Learning Regularity in an Economic Time-Series for Structure Prediction

D. Bhattacharya, J. Mukhoti, A. Konar, *Senior Member, IEEE* and A. Saha

***Abstract*— Although an economic time-series apparently looks like a random sequence of data points, there exists certain regularity in the functional behavior of the series. This paper attempts to identify the regularly occurring structures in an economic time-series with an aim to describe the series as a specific sequence of such structures for forecasting applications. The applications include prediction of the most probable structure with its expected duration, and the sequence of time-series values lying on the forecasted structure. Representation of a time-series by regularly occurring structures is undertaken using three main steps: i) segmentation of the time-series into time-blocks of non-uniform length, ii) clustering of the generated segments to identify the recurrent patterns, and iii) representing the sequence of regular patterns using a specially designed automaton. The automaton is used here to both encode the sequence of structures representing the time-series and also to act as an inference engine to stochastic forecasting about the time-series. Extensive experiments have been undertaken to demonstrate the performance of the proposed technique in structure prediction. The results obtained thereby are very encouraging with an average precision of 78.43%, an average recall of 81.97% and an average accuracy of 76.03%.**

***Index Terms*— Knowledge acquisition, time-series segmentation, multilevel clustering, stochastic automaton, business forecasting**

I. Introduction

A time-series represents a discrete time-valued function indicating population growth [1], variation of atmospheric temperature [2] and rainfall [3], economic growth [4] and the like. Most of the existing works on time-series are concerned with modeling of a given time-series using deterministic [5], stochastic [6], neural [7-9], fuzzy [10-14], chaos theoretic [15] approaches and the like [16], primarily to predict the time-series value at the next time point from its current and preceding functional values. Besides the above, there exist a few research works dealing with matching [17], segmentation [18-26], dimensionality reduction [27] and clustering [28] of time-series. One important but scarcely studied aspect of time-series research is structure prediction, where the motivation is to recognize primitive structures embedded in the series along with their sequencing to predict the next structure at the current time-point. This is important in economic time-series, as the predicted structures offer an insight into the temporal behavior of the time-series aiding investors and traders in their action-planning.

Determining structures (consecutive set of ordered data points having a specific geometry and regularity) in a time-series is a complex problem because of the lack of knowledge about the structure shape, duration, the location of the end-points of the structures and their counts. This paper attempts to automatically recognize the repetitive structures present in the time-series in the absence of knowledge about the above characteristics of the time-series. One possible approach to handle the problem is to segment the time-series using certain heuristics (thumb-rules), capable of capturing the fundamental moves in the time-series. For instance, in an economic time-series continuous rise, continuous fall and sideways movement are fundamental moves. Keeping this in mind, we segment a time-series into time-blocks of interest by using the following heuristic: mark the time-points where a change-over in fundamental moves (rise/fall/near zero slope) takes place, and hence obtain the segments between each pair of consecutive marked points. In fact, such a heuristic yields quite a few segments from a large daily time-series of several years.

An in-depth survey of the segments reveals that they can be grouped into structurally similar categories, even though the members in a group have a wide disparity in their dynamic range. This motivated us to first normalize the dynamic range of all the generated segments and then pass them through an automatic clustering [29] process to identify the cluster centers, representative of the regularly occurring structures. It is observed that only a few clusters have a large population-size, indicating the regularity of the corresponding structural patterns in the time-series. One approach to predict the structure that may emerge from the current time-point is to partition the dynamic range of the time-series into k partitions for any positive integer k (>1), and then to identify from the time-series all possible occurrences of the structures originating from the selected partition along with their probability of occurrence. This, however, requires an extensive search of possible structures originating from a given partition, which usually is not affordable in real time during prediction. The search requirement during prediction can be alleviated by the adoption of a specialized automaton [30] with dynamic (user-specified) start and end states, called Dynamic Stochastic Automaton (DSA).

The DSA, which we would like to use to efficiently predict the structure emanating at time-point t, should have the following features. The states of the DSA represent (equal sized) partitions of the dynamic range of the time-series; the input symbols represent the structures extracted from the time-series and the transition functions describe transitions to new states/partitions from a given state due to the occurrence of an input symbol. Each transition is associated with two labels: a) the probability of occurrence of the transition and b) the expected duration of the transition. These attributes of a DSA are computed offline once for all during its construction, and need not be re-computed further during prediction. The DSA offers the benefit of determining the structure emanating from time point t using the knowledge acquired therein. The contribution of the paper in structure prediction is briefly summarized below.

1. The formulation of the structure prediction problem introduced here using segmentation, clustering, and automaton based prediction is novel.
2. The algorithms designed for segmentation, automaton construction and prediction using the automaton, are novel.
3. The hierarchical extension of the well-known data density dependent DBSCAN [31] algorithm for multi-resolution based multi-level clustering is also novel.

The selection of DBSCAN here is inspired by its independence on the number of clusters, which in most of the traditional clustering algorithms needs to be specified as a parameter.

The authors are associated with Jadavpur University, Kolkata, India.

