Journal of Computer Science 6 (12): 1389-1395, 2010
ISSN 1549-3636
© 2010 Science Publications

Index Financial Time Series Based on Zigzag-Perceptually Important Points

[1]Chawalsak Phetchanchai, [2]Ali Selamat, [3]Amjad Rehman and [2]Tanzila Saba
[1]Department of Computer Science, Faculty of Science and Technology,
Suan Dusit Rajabhat University, Bangkok, Thailand, 10300
[2]Department of Software Engineering, Faculty of Computer Science and Information Systems,
University Technology Malaysia, 81300 Skudai, Johor, Malaysia
[3] Department of Computer Science, Faculty of Computer Science and Information Sciences
Al-Imam Muhammad ibn Saud Islamic University, Riyadh, Kingdom of Saudi Arabia

Abstract: Problem statement: Financial time series were usually large in size, unstructured and of high dimensionality. Since, the illustration of financial time series shape was typically characterized by a few number of important points. These important points moved in zigzag directions which could form technical patterns. However, these important points exhibited in different resolutions and difficult to determine. **Approach:** In this study, we proposed novel methods of financial time series indexing by considering their zigzag movement. The methods consist of two major algorithms: first, the identification of important points, namely the Zigzag-Perceptually Important Points (ZIPs) identification method and next, the indexing method namely Zigzag based M-ary Tree (ZM-Tree) to structure and organize the important points. **Results:** The errors of the tree building and retrieving compared to the original time series increased when the important points increased. The dimensionality reduction using ZM-Tree based on tree pruning and number of retrieved points techniques performed better when the number of important points increased. **Conclusion:** Our proposed techniques illustrated mostly acceptable performance in tree operations and dimensionality reduction comparing to existing similar technique like Specialize Binary Tree (SB-Tree).

Key words: Financial time series indexing, important points, ZM-Tree, time series

INTRODUCTION

Financial time series including stock prices, exchange rates and dollar indices are exhibited as a sequence of observed data that are generated chronologically. As financial time series are usually large in size, unstructured and of high dimensionality, the method to locate the time series of interest is a nontrivial problem. Mining these data is usually taken high costs. Indexing these financial time series is one of the possible solutions to increase the processing performance (Basu and Meckesheimer, 2007).

The illustration of the financial time series shape is typically characterized by a few important points within the multi-resolution views (Fu *et al*., 2008). These important points represent the trend movement changes of up and down directions which are formed in a zigzag manner in a specific view. A zigzag movement can be considered as a zigzag feature, which is observed by filtering out random noise. A zigzag feature demonstrates the past performance trends and only the most important changes. Most of the uses of a zigzag feature in technical analysis are related to the identification of reversal patterns, price-retracement measures and counting of corrective waves.

In this study, we have proposed a financial time series indexing approach. Indeed, we have also proposed a method for identifying the important zigzag points in multi-resolution, namely the Zigzag-Perceptually Important Points (ZIPs) identification method. Furthermore, by using our proposed Zigzag-based M-ary Tree (ZM-Tree), we were able to index these important zigzag points.

Related study:

Indexing a time series: Indexing a time series is necessary for several researches on similarity searching (Denton *et al*., 2009). A fast subsequence matching technique that allows similarity search among time series of different sizes is proposed in (Agrawal *et al*.,

Corresponding Author: Chawalsak Phetchanchai, Department of Computer Science, Faculty of Science and Technology, Suan Dusit Rajabhat University, Bangkok, Thailand, 10300

1993). They extracted a time-series feature and divided it into several subsequences by using a sliding window. Each subsequence was represented by a Minimum Bounding Rectangle (MBR), which was further indexed and stored using an ST-index. Furthermore, Lin *et al*. (2003) proposed a new method based on Piecewise Aggregate Approximation (PAA) (Yi and Faloutos, 2000; Bhansali and Ippoliti, 2005), known as Symbolic Aggregate Approximation (SAX). SAX allows a time series of arbitrary length s to be reduced to equi-length segmented time series p by using PAA where p<<s and then symbolizes the PAA representation into a discrete string. SAX was later improved as indexable SAX or iSAX and indexed for mining terabyte-sized time series (Shieh and Keogh, 2008). However, the above-mentioned approaches of indexing focus on dividing a time series into several subsequences without concentrating on the important points that are represented in the financial time series.

