar idea is adopted by Fu et al. (2008c) for investigating the correlation between stock prices and news sentiment. The major events of a leased company are identified by the PIPs in its stock price and cross checked with the news.

Burtini et al. (2013a; 2013b) also deal with time series visualization. They propose to reduce the dimensions of time series by using a combination of Piecewise linear and Polynomial Approximation (PPA) for chart generation. They compare the proposed approach with the generic PIP approach.

6.6 Rule Discovery and Prediction

In many researches, prediction is one of the ultimate goals in time series data analysis (Sapankevych and Sankar, 2009). In (Fu et al., 2006b), the identified PIPs are fitted to the Artificial Neural Network (ANN) for the prediction purpose. It is believed that the good performance of the proposed method over traditional one is due to the PIPs identified from the sequence can capture the shape of the subsequences’ patterns.
Prediction based on these patterns is similar to the traditional technical analysis method. Cammarano et al. (2013) focus on energy prediction. They adopt PIP method to identify the N+1 points with the greatest impact on the shape of the daily harvesting profile.

As mentioned in section 4.3, Tsinalanidis and Kugiumtzis (2014) propose a nonlinear prediction scheme for assessing statistically the efficient market hypothesis (EMH) on simulated and real financial price series. PIPs and DTW were combined for the proposed scheme.

Leitao et al. (2016) use PIP identification to reduce the dimensions of the time series data and at the same time, maintains the main characteristics of its data. Then, the relationship between PIPs is mapped to symbols based on SAX. The combination among rules between PIPs and SAX representation is then optimized by Genetic Algorithm (GA) and they call this approach Symbolic Important Rules (SIR). This approach creates investment rules for the buy/sell decision in the stock market.

Yu et al. (2011) propose an approach named PIPs-SAX for mining Emerging Patterns (EPs) from time series data. The proposed approach first transforms the time series data into symbolic representation based on SAX and PIP; that is PIPs are first identified and then converted to symbols by SAX which is similar to the operation in (Fu et al., 2008a). Then, time series EPs are mined with time gap constraint in the symbolic representation.

As mentioned in section 2.5, modified PIP identification approach is used in (Song and Lee, 2013) to find critical points of atypia-amplitude signature for detecting pancreatic ductal adenocarcinoma (PDAC) in the medical field.
6.7 The Integration of PIP and Deep Learning

Classification is a traditional data mining task. In the time series domain, special treatment must be considered due to the nature of the data. A time series is a collection of observations made chronologically. The nature of time series data includes: large in data size, high dimensionality and update continuously. Moreover, time series data, which is characterized by its numerical and continuous nature, is always considered as a whole instead of individual numerical field. Therefore, unlike traditional databases where similarity search is exact match based, similarity search in time series data is typically carried out in an approximate manner.

6.7.1 A Brief Review on Time Series Classification

Geurts (2001) proposes a classification method for time series data based on combining local patterns in the time series. Zhang et al. (2004) present a representation method using wavelet decomposition that can do auto selection of the parameters for classification. They suggest a nearest neighbor classification algorithm with the derived appropriate scale. Kadous and Sammut (2005) develop meta-feature approach for example, local maxima in time series to generate classifiers. Similarly, Yang et al. (2005) use common principle components to build feature subset selection (FSS) which called CleVer that hold the correlation information among original features. Classification is used to evaluate the effectiveness of the feature selection.

More researchers have focused on development of classifiers for time series data. For example, Povinelli et al. (2004) present a signal classification approach built on modeling a dynamics system as they are captured in a reconstructed phase using Gaussian Mixture models of time domain signatures. Rodriguez and Alonso (2004) study interval and DTW-based decision trees that are for the classification. Ensembles are used to combine base classifiers while Wei and Keogh (2006) study the
combination of the numericity reduction using DTW and nearest-neighbor classifiers for time series classification. In addition, Xi et al. (2006) propose a semi-supervised time series classifiers for small labeled example set.

With the significant improvement of the computation power and the abundant data on the Internet, deep learning (LeCun et al. (2015), Schmidhuber (2015)) becomes the most popular method in the AI as well as data mining fields in recent years. Examples in the field of time series data mining like Qiu et al.(2014) and Lv et al. (2015) adopt deep learning for time series forecasting. A review on the uses of deep learning in time series data can be found in (Langkvist et al., 2014).

When further focusing on deep learning for time series classification, Wang et al. (2016) focus on the use of fully convolutional network, Zhao et al. (2017) focus on the use of convolutional neural network and Pankaj et al. (2017) propose TimeNet, a multilayered recurrent neural network. Most of the current related research are working on the study of the neural network architectures. Raw time series data is always to be used as the input data.