The paper is divided into nine sections. Section II provides a brief review of related works on time-series segmentation and clustering. Section III provides an overview of the DBSCAN clustering algorithm. Section IV presents a novel algorithm for time-series segmentation. In section V, we extend the DBSCAN algorithm for hierarchical multi-resolution clustering. Section VI is concerned with the construction of a dynamic stochastic automaton (DSA). Section VII deals with prediction experiments on three well-known economic time-series: Taiwan Stock Exchange Index (TAIEX) [32], National Association of Securities Dealers Automated Quotations (NASDAQ) [33] and Dow Jones Industrial Average (DJIA) [34] for each year in the period 1990-1999, resulting in an average structure prediction accuracy of 76.03%. Section VIII discusses the performance analysis of the proposed time-series segmentation and clustering algorithms. Conclusions are listed is section IX.

II. Related Works

This section provides an overview of the existing research on segmentation and clustering issues in connection with time-series. Among the well-known time-series segmentation algorithms, the following need special mention. The first one, called the sliding window (SW) algorithm [19], determines the location of the next segment boundary by iterative widening of the current segment, until the approximation error (like Maximum Vertical Distance [22] or Root Mean Square Error [14]) of the segment exceeds a given value. The second algorithm, called the top-down algorithm [20] employs a divide and conquer rule for time-series segmentation in a recursive fashion. Each step of recursion here, attempts to optimally (i.e., with minimum approximation error) place a segment boundary within a given fragment of a time-series in order to divide it into two parts.

Unlike top down realization, which employs splitting of the time-series for optimal segmentation, the third algorithm employs a bottom-up [21] approach to merge segments to determine segment boundaries. Starting with each data point as a segment boundary, the algorithm merges segments greedily until the approximation error of the time-series exceeds a given threshold. The fourth one, called the Sliding Window and Bottom Up (SWAB) algorithm [18] combines the joint benefits of SW and bottom-up techniques to perform online segmentation of a time-series using a buffer. The contiguous blocks of time-series data are placed in the buffer sequentially for online segmentation using the bottom-up algorithm.

Besides the above, there remain a few other interesting algorithms on time-series segmentation. Liu *et al*. [22], for instance, extend the SW algorithm to reduce its computational overhead. They developed two extensions of SW, referred to as Feasible Space Window (FSW) and the Stepwise Feasible Space Window (SFSW) algorithms. Unlike the SW algorithm, where the approximation error for the entire segment needs to be calculated every time a new data point is added to the segment, both the FSW and SFSW algorithms use a modified segmentation criterion based on the MVD error metric. The segmentation criterion is designed in a way such that the error computation for a newly added point is carried out once, thus improving upon the computational complexity of the SW algorithm. Other noticeable efforts in time-series segmentation include the use of dynamic programming [23], fuzzy clustering [24], least square approximations [25] and evolutionary algorithms [26]. The segmentation algorithm that we develop in this paper provides a novel non-parametric online approach to natural segmentation of a time-series based on rising, falling or zero slopes of time-blocks in the series.

Clustering [29] is an interesting machine learning technique [35], which has found its way into several applications to process time-series data. Clustering is primarily used to detect structural similarity in unlabeled data sets based on various distance metrics like Euclidean distance, Minkowski distance [28] and others. A literature survey of clustering algorithms applied on time-series data can be found in [28]. It is indeed important to mention here, that in [36], clustering performed on overlapping time-series segments (windows) of fixed length, yields similar patterns/structures from any time-series [37]. However, clustering of semantically significant disjoint segments, as undertaken in the present work, results in insightful structures, which do differ for different time-series.

As has been mentioned before, for the purpose of identifying clusters of variable densities, we use an extended version of the data density based DBSCAN clustering algorithm. Among the existing extensions of DBSCAN which solve the problem of clustering data with variable densities, the DBSCAN-DLP [38] algorithm requires special mention. In DBSCAN-DLP, the entire data set is preprocessed and ordered into multiple layers based on their density. Each layer is later clustered using the DBSCAN algorithm. Although we employ a similar principle in our approach, the primary difference lies in the application of a greedy recursive technique where, instead of preprocessing the data set, we layer the data set by isolating the maximum density clusters and removing points with lower surrounding density as outliers. The main advantages of such recursive layering are that cluster centers are hierarchically arranged in decreasing order of data density. In addition data points of uniform, density, scattered spatially are clustered at the same level.

III. DBSCAN Clustering - An Overview

This section outlines the DBSCAN algorithm. DBSCAN is a density-based spatial clustering algorithm that groups points lying in a data-dense region into a cluster and marks points with non-dense surroundings as noise. It requires two input parameters: i) the radius ε describing the neighborhood of a point, and ii) a threshold m representing the minimum number of points to lie in the neighborhood of a (randomly or otherwise) selected point. The following terminologies are required to explain the rest of this section.

Definition 3.1: A point P in a given domain of points D is called a **core point**, if there exists a set of points $P' = \{P_1, P_2, P_3, ..., P_k\}$, such that the distance between $P_i \in P'$ and P is less than or equal to a pre-assigned small positive number ε (i.e., the point P_i lies in the ε**-neighborhood** of P) and the number of points k in the set P' is greater than or equal to an empirically selected threshold value m as given in (3.1-3.2).

$$\|P - P_i\| \le \varepsilon, \ \forall P_i \in P' \tag{3.1}$$

and

$$k \ge m \tag{3.2}$$

Any point $P_i \in P'$, lying in the ε**-neighborhood** of P is said to be **directly density-reachable** from the point P. **Data-density** of point P here, is defined as e/ε where e is the number of points

in the ε-neighborhood of P.