Important points are very significant as well as nontrivial in financial time series analysis (González-Concepción *et al*., 2009). Ali (2006) introduced the Perceptually Important Points (PIPs) identification method for using in patterns matching of financial applications. Later, Fu *et al*. (2008) proposed a method to index PIPs more structurally by means of the tree data structure known as the Specialized Binary Tree (SB-Tree), which provided the benefits of efficient computation of cumulative new data points, retrievals and access. Unfortunately, the representation of important points in a time series was not in the zigzag form. Smadi and Mjalli (2007) proposed a Landmark Model that relies on human spatial memory. The Landmark Model produced a zigzag compressed time series. The model was observed to collect the greatest important points in a time series with a specific Minimal Distance/Percentage Principle (MD/PP) that has its own intuitive meaning. For example, if a stock trader trades once for a week (5 business days) and regards a 10% gain or loss as significant, then MD is set to 5 and PP is set to 10. The Landmark Model represents a time series that filters out the noise and retains only the significant important points underlying the specific MD/PP. For querying the Landmark sequence, they recommended that the Landmark sequence must be more similar to a string sequence, rather than a multidimensional sequence; thus, string indexing is more suitable than the use of R-tree. Similar concepts regarding the identification of important points can be found in studies conducted in (Pratt and Fink, 2002; Fink and Pratt, 2003). They compressed a time series by selecting some of the minima and maxima or important points of peaks and troughs and dropping the other points. The intuitive idea is to discard the minor fluctuations and retain the major minima and maxima. The compression rate is controlled using a parameter R, which is always >1; an increase in R leads to the selection of fewer points. In addition, they also developed a technique for indexing compressed series, which supports the retrieval of series similar to a given pattern by considering their major inclines that are upward and downward segments of the series. However, these techniques do not illustrate the data structure as being well-organized and store the identified important points.

Perceptually important points identification method: The concept of Perceptually Important Points (PIPs) identification is based on the importance of data points (Ali, 2006). This importance is defined by the domination of a data point on the shape of the time series. A data point that has a greater domination on the overall shape of the series is considered to be more important.

For a given time series T, all the data points, t_1, t_2, t_3....... t_m in T will go through the PIP identification process. Initially, the first two PIPs are collected from the first and the last points of T. The next PIP will be the point in T with the greatest distance to the first two PIPs. Subsequently, the fourth PIP will be the point in T with the greatest distance to its two adjacent PIPs, either between the first and second PIPs or between the second and the last PIPs. The process of locating the PIPs continues until all the points in T are attached to a list. To calculate the distance to the two adjacent PIPs, three data point importance evaluation methods have been proposed. Fu *et al*. (2008) introduced three methods to evaluate the importance of the PIPs in a time series, namely Euclidean Distance (PIP-ED), Perpendicular Distance (PIP-PD) and Vertical Distance (PIP-VD), as shown in Fig. 1.

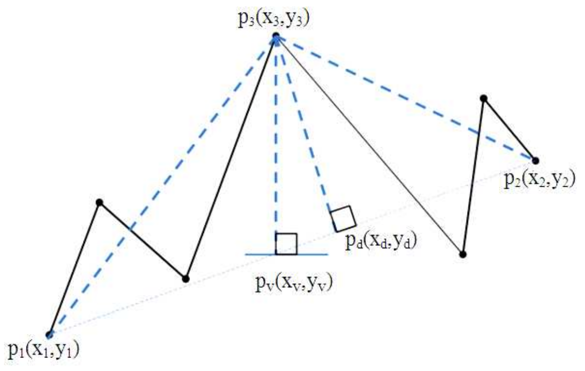

Fig. 1: PIP measurements

PIP-ED ((p_1p_3, p_3p_2) calculates the sum of the Euclidean distance of the test point to its adjacent important points; PIP-PD (p_3p_v) calculates the perpendicular distance between the test point and the line connecting the two adjacent PIPs and PIP-VD (p_3p_v) calculates the vertical distance between the test point and the line connecting the two adjacent PIPs.