6.7.2 The Deep Learning Network Adopted

In recently years, deep learning neural network achieves outstanding performance on many important problems especially in the fields of computer vision, speech recognition and natural language processing due to the powerful computing speed of Graphical Processing Unit (GPU) having been developed to increase the performance. However, its computing time is still a problem. In a certain long time series, a neural network for filtering noise needs a lot of training time and a large number of hidden nodes or layers (physical storage space) to do it. PIP is expected can filter out certain noise and reduce the dimension by taking out the important points for the neural network to recognize. It is working well on the data segments with different lengths as well.
A deep-learning architecture is a multilayer neural network. Each hidden layer can be treated as distorting the input in a non-linear way. Two or more layers can transform the dataset to a linear format. After trained the network to be convergence, the error reaches a rational level. Finally, the format will become linearly separable by the last layer (output layer). Figure 29 shows the architecture of the neural network that we adopted in this research.

![Diagram of neural network architecture]

Figure 29. PIP integrated with Deep learning network architecture

Firstly, the input data will be either the raw data points of the time series pattern or the transformed time domain representation (i.e. PIP or PAA). When using the raw data, the number of inputs is the same as the length of the time series pattern and the maximum is 607 in our cases.

To estimate the appropriate number of PIP required to represent the time series pattern, the method proposed in (Fu et al, 2006c) is adopted. The optimized number of PIP will be the number when the error of representing the time series pattern by the identified PIPs becomes stable. The error is calculated by summing up the Vertical Distance (VD)
between each data point the shape formed by the identified PIPs. Figure 30 shows a visualized sample and the error is calculated by:

\[
VD \text{ Error } = \sum_{i=1}^{n-1} \left( \sum_{y=x+1}^{y-1} \sum_{k=x+1}^{y+1} (VD(S_x, S_y, S_k)) \right)
\]

where \( S \) is the original time series, \( P \) is the identified PIP series, \( n \) is the point number of PIP series. PIP series obtained in Figure 30 is sorted by its position at the original time series, i.e. PIP \([P_1, P_2, P_3, P_4, P_5] = [S_1, S_5, S_7, S_8 & S_{10}]\).

Two hidden layers and one-hot-vector mapping are adopted based on the number of classes to convert the output to the probability of each class.
6.7.3 Comparison on PIP, PPA and Raw Data for Time Series Classification

In this section, the classification ability of using raw and different time series representation methods (i.e. PIP and PAA) will be evaluated. The algorithms are implemented with TensorFlow which built on Python. TensorFlow is an open source software library created by Google for numerical computation using data flow graphs. (http://www.tensorflow.org). It has nodes and graph edges to build a graph to represent the whole working mechanism. Nodes represent mathematical operations while graph edges represent data to be communicated between 2 nodes. The data can be multidimensional arrays (tensors). Operations in nodes can be distributed to GPU or CPU or servers. Figure 31 is a simple example to show how the TensorFlow design a program. It is a Cosine Function used to Measure the similarity between 2 vectors. It demonstrates the dot and multiple operation on these 2 vectors (\(d\) and \(q\)) that can be conducted in the 2 CPU or GPU parallelly.

\[
\text{CosSim}(d, q)
\]

![Figure 31 A simple TensorFlow graphic design](image)

Figure 31 A simple TensorFlow graphic design
The program is run on Ubuntu Linux which is configured as 8 CPU with 32 GB RAM. The general settings for training the deep neural network is 1,000 iterations, 10 samples as a batch and 0.001 as the learning rate.

The first dataset we adopt is a Bird and Chicken dataset which is public available at http://timeseriesclassification.com/description.php?Dataset=BirdChicken and is donated by J. Hills and A. Bagnall. The dataset has 20 training and 20 testing time series patterns. Each pattern is extracted from the outline of either a bird or chicken shape which is mapped to a 1-D series of distance to the center. The length of the raw time series patterns is 512. Figure 32 shows that the birds and chickens have similar shape in the dataset. Comparing the 2 classes in detail, the time series patterns have fundamental differences like the bird class has a deep “V” shape. Some of them are at the tail like #4, #8 and some are at the middle like #5, #12, #14. Figure 33 and Figure 34 are the training set of bird and chicken shown in graphic mode to compare how PIP, PAA and Raw Point used to represent the dataset.

Figure 32 Binary image of Bird and Chicken dataset (From: http://timeseriesclassification.com)
Figure 33 Bird Class Training set
Although the shapes of the birds and chickens can be identified by a neural network with 1 hidden layer, as they have different rotations, the deep learning network needs one more hidden layer to handle the rotation (transformation) of the image. Therefore, a two-hidden-layer network is used.

First, it is necessary to estimate the number of PIP for representing these patterns based on the method described in Section II. Figure 35 shows the error (PIP-VD) when representing the training set by different numbers of PIPs.