Definition 3.2: A point P is **density-reachable** from a point Q, if there exists a sequence of points $P_1, P_2, ..., P_n$ such that P is directly density-reachable from P_1, P_{i+1} is directly density-reachable from P_i, $\forall i \in \{1, 2, ..., n-1\}$ and Q is directly density-reachable from P_n. A point P_k is an **outlier**, if it is not density-reachable from any other point.

Definition 3.3: Two points P_i and P_j are **density-connected,** if there exists a point P such that both P_i and P_j are density-reachable from P.

We call a point P_i processed, when it is selected to examine the points in its $\varepsilon-$neighborhood. Any point not processed so far is called unprocessed. The algorithm, given in Pseudo Code A.1 of the Appendix [45], includes three main steps.

1. It begins by selecting an arbitrary unprocessed point P_i from the given data set D and checks if the point is a core point. If not, P_i is ignored as noise. However, if it is a core point, the algorithm identifies all points in the $\varepsilon-$neighbourhood of P_i and stores them in a set N. This set is used to store points which are yet to be processed for the creation of an individual cluster.
2. Next, the algorithm processes each point P_j in the set N by finding points lying in the $\varepsilon-$neighbourhood of P_j (storing them in a separate set M) and including them in N, (i.e., $N \leftarrow N \cup M$). The inclusion of new unprocessed points in N, with progressively increasing distance from the initial core point leads to the growth of the cluster boundary. When all points in N are processed, a cluster is formed and a new core point is searched.
3. The algorithm ends when unprocessed core points can no longer be detected.

It should be noted that a cluster always has the following two properties.

Property 1: Any two points P_i and P_j, lying in a cluster are mutually density-connected.

Property 2: If P_i is a point which is density-reachable from point P_j and $P_j \in C$, where C is a cluster, then $P_i \in C$.

IV. Slope-Sensitive Natural Segmentation

Segmentation algorithms designed to segment a time-series usually do not consider the expected characteristics of the time-series. Here we would like to design a segmentation algorithm that would take into account the essential characteristic moves of an economic time-series, such as bullish, bearish and sideways movements [39], respectively indicating rise, fall and near-zero slope in the expected segments. With this motive in mind, we propose a segmentation algorithm that keeps track of the slope in the local neighborhood of individual time-points and identifies the beginning and end points of a segment where slope change is significant with respect to those of neighboring time-points. Such segmentation has a similarity with the natural labeling of ridges in a mountain as rising/falling and plateau (zero slope).

The segmentation algorithm developed works in two phases. In the first phase, it attempts to label the lines joining consecutive pairs of data points as rise (R), fall (F) and zero-slope (E). In the second phase, it classifies windows covering a fixed number (intuitively chosen as five) of consecutive labels of R/F/E into one of the three types: rise, fall and zero-slope, depending on the highest frequency count of the labels in a given window. A set of one or more consecutive windows of same type (R/F/E) forms a segment. The algorithm is found to generate segments of a fixed set of geometry over a given time-series of varying length, and different sets of geometric patterns for different time-series. In addition, the algorithm is less sensitive to small fluctuations in the time-series.

A. Definitions

Definition 4.1: A **time-series** is a discrete sequence of samples of a measurable entity, such as temperature, atmospheric pressure, and population growth, taken over a finite interval of time. For an entity x, varying over time t in $[t_{min}, t_{max}]$, the time-series of length n can be expressed as a vector $\vec{X} = [x(kT)] = [x_k]$, where T is the sampling interval and k is an integer in $[1, n]$.

Definition 4.2: Given a time-series $\vec{x}$, let X_{max} and X_{min} respectively denote the maximum and minimum sample values of the series. Partitioning the time-series here refers to dividing the range $Z = [X_{min}, X_{max}]$ into k non-overlapping contiguous blocks $Z_1, Z_2, ..., Z_k$, called **partitions**, such that the following two conditions jointly hold:

$$Z_p \bigcap Z_q = \varnothing, \forall p, \forall q \in \{1, 2, ..., k\}, p \neq q \tag{4.1}$$

$$\bigcup_{i=1}^{k} Z_i = Z, \text{ for integer } i \tag{4.2}$$

Example 4.1: Let $\vec{X} = [2, 5, 4, 10, 3]$ be a time-series. According to definition 4.2, $X_{min} = 2$ and $X_{max} = 10$. We divide the range $[2, 10]$ into four partitions $Z_1 = [2, 4)$, $Z_2 = [4, 6)$, $Z_3 = [6, 8)$ and $Z_4 = [8, 10]$. It should be noted that both the upper and lower bounds of Z_3, the upper bound of Z_2 and the lower bound of Z_4 are not present in the time-series $\vec{X}$. □

Definition 4.3: Let $Z_p = [Z_p^-, Z_p^+]$ and $Z_q = [Z_q^-, Z_q^+]$ be two partitions in a time-series $\vec{X}$ and (x_i, x_{i+1}) be two successive data points in the series, such that $x_i \in Z_p$ and $x_{i+1} \in Z_q$. Then the **transition** of x_i to x_{i+1}, denoted by $Tr(x_i, x_{i+1})$, is assigned one of three possible linguistic labels: Rise (R), Fall (F) and Zero-slope (E) using (4.3).