MATERIALS AND METHODS

Zigzag-perceptually important point identification method: For a time series T = {t_1, t_2, t_3....... t_m}, where m is the number of time instances, the main idea of ZIP algorithm is to collect the important points that influence the zigzag shape of the time series. These important points are collected from a time series by evaluating the distance of the point to its adjacent important points.

The selected data point importance evaluation is based on the Vertical Distance (VD) (González-Concepción *et al*., 2009). However, in our case, we made some modifications in the VD measurement to evaluate their zigzag states by removing the absolute function according to the sign of the VD for the later evaluation of their Zigzag Turning Signals (ZTS) ('+' if VD is less than 0 or '–' if VD is more than 0). The modified function of VD can be depicted as follows:

$$VD(p_z, p_v) = y_v - y_3 = (y_1 + (y_2 - y_1)\frac{x_v - x_1}{x_2 - x_1}) - y_3$$

Where:

(x_1, y_2) and (x_2, y_2)	=	The coordinates of the points at the start and end of a segment or p_1 and p_2, respectively
p_3 (x_3, y_3)	=	A point whose VD is determined by comparing it with p_1 and p_2
p_v (x_v, y_v)	=	A projection of p_3 on the line connecting p_1 and p_2

The following is the algorithm for the identification of ZIPs.

Algorithm 1: ZIP Identification:

```
Input: sequence T[1..m]
Output: ZIPList S[1..n]
  Begin
    Set S[1] = T[1], S[2]=T[m];
    Repeat until all segments are marked as 'N'
       Begin
          Select Point p of a segment that it is-
            not marked as 'N'; with maximum-
            distance to the adjacent ZIPs in S.
          Assign ZTS to p
          If ZTS of p equals to one of its-
            adjacent ZIPs then
              Select point q of this subsequence-
                with maximum distance to -p
                and an adjacent point in S
              Assign ZTS to q
              If ZTS of q equals to one of its-
                adjacent ZIPs or q is not available
                  Mark this segment as 'N'
              Else
                  Add p, q and their ZTSs to S
              End
            Else
              Add p and its ZTS to S
            End
          End
        End
```

Zigzag-perceptually important point identification method: To represent the ZIP identification process, we have illustrated each step of the method graphically. The empirical example is a synthetic data sequence with 17 data points. Figure 2 depicts the steps of ZIP identification, Table 1 shows the ZIP list arranged based on their importance level collected using ZIP identification process of Algorithm 1.

Time series indexing: To structurally organize the important points, indexing technique of financial time series based on ZIP identification is presented. The ZIPs collected from the ZIP identification method have been used to construct the tree data structure. The constructed tree is a type of M-ary tree known as the ZM-Tree. The ZM-Tree follows the M-ary Tree constraints with some additional constraints. The followings are discussions about the ZM-Tree.