Starting from 7 PIPs, the error is decreased steadily and there is a drastic decrease before 7. The best number of PIPs should be around 6 to 8. Therefore, the accuracy of the testing data classification of using 6, 7 and 8 PIPs are further evaluated across different numbers of hidden nodes and the result is shown in Figure 36. The 7 PIPs have the best result, and this verified the result in Figure 35.
Based on 7 PIPs, the results among raw time series data and representing time series by PIP and PAA are compared. 7 PAA is also adopted to align with PIP. Figure 37 shows the result when different numbers of hidden nodes is used. In all the cases, PIP representation performs the best.
Neural network is trained with each config for 10 times and take out the maximum accuracy as the result.

Table 2 shows that each image in training and testing set go through testing of the trained neural network. 1 stands for correct and 0 stands for incorrect. PIP and PAA have all correct in training set but only Raw Point has 1 incorrect on train set #12, ref. Title “Bird Class Train Set: 12” in Figure 33. That means PIP and PAA having less points, so the neural network can memorize all images, but the raw point has too much point, 512 points, that the neural network cannot memorize all correctly.

<table>
<thead>
<tr>
<th>Image</th>
<th>PIP</th>
<th>PAA</th>
<th>Raw Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train #1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train #2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train #3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train #4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train #5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train #</td>
<td>Test #</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #6</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #7</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #8</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #9</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #10</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #11</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #12</td>
<td>1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #13</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #14</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #15</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #16</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #17</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #18</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #19</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Train #20</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #1</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #2</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #3</td>
<td>0 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #4</td>
<td>1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #5</td>
<td>1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #6</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #7</td>
<td>1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #8</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #9</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #10</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #11</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #12</td>
<td>1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #13</td>
<td>1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #14</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #15</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #16</td>
<td>1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #17</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #18</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #19</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #20</td>
<td>1 1 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Detail classification result of the Bird and Chicken testing dataset
Figure 38 shows the testing patterns that PIP and PAA are both correctly recognized, but it is recognized wrongly by using the raw data. There are too many turning points and some near-by turning points at the same level. PIP can filter or discard these extra turning points and PAA can smooth out the extra turning points but they are existed as noise in the original pattern.

Figure 38 Testing patterns that PIP and PAA recognize it correctly but not in original time series pattern (i.e. #4, #12 & #13)

Figure 39 shows the testing pattern that only PIP recognizes correctly. Further to the observation above, PAA has smoothed out the turning point in this case. The pattern has 5 turning points that PIP identify it very clearly but PAA only shows 3 turning points and this causes the wrong recognition.
The second experiment is conducted on a large set of financial technical patterns which is generated by the method specified in (Fu et al., 2008). 10,000 time series patterns with 607 data points are generated. Each of them belongs to one of the five technical patterns (i.e. classes): head-and-shoulder (H&S), double tops, triple tops, rounded top and spike top. Each technical pattern was used to generate 2,000 variants by applying different levels of scaling, time wrapping and noise. First, the patterns are uniform time scaling from 7 data points to 607 data points. Then, each critical point of the patterns can be warp between its previous and next critical points. Finally, noise is added to the set of patterns. Adding noise is controlled by two parameters, namely, the probability of adding noise for each data point and the level of noise being added to such point. The detail operation can be found in (Fu et al., 2008b). Figure 40 shows a few samples of each class. The dataset is divided in 8,000 training data and 2,000 testing data.
Based on the training data, we first counter check the optimized PIPs required to represent the patterns. As shown in Figure 41, 7 PIPs is suggested by the method as described in Section 6.7.2.
Figure 41 Error (i.e. PIP-VD) by representing the synthetic financial training set with different numbers of PIPs

Based on 7 PIPs, the result among raw time series data and representing time series by PIP and PAA is compared as previous section. 7 PAA is also adopted to align with PIP. Figure 42 shows the result when different numbers of hidden nodes is used. In all the cases, PIP representation performs the best.

Figure 42 Accuracy of the synthetic financial testing dataset classification using different numbers of hidden nodes and different input data
In addition, **Figure 43** shows the time series used to train the deep neural network. As the number of input nodes is greatly reduced from 607 to 7 when PIP or PAA representation is adopted, the training speed can be greatly reduced when comparing to the use of raw time series pattern data as the input.