$$Tr(x_i, x_{i+1}) = \begin{cases} R, if\ Z_p^+ \leq Z_q^- \\ F, if\ Z_p^- \geq Z_q^+ \\ E, if\ Z_p^+ = Z_q^+ \ and\ Z_p^- = Z_q^- \end{cases} \tag{4.3}$$

Example 4.2: Considering the time-series $\vec{x}$ given in Example 4.1, $x_1 = 2$ and $x_2 = 5$. Thus, $x_1 \in Z_p = [2, 4)$ and $x_2 \in Z_q = [4, 6)$. Since, $Z_q^- \geq Z_p^+$, by definition 4.3, $Tr(x_1, x_2) = R$. □

Definition 4.4: For a time-series $\vec{X} = [x_1, x_2, ..., x_n]$, a string of the form $\vec{S} = [S_1, S_2, ..., S_{n-1}]$ where $S_i = Tr(x_i, x_{i+1})$, for any integer $i \in [1, n-1]$, is called a **ternary string of transitions (T-SOT)**.

Example 4.3: For the time-series $\vec{x}$ of Example 4.1, $x_1 = 2,\ x_2 = 5,\ x_3 = 4,\ x_4 = 10$ and $x_5 = 3$. It can be verified using Z_1, Z_2, Z_3 and Z_4 that $Tr(x_1, x_2) = R$; $Tr(x_2, x_3) = E$; $Tr(x_3, x_4) = R$ and $Tr(x_4, x_5) = F$. Thus, the ternary string of transitions $\vec{S}$ for $\vec{X}$ is given by $\vec{S} = [R, E, R, F]$. □

Definition 4.5: Let $\vec{S} = [S_1, S_2, S_3, ..., S_{n-1}]$ be a T-SOT constructed from the time-series $\vec{X} = [x_1, x_2, ..., x_n]$. For each character S_i, a string of length $L \geq 3$ characters selected from $\vec{S}$ centered around S_i is referred to as **window** $\vec{W}_i$. For instance, a window $\vec{W}_i$, of length L characters is given in (4.4).

$$\vec{W}_i = [S_{i-\lfloor L/2 \rfloor},\ S_{i-\lfloor L/2 \rfloor+1},\ ...,\ S_i,\ ...,\ S_{i+\lfloor L/2 \rfloor-1},\ S_{i+\lfloor L/2 \rfloor}] \quad (4.4)$$

where $i \in \{\lfloor L/2 \rfloor + 1,\ \lfloor L/2 \rfloor + 2,\ ...,\ n - \lfloor L/2 \rfloor - 1\}$. Here onwards, we use a window of five (intuitively chosen) linguistic characters, of the form $\vec{W}_i = [S_{i-2},\ S_{i-1},\ S_i,\ S_{i+1},\ S_{i+2}]$, for the development of the SSNS algorithm.

Example 4.4: Let $\vec{S} = [R, F, E, E, R, R] = [S_1, S_2, S_3, S_4, S_5, S_6]$ be a T-SOT. Here, $n - 1 = 6$ and hence, $n - 3 = 4$. So, we have two valid windows of length, $L = 5$ characters: $\vec{W}_3 = [R, F, E, E, R]$ and $\vec{W}_4 = [F, E, E, R, R]$ for $i \in \{3,\ 4\}$. □

Note 4.1: It is important to note that windows containing a fixed length of r linguistic symbols refer to a time-series of length $r + 1$ data points. □

Definition 4.6: Let $\vec{X} = [x_1, x_2, ..., x_n]$ be a time-series and $\vec{S} = [S_1, S_2, S_3, ..., S_{n-1}]$ be the T-SOT constructed from $\vec{X}$. Let $\vec{W}_i = [S_{i-2}, S_{i-1}, S_i, S_{i+1}, S_{i+2}]$ for $3 \leq i \leq n-3$ be the i^{th} window of the time-series. Let $f_x(\vec{W}_i)$ denote the frequency count (number of occurrences) of the linguistic character $x \in \{R, F, E\}$ in $\vec{W}_i$. The window $\vec{W}_i$, is assigned a **label** $L(\vec{W}_i)$ based on the following policy.

$$L(\vec{W}_i) = R, \text{ if } f_R(\vec{W}_i) > f_x(\vec{W}_i),\ x \in \{F, E\} \quad (4.5)$$

$$= F, \text{ if } f_F(\vec{W}_i) > f_x(\vec{W}_i),\ x \in \{R, E\} \quad (4.6)$$

$$= E, \text{ if } f_E(\vec{W}_i) > f_x(\vec{W}_i),\ x \in \{R, F\} \quad (4.7)$$

If there exists three different linguistic characters: $p, q, r \in \{R, F, E\}$, such that $f_p(\vec{W}_i) = f_q(\vec{W}_i) > f_r(\vec{W}_i)$, then

$$L(\vec{W}_i) = p, \text{ if } L(\vec{W}_{i-1}) = p \quad (4.8)$$

$$= q, \text{ if } L(\vec{W}_{i-1}) = q \quad (4.9)$$

$$= p \text{ or } q \text{ arbitrarily otherwise.} \quad (4.10)$$

Note 4.2: In definition 4.6, we assign a label $x \in \{R, F, E\}$ to a window $\vec{W}_i$, if the frequency count of x is the highest in $\vec{W}_i$. Furthermore, if frequency count of any two of the three labels are equal and higher than the rest, and if any one of these labels has already been assigned to $\vec{W}_{i-1}$, then we assign the same label to $\vec{W}_i$. In case, the former condition holds but the latter condition fails, we can arbitrarily assign any one of the labels with the highest frequency count, to $\vec{W}_i$. □