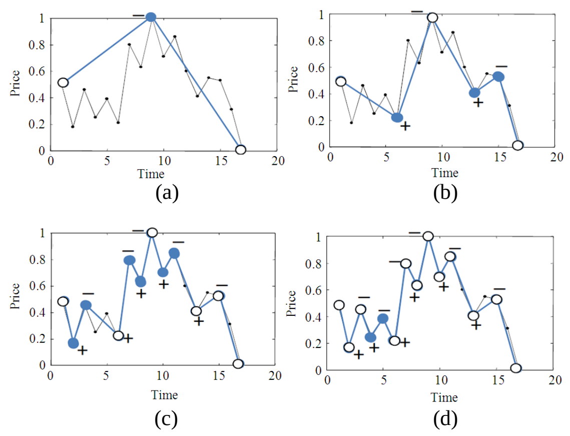

Fig. 2: Steps of ZIP identification process

Table 1: ZIP LIST from ZIP-identification process

Level	ZIP	y	VD	ZTS
0	1	0.50	n/a	n/a
0	17	0.00	n/a	n/a
1	9	1.00	0.75	-
2	6	0.21	-0.60	+
2	15	0.53	0.28	-
2	13	0.41	-0.28	+
3	2	0.18	-0.26	+
3	3	0.46	0.27	-
3	7	0.80	0.33	-
3	8	0.63	-0.27	+
3	10	0.86	-0.16	+
3	11	0.71	0.22	-
4	4	0.25	-0.13	+
4	5	0.39	0.16	-
0	5	10.00	15	20

The ZM-tree structure: The proposed ZM-tree is a type of incomplete M-ary tree, where M indicates the number of available children for each node, which is equal to 3. The ZM-tree is an ordered tree in which:

- The nodes hold 1-2 distinct keys
- The keys in each node are sorted
- A node with k elements has k+1 subtrees, where the subtrees may be empty
- The ith subtree of a node [v_1,..... v_k], 0<i<k, may hold only values v in the range of $v_i<v<v_{i+1}$ (v_0 is assumed to be equal-infinity and v_{k+1} is assumed to be equal + infinity)

Furthermore, the ZM-Tree comprises constraints and information additional to those for the M-ary tree. The root contains exactly two keys of the ZIPs from level-zero important points and each node has one or two key (s) depending on the number of retrieved ZIPs on that iteration level. The information stored in each key comprises ZIP and ZTS (placed over the key block).

The node structure and contents: There are three types of nodes available in ZM-Tree: a single-key node, a double-key node and a root node. A single-key node is a node that contains only one key, which holds the available two children at the left and right subtree. A double-key node is a node that contains two ordered keys, which holds the available three children at left, middle and right subtree. Lastly, a root node is a special kind of a double-key node, which holds only the available one child at the middle key, while the left and right keys are set to null. A node can be a single-key node or a double-key node based on the collected ZIPs of a subsegment at the specific level.

Each node contains a key or keys depending on its type. However, each key also comprises some necessary information; e.g., a key's ZTS and its VD. ZTS is placed on the top of the corresponding key of each node, except the root node, in which there is no ZTS. On the other hand, VD is stored in a key structure along with a key value, which is necessary for tree pruning or tree-retrieval activities.

Tree construction: To build a ZM-Tree, first, all the ZIPs are collected and maintained in a sequential ZIPs list. The root of the tree is created by reading the first two ZIPs from the ZIPs list and storing them in the root node. Subsequently, all the other ZIPs are read and put to the appropriate nodes and positions. The algorithm for creating a ZM-Tree can be illustrated as follows:

Algorithm 2: Insert a new key to the ZM-tree:

```
cur ← root of the tree;
prev ← null;
IF (root is null) THEN
        root ← create a new TreeNode(newKey);
ELSE
        pos ← seek_position(root, newKey);
IF(pos is the tree root) THEN
        IF (newKey.zip < pos.key(1).zip) THEN
          pos.key(1) ← pos.key(1);
          pos.key(2) ← newKey;
        ELSE
          pos.key(2) ← newKey;
        END IF
ELSE
        IF (pos.key(2) is null) THEN
        IF (pos.key(2).level equals to newKey.level)
            THEN
             IF (newKey.zip < pos.key(1).zip) THEN
                 pos.key(2) ← pos.key(1);
                 pos.key(1) ← newKey;
             ELSE
                 pos.key(2) ← newKey;
             END IF
        ELSE
             IF (newKey.zip < pos.key(1).zip) THEN
                   pos.left ← new TreeNode(newKey);
             ELSE
                   pos.right ← new TreeNode(newKey);
             END IF
        END IF
ELSE
        IF (newKey.zip < pos.key(1).zip) THEN
             pos.left ← new TreeNode(newKey);
        ELSE IF (newKey.zip < pos.key(2).zip)
             THEN
             pos.middle ← new TreeNode(newKey);
        ELSE
```

```
                pos.right ← new TreeNode(newKey);
              END IF
     END IF
     END IF
     END IF
     RETURN root;
End Insert
```