![Figure 43 Speed for training the deep neural network by using the raw data, PIP and PAA representation.](image)

**Figure 44** shows a synthetic pattern that both PIP and PAA can recognize correctly but it cannot be recognized by using raw data which serving as the input of the neural network. Both PIP and PAA representations can keep the shape of the round top pattern but original data has too many turning points. Furthermore, **Figure 45** shows a synthetic triple tops pattern which using the raw data and the PIP representation can recognize it correctly but not in PAA representation. As the second and third tops are very close, PAA representation smooths out the bottom point between the second and third tops and this effect leads to the second and third tops cannot be shown clearly.
Although current deep learning network can work well on the raw data, experiments in previous section demonstrated that transforming the original time series pattern data to the time domain representations could obtain better result. It is because representation like PIP has its own advantages over working on the raw data when the shapes of the time series patterns are important with the present of noise and shape distortion. By increasing the number of nodes in the hidden layers, we have preliminary tested and improvement in terms of accuracy can be achieved when raw time series pattern data is used. However, for time critical tasks like real-time monitoring applications, time domain representations can obviously provide better solution as shown in Figure 43.

On the other hand, the input time series patterns should have the same length to serve as the input of the deep learning network. However, in many cases, we would like to capture the patterns from multi-resolution time series such as financial technical
patterns in long-term and short-term considerations. This can only be achieved by applying time domain representation after transforming the time series patterns in different lengths to the same size to serve as the input. This is another important characteristic and benefit that using raw input data cannot achieve.

To sum up, time domain representation for time series data in deep learning has its own advantages even the computational power of current system can handle the learning process with high dimension raw input data.
Chapter 7 Conclusions

This research first revisited the development of Perceptually Important Point (PIP) identification. We also reviewed the applications of PIP in different areas, which include: dimensionality reduction, similarity measure and pattern matching, segmentation, pattern discovery, time series data visualization, and also rule discovery and prediction, in the past 15 years. Different applications already demonstrate the usability of PIP.

However, due to the performance issue of the PIP identification process, one of the major contributions of this research is to propose algorithms in both algorithmic level (namely Caching and Splitting) and distributed environment (namely Distributed Distance Calculation Approach (DDCA) and Distributed SB-Tree Building Approach (DSBA)). According to the evaluation, significant speed improvement is obtained by adopting these improvement algorithms. The proposed algorithms solve the bottleneck when adopting PIP identification process for the Big time series data analytics.

In addition, a study on the roles of time domain representation for time series classification using deep learning network is conducted. Although the current computational power allows the neural network to learn directly from the raw data, experiments show that time domain representation of original time series pattern data like PIP and PAA provide additional benefits on filtering noise and handling distortion and obtain better classification result. While the accuracy may further be increased by increase the number of hidden nodes or even the number of hidden layers when working on raw data, the training time will increase accordingly and not suitable in many time critical applications like real-time streaming data analysis. In other words, by reducing the number of input nodes using time domain representation, a simpler deep learning network is needed for providing similar or even better classification
result in a much shorter time. In the future, we are planned to test on different types of deep learning network like convolutional neural network and different types of time series pattern data.

By solving the performance issue of the PIP identification process and the demonstration of its usage in deep learning, it is expected that more research can be conducted on adopting PIP identification process on various Big time series data analysis.
References


Fu, T.C., Cheung, T.L. and Chung, F.L. An Innovative Use of Historical Data for Neural Network Based Stock Prediction. *In Proceeding of the 5th International Conference on Computational Intelligence in Economics and Finance*, 2006b, 235-238.


Fu, T.C., Chung, F.L., Kwok, K.Y. and Ng, C.M. Stock Time Series Visualization Based on Data Point Importance. *Engineering Applications of Artificial Intelligence, 21*(8), 2008a, 1217-1232.


Fu, T.C., Chung, F.L., Luk, R. and Ng, C.M. Preventing Meaningless Stock Time Series Pattern


Fu, T.C., Chung, F.L., Luk, R. and Ng, C.M. Representing Financial Time Series Based on Data Point Importance. Engineering Applications of Artificial Intelligence, 21(2), 2008b, 277-300.


Analysis, 2007b, 259-262.


Li, Q., Moon, B., and Lopez, F.V. Skyline index for time series data. *IEEE Transactions on Knowledge and Data Engineering*, 16(6), 2004, 669-684.


Povinelli, R.J., Johnson, M.T., Lindgren, A.C. and Ye, J. Time Series Classification Using Gaussian Mixture Models of Reconstructed Phase Spaces. IEEE Transactions on Knowledge and Data Engineering, 16(6), 2004, 779-783.


Song, C., Pei, T. and Yao, L. Analysis of the Characteristics and Evolution Modes of PM 2.5 Pollution Episodes in Beijing, China During 2013. Internation Journal of Environmental Research and Public Health, 12, 2015, 1099-1111.


Tortonesi, M., Michaelis, J., Morelli, A., Suri, N. and Baker, M.A. SPF: An SDN-based


Xi, X., Keogh, E., Shelton, C., Wei, L. and Ratanamahatana, C.A. Fast Time Series