Example 4.5: For a given time-series $\vec{X}$, let $\vec{S} = [R, F, R, E, R, R, E, E, F, E]$ be a T-SOT. The windows in the above series are $\vec{W}_3 = [R, F, R, E, R]$, $\vec{W}_4 = [F, R, E, R, R]$, $\vec{W}_5 = [R, E, R, R, E]$, $\vec{W}_6 = [E, R, R, E, E]$, $\vec{W}_7 = [R, R, E, E, F]$, and $\vec{W}_8 = [R, E, E, F, E]$. Since, $f_R(\vec{W}_3) = 3 > f_E(\vec{W}_3) = f_F(\vec{W}_3) = 1$, hence, $L(\vec{W}_3) = R$. Similarly, $L(\vec{W}_4) = R$ and $L(\vec{W}_5) = R$. In $\vec{W}_6$, $f_E(\vec{W}_6) = 3 > f_R(\vec{W}_6) = 2 > f_F(\vec{W}_6) = 0$. Hence, $L(\vec{W}_6) = E$. In $\vec{W}_7$, $f_E(\vec{W}_7) = 2 = f_R(\vec{W}_7) > f_F(\vec{W}_7) = 1$. Since, $L(\vec{W}_6) = E$, following definition 4.6, $L(\vec{W}_7) = E$. Again, as $f_E(\vec{W}_8) > f_R(\vec{W}_8) = f_F(\vec{W}_8)$, $L(\vec{W}_8) = E$ follows. □

Definition 4.7: A **structure** is a collection of one or more contiguous windows which have been assigned the same label.

Example 4.6: In example 4.5, $L(\vec{W}_3) = L(\vec{W}_4) = L(\vec{W}_5) = R$. So, $\vec{W}_3\vec{W}_4\vec{W}_5$ together forms a structure having the same label R, as those of the involved windows. The label of the i^{th} structure is denoted by $L(struct_i)$. □

Definition 4.8: For two consecutive structures $struct_i$ and $struct_{i+1}$, if $L(struct_i) \neq L(struct_{i+1})$, a segment boundary exists at the center point S_j of the last window $\vec{W}_j$ (say) of $struct_i$. Since, S_j denotes the transition between time-series data points x_j and x_{j+1}, the segment boundary is placed in between these two data points.

Example 4.7: In example 4.5 and 4.6, windows $\vec{W}_3$ to $\vec{W}_5$ in conjunction, form a structure having the label R. Again, $\vec{W}_6$ to $\vec{W}_8$ form a structure having the label E. Since, the labels of these two consecutive structures are not equal, a segment boundary appears at the center point S_5, of the last window $\vec{W}_5$ in the first structure. As S_5 denotes the transition between time-series data points x_5 and x_6, the segment boundary is placed in between these two data points. □

B. The SSNS Algorithm

The SSNS algorithm includes four steps. In the first step, we partition the time-series horizontally into k intervals of uniform width w as given in (4.11)

$$w = \frac{1}{n-1} \sum_{i=1}^{n-1} \left| x_{i+1} - x_i \right| \quad (4.11)$$

where x_i and x_{i+1} for $i = 1$ to $n-1$ are consecutive points in the time-series. In case a partition is empty, we merge it with its immediate lower partition. This ensures that no partition is empty and thus, helps in capturing small transitions in the time-series. It may be noted that the lower-most and upper-most partitions being at the boundaries of the dynamic range ($= k \times w$) of the time-series, includes at least one point. Next three steps of segmentation are transition labeling, window labeling and segment boundary determination. They directly follow from the definitions introduced above and are point-wise included in Pseudo code 1.

A trace of the SSNS segmentation algorithm for the time-series

[2,4,3,4,6,5,8,9,8,10,8,9,7,6,5] has been illustrated in Fig. 1. Fig. 1(b) depicts the partitioned time-series from which the T-SOT [*R*, *F*, *R*, *R*, *E*, *R*, *E*, *E*, *R*, *F*, *E*, *F*, *F*, *E*] is generated. The corresponding window labels and their relative frequencies are plotted in Fig. 1(d). The window label changes from *R* to *E* at the sixth data point and from *E* to *F* at the eleventh data point, thereby resulting in the segment boundaries as illustrated in Fig. 1(e).

C. Computational Complexity of SSNS Algorithm

The first step of the SSNS segmentation algorithm requires an iteration over each pair of consecutive data points in the time-series. Hence, the time complexity for partitioning is $O(n)$ and the space complexity is $O(n+z)$ where n is the number of data points in the time-series and z is the number of partitions.