Algorithm 3: Seek a position in a ZM-Tree for inserting a new key:

```
  cur ← root;
  prev ← null;
  REPEAT WHILE cur < > null
  BEGIN
  prev ← cur;
    IF (cur is root) THEN
      IF (cur.key(1) is null) THEN
         cur ← cur.key(1);
      ELSE IF (cur.key(2) is null) THEN
         cur ← cur.key(2);
      ELSE
         cur ← cur.middle;
      END IF
    ELSE
       IF (cur.key(2) is null) THEN
          IF (cur.key(2).level equals to
          newKey.level) THEN
             cur ← cur.key(2);
          ELSE
             IF (newKey.zip < cur.key(1).zip) THEN
                cur ← cur.left;
             ELSE
                cur ← cur.right;
             END IF
                END IF
             ELSE
                IF (newKey.zip < cur.key(1).zip)
                     THEN
                     cur ← cur.left;
                ELSE IF (newKey.zip < cur.key(2).zip)
                 THEN
                    cur ← cur.middle;
                 ELSE
                    cur ← cur.right;
                  END IF
                END IF
             END IF
           END
      RETURN prev;
End Seek_position
```

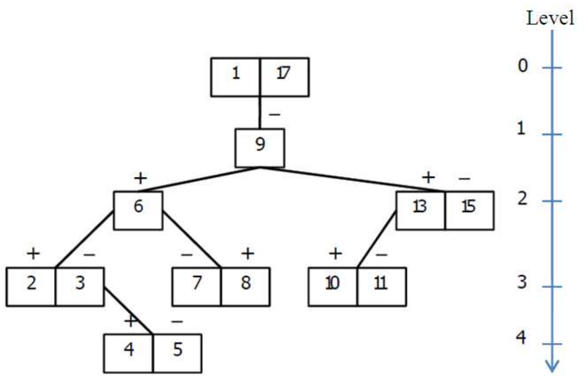

Fig. 3: ZM-tree representation of ZIPs from Table 1

Example of tree construction: To illustrate the tree construction, the ZIPs from Table 1 are used. The constructed ZM-Tree is shown as Fig. 3.

Basic tree operations:
Retrieval: To retrieve a time series from the ZM-tree, the data points are retrieved according to their importance.

The retrieval process starts from the root to their leaves, recursively. The steps of the retrieval process can be depicted as follows:

- The first two important points stored in the root are retrieved and put in the sorted heap
- The levels of importance and VDs of the children nodes are determined. The children nodes are retrieved in an ordered manner from left to right. Hence, the nodes may comprise one or two keys and in this case, all the keys in a node must be retrieved at the same time
- The retrieval can be done recursively until the maximum number of the required ZIPs is reached or until the last node is retrieved

Tree pruning: In mining of financial time series, dimensionality reduction is very important in many tasks; e.g., pattern matching and stock trading. Dimensionality reduction aims to preserve the nature of the shape of the time series by using a minimum number of outstanding important points. A few numbers of important points representing a time series may cause a very large number of errors. However, an increasing number of important points may result in a decrease in the number of errors. Fu *et al*. (2008) proposed two methods of dimensionality reduction using SB-Tree. The first method is based on tree pruning method, in which the less important points are filtered according to a specified threshold VD value.

The second method is an error threshold approach, in which the time series is compressed based on determining the error of the representation, when compared with the original time series.

RESULTS

For evaluation of our proposed methods; ZIP identification, indexing and time series dimensionality reduction, the experiments were implemented with Java programming language performed on IBM compatible PC (Windows XP Professional, CPU Intel Pentium IV 1.7 GHz, RAM: 512 MB). Our data set was the daily close price stock index time series of Stock Exchange of Thailand (SET) collected between January 4, 1999 and July 31, 2006. The indexing performance was evaluated by examining the data set, by measuring the CPU cost against the number of ZIPs. The various numbers of ZIPs employed to build the tree show that the increase in the number of ZIPs causes a linear increase in the time to build the tree, as can be seen in Fig. 4. Similarly, the CPU cost of tree retrievals appear linearly increase when the number of retrieved ZIPs increase or the tree height increase as shown in Fig. 5 and Fig. 6 respectively.

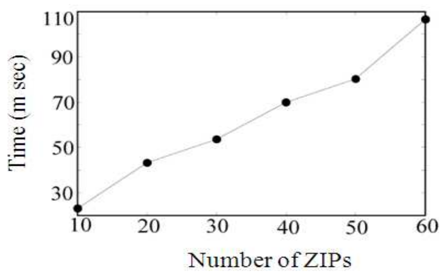

Fig. 4: CPU cost of tree construction against number of ZIPs

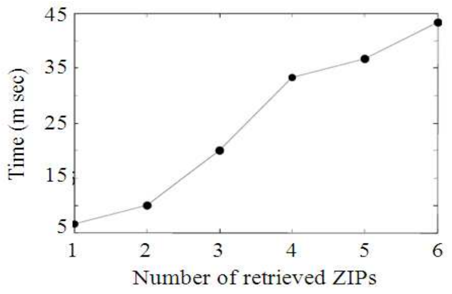

Fig. 5: CPU cost of tree retrieval against the numbers of retrieved ZIPs

Finally, we investigated the task of dimensionality reduction by using the tree-pruning approach. The increasing of pruning threshold causes the error increase instantly (Fig. 7). The comparison of the errors of dimensionality reduction demonstrated that the costs of dimensionality reduction in ZM-Tree and SB-Tree approaches are very high when using a very few number of important points and the number of errors decreases quickly when the number of important points is increased. The graph shown in Fig. 8 indicates a slight decline when the number of important points is >12. Furthermore, the examples of the comparison of reconstructed time series of the retrieved important points with their original time series are illustrated in Fig. 9 which shows its zigzag shape movement time series.

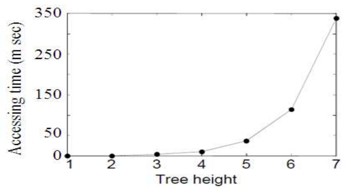

Fig. 6: CPU cost of the tree retrievals against the tree-height level

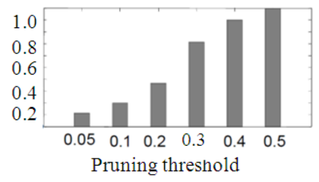

Fig. 7: Tree-pruning thresholds and their errors

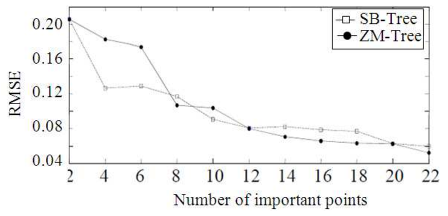

Fig. 8: Comparison of dimensionality reduction errors between ZM-tree and SB-tree

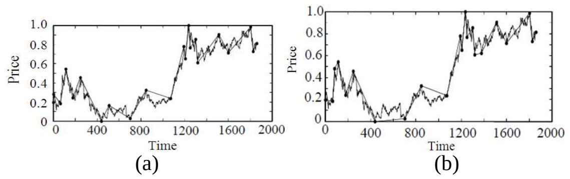

Fig. 9: Comparison of dimensionality reduction errors for (a) ZM-Tree and (b) SB-Tree

DISCUSSION

Our study experimental results show that the proposed technique of financial time series indexing using ZM-Tree illustrates in acceptable performance comparing the similar existing technique like SB-Tree. Furthermore, ZM-Tree represents the reconstructed time series in zigzag manners which are useful in the works of pattern mining.