Construction of the T-SOT involves iterating over each pair of consecutive data points in the time-series and identifying the partition to which each data point belongs. The array of partitions being ordered, a binary search is employed here. Hence, the partition to which a data point belongs can be computed in $O(\log z)$ time. This computation is done for n data points and hence, the total time complexity is $O(n \log z)$. Furthermore, the windowing, window labeling and segmentation steps together, iterate over each character in the T-SOT and place a marker at each segment boundary. The window being of fixed length, the decision to place a segment boundary can be taken in $O(1)$ time. As the T-SOT contains $n-1$ linguistic characters, this step has a time complexity of $O(n)$. Thus, the overall time complexity is $O(n)+O(n\log z)+O(n) \approx O(n\log z)$ and the space complexity is $O(n+z)$.

Pseudo code 1: SSNS Segmentation

Input: A time-series $\vec{X} = [x_1, x_2, ..., x_n]$, containing n real valued data points.

Output: A vector $\vec{I} = [i_1, i_2, ..., i_m]$, where i_j is the index or the time instant of the j^{th} segment boundary.

Step 1. Partitioning: Evaluate partition width, w following (4.11). Partition the entire range $[min(\vec{X}), max(\vec{X})]$, of the time-series $\vec{X}$ into k partitions, where $k = (max(\vec{X}) - min(\vec{X}))/w$ ($min(\vec{X})$ and $max(\vec{X})$ respectively denote minimum and maximum elements of the time-series). Merge partitions containing no data points with their immediate lower partition.

Step 2. Transition labeling: For each two consecutive data points x_i and x_{i+1}, $1 \le i \le (n-1)$, determine transition between x_i and x_{i+1} following (4.3). Store each computed transition in a T-SOT $\vec{S} = [S_1, S_2, ..., S_{n-1}]$ where $S_i = Tr(x_i, x_{i+1}), \forall i \in \{1, 2, ..., n-1\}$.

Step 3. Window labeling: For each group of five consecutive symbols in the T-SOT $\vec{S}$, obtain the windows $\vec{W}_i$ following (4.4) for $i = 3$ to $(n-3)$. Assign the label $L(\vec{W}_i)$ to the window $\vec{W}_i$, following (4.5-4.10).

Step 4. Segmentation: For two successive windows, $\vec{W}_{i-1}$ and $\vec{W}_i$, if $L(\vec{W}_{i-1}) \neq L(\vec{W}_i)$, place a segment boundary at the center of $\vec{W}_{i-1}$ (i.e., at symbol S_{i-1}) and correspondingly at the center of x_{i-1} and x_i.

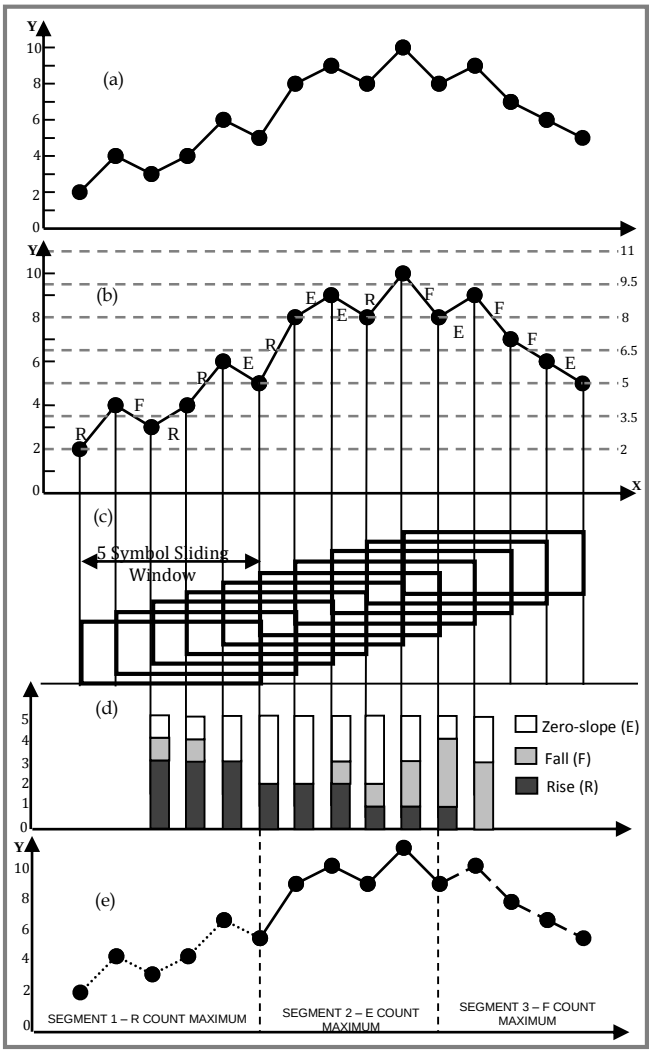


Fig. 1. Trace of the SSNS segmentation (a) time-series, (b) partitioning and T-SOT construction, (c) windowing, (d) freq. of window labels, (e) segmentation

V. MULTI-LEVEL CLUSTERING OF SEGMENTED TIME-BLOCKS

After a time-series is segmented into time-blocks, we attempt to group them into clusters based on their structural similarity. Since the segmented time-blocks are of non-uniform length, clustering is preceded by a pre-processing step, where the time-blocks are re-organized as a pattern of uniform length. The uniform length patterns having varied ranges need to be normalized by additional transformation (Z-score standardization [40]).

After pre-processing, we go for clustering the pre-processed time-blocks by using an extension of DBSCAN clustering, where the motivation is to cluster equally dense regions at the same level, and pass the less dense data points to the next level of clustering in a recursive manner until the data points are less than a prescribed threshold.

A. Pre-processing of Temporal Segments

Pre-processing is a two-step process. The first step includes transforming the variable length segmented time-blocks into vectors of uniform (here 10) length. We choose the value 10, as it is observed that the approximate average length of temporal segments generated by the SSNS algorithm is 10 data points. This transformation is undertaken by the following three sub-steps.

1. Join each pair of consecutive points in a segmented time-block

by a straight line, thereby generating a piecewise linear curve.
2. Divide the entire duration of the segment into ten equal parts and mark the corresponding time-points.
3. Determine the ordinates for the marked points on the time-axis of the curve obtained in step 1.

The second step of pre-processing is required to normalize the range of the time-blocks. Let there be l temporal segments in the time-series. Then a matrix M of $(10\times l)$ elements can be used to store the representation vectors of all the temporal segments, where the i^{th} column in M corresponds to the i^{th} temporal segment. In order to scale the temporal segments we use Z-score standardization as shown in (5.1)

$$M_{i,j} = \frac{M_{i,j} - mean(M, j)}{std(M, j)} \tag{5.1}$$

where $mean(M, j)$ and $std(M, j)$ are the mean and the standard deviation of the j^{th} column of matrix M respectively. Z-score standardization scales the temporal segments to have a zero mean and a unit variance.

B. Principles of Multi-level DBSCAN Clustering

The proposed multi-level DBSCAN clustering is a recursive extension of the DBSCAN algorithm, where at each level of recursion we cluster the data points of highest available density, and pass on the outliers (lower density data points) at that level for further clustering at the next level. This is illustrated in Fig. 2. To efficiently undertake the operations at a given level of recursion, we need two additional computations. First, we require evaluating the ε parameter at each level, and next we need to construct a point information table (PIT) at that level.

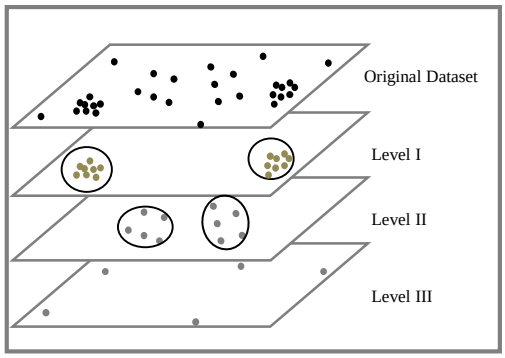


Fig. 2. Multilevel density-based clustering: Points of higher density are clustered at higher levels, transferring lower density points for clustering at lower levels

The PIT stores the number of points within an ε-radius of all possible points, if the number of neighborhood points around that point exceeds a given threshold. It can be organized by different approaches. Given a set of l points of the form $P = \{p_1, p_2, ..., p_l\}$, one simple approach to realize this (PIT) is to allocate an array of pointers, where the i^{th} element points to a linear linked list containing the count N_i and indices of the points lying in the ε-neighborhood of the i^{th} point p_i. The array of pointers (Fig. 3) only point to the linked list of $(l-m)$ points, that satisfy the necessary criterion that the number of data points in the ε-neighborhood of a point exceeds a pre-defined threshold. In other words, out of l points only $(l-m)$ points satisfy the necessary criterion, and remaining m points do not satisfy it and thus are treated as outliers at that level.

The primary purpose of the PIT is to compute the outliers in each density based stratum. A point p_i is defined as an outlier if the number of points N_i in the ε-neighborhood of p_i is less than the average of the number of neighborhood points of all l points in the given data set. This is given in (5.2).

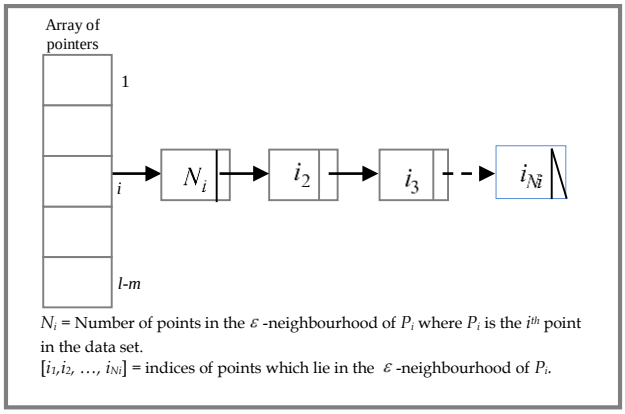


Fig. 3. Structure of the Point Information Table

$$\left.\begin{aligned} outlier(p_i) = true,\ if\ N_i < (\frac{1}{l})\sum_{j=1}^{l} N_j \\ = false, \text{otherwise.} \end{aligned}\right\} \tag{5.2}$$

Furthermore, the PIT is scanned to identify the point having highest neighbourhood density as the seed for generating a new cluster. This ensures that cluster centers are hierarchically arranged in decreasing order of density.