CONCLUSION

In this study, we have proposed novel methods of financial time series indexing by considering their zigzag movement. The methods consist of two major algorithms: first, the identification of important points, namely the ZIPs identification method and next, the indexing method by applying ZM-Tree. The ZM-Tree data structure clearly illustrates its acceptable benefits in dimensionality reduction, when compared with the SB-Tree. However, future experiments must be carried out using several sizes of time series and various data sets, to clearly compare its performance in various conditions.

REFERENCES

Agrawal, R., C. Faloutsos, A.N. Swami, 1993. Efficient similarity search in sequence databases. Proceeding of the 4th Conference on Foundations of Data Organization and Algorithms, Oct. 13-15, Springer-Verlag, London, UK., pp: 69-84. http://portal.acm.org/citation.cfm?id=652239

Ali, A.A., 2006. On optimistic concurrency control for real-time database systems. Am. J. Applied Sci., 3: 1706-1710. http://www.scipub.org/fulltext/ajas/ajas321706-1710.pdf

Basu, S. and M. Meckesheimer, 2007. Automatic outlier detection for time series: An application to sensor data. Knowl. Inform. Syst., 11: 137-154. DOI: 10.1007/s10115-006-0026-6

Bhansali, R.J. and L. Ippoliti, 2005. Inverse correlations for multiple time series and Gaussian random fields and measures of their linear determinism. J. Math. Stat., 1: 287-299. http://www.scipub.org/fulltext/jms2/jms214287-299.pdf

Denton, A.M., C.A. Besemann and D.H. Dorr, 2009. Pattern-based time-series subsequence clustering using radial distribution functions. Knowl. Inform. Syst., 18: 1-27. DOI: 10.1007/s10115-008-0125-7

Fink, E. and K.B. Pratt, 2003. Indexing of compressed time series. Data Min. Time Ser. Databases, 1: 51-78.

Fu, T.C., F.L. Chung, R. Luk and C.M. Ng, 2008. Representing financial time series based on data point importance. Eng. Applied Artif. Intel., 21: 277-300. DOI: 10.1016/j.engappai.2007.04.009

González-Concepción, C., M.C. Gil-Fariña and C. Pestano-Gabino, 2009. The numerical computation of rational structures and asymptotic standard deviations in causal time series data. J. Mathe. Stat., 5: 215-225. http://www.scipub.org/fulltext/jms2/jms253215-225.pdf

Lin, J., E. Keogh, S. Lonardi and B. Chiu, 2003. A symbolic representation of time series, with implications for streaming algorithms. Proceedings of the 8th ACM SIGMOD workshop on Research Issues in Data Mining and Knowledge Discovery, June 13-13, ACM Press, San Diego, California, pp: 2-11. DOI: 10.1145/882082.882086

Pratt, K.B. and E. Fink, 2002. Search for patterns in compressed time series. Int. J. Image Graph., 2: 89-106.

Shieh, J. and E. Keogh, 2008. iSAX: Indexing and mining terabyte sized time series. Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 24-27, ACM Press, Las Vegas, Nevada, USA., pp: 623-631. DOI: 10.1145/1401890.1401966

Smadi, M.M. and F.S. Mjalli, 2007. Forecasting air temperatures using time series models and neural-based algorithms. J. Math. Stat., 3: 44-48. http://www.scipub.org/fulltext/jms2/jms23244-48.pdf

Yi, B.K. and C. Faloutos, 2000. Fast time series indexing for arbitrary Lp norms. Proceedings of the 26th International Conference on Very Large Data Bases, Sept. 10-14, Morgan Kaufmann Publishers Inc., San Francisco, CA., USA., pp: 385-394. http://portal.acm.org/citation.cfm?id=645926.671689