Determining the ε parameter for each layer: Given a set of l 10-dimensional points representing Z-score standardized time-blocks, we construct an $l\times l$ distance matrix D, whose $(i, j)^{th}$ element $d_{i,j}$ represents the Euclidean distance between the points p_i and p_j. Now, to compute the ε parameter for the first layer, we sort the distances $d_{i,j}$ in ascending order of their magnitude and take the mean of the first k distances in the sorted list, where k is an empirically chosen parameter, typically taken as 10% of the members in the list. The process of ε parameter computation is repeated in the same manner for each layer of data points.

The proposed multilevel DBSCAN algorithm includes the following three steps in each layer of recursion.

1. From a given set of data points in a layer, the algorithm evaluates the ε-parameter by the procedure introduced above.
2. The PIT for the given level is constructed to eliminate outliers at that level and also to select the points with highest neighbourhood density as core points for clustering at that level using the DBSCAN algorithm.
3. The outliers at a given level are defined as the input data set for clustering at the next level.

The following property indicates that no additional outliers can be obtained after clustering at a given level.

Property 5.1: *The clustering at each layer of similar density data points does not produce any outliers other than those dropped from the PIT constructed for that layer.*

Proof. We prove the property by the method of contradiction. Let

us assume that after clustering at a given layer, there exist additional outliers excluding those dropped from the PIT for that layer. Outliers being points with less than threshold number of ε -neighborhood points, are dropped from the PIT. Thus, the points considered for clustering at the layer do not include any point containing less than the threshold number of ε-neighborhood points. Hence, all points in a given layer are clustered. This contradicts the assumption that there exist additional outliers at a given layer of clustering, and hence its contradiction is true. □

The proposed algorithm is given in Pseudo Code A.2 of the Appendix [45]. The modified-DBSCAN algorithm differs from traditional DBSCAN with respect to the method for core point selection. In traditional DBSCAN, core points are selected randomly, whereas in the modified-DBSCAN the point with the highest neighborhood density in the PIT is selected as the core point for generation of an individual cluster. The modification in core point selection is advantageous for layered clustering as it generates cluster centers in decreasing order of density without requiring additional time-complexity. Also, in traditional DBSCAN, we generally do not compute cluster centers (as seen in Pseudo Code A.1 of the Appendix [45]). However, as we require the representative patterns (structures) for each cluster in our application, we compute the 10-dimensional cluster centers as the mean of the corresponding data points in their clusters.

The proposed clustering algorithm has been successfully applied to TAIEX close price time-series for the period 1990-2000 with resulting 9 cluster centers of diverse geometry as illustrated in Fig. 4.

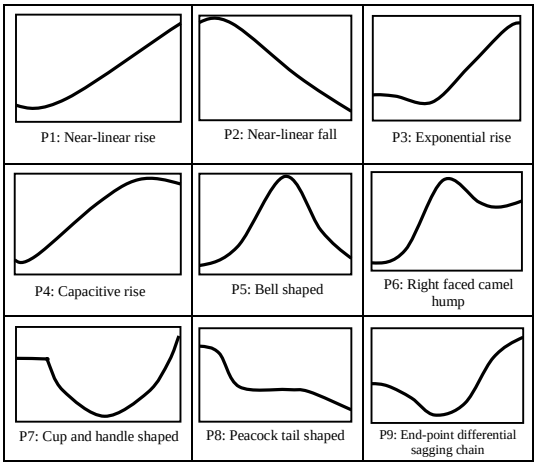


Fig. 4. Primitive patterns (cluster centers) extracted from the TAIEX time-series with their proposed names based on similarity with real world objects

C. Computational Complexity of Extended DBSCAN

Let there be l temporal segments obtained from the time-series. In the initialization step of the proposed multi-layered DBSCAN clustering algorithm, a distance matrix is computed from the given l points in $O(l^2)$ time. In each density based stratum, the first step involves tuning the value of ε corresponding to the maximum density clusters. This is done by taking the mean of the least k distances from the distance matrix, where k is an empirically chosen constant. Hence, the time complexity involved is $O(l^2)$.

The second step constructs the PIT with l entries where the i^{th} entry stores the indices of points lying in the ε-neighborhood of the i^{th} point. This also takes $O(l^2)$ time. The third step involves removal of outliers requiring an iteration over each entry in the PIT, taking constant time to decide if a point is an outlier. Hence, this step can be done in $O(l)$ time. The fourth step, i.e., DBSCAN clustering of the remaining points normally has a time complexity of $O(l^2)$. With the current modification of DBSCAN, searching for a core point takes $O(l)$ time. However, the PIT can be used to identify the $\varepsilon-$neighborhood points of the core point in constant time. Thus, the total time complexity of modified-DBSCAN is also $O(l^2)$. It is important to note that the number of density based strata is significantly less compared to the number of points l, to be clustered and hence, can be considered as a constant. Thus the overall time-complexity for the proposed clustering algorithm is $O(l^2)+O(l^2)+O(l)+O(l^2)\approx O(l^2)$. Since, the distance matrix takes $O(l^2)$ space, the space complexity of the algorithm is also $O(l^2)$.

VI. Representation of Structural Transitions in a Time-Series by Dynamic Stochastic Automaton

Detection of possible structures that may emanate from a given time-series data point requires an extensive search of the structures that originate from the same partition containing the data point. Each time we want to predict such a structure we need to spend the above search cost. In order to ensure that the search of structures is carried out once, for all predictions, we need to store the possible structures emanating from all the partitions in a specialized data structure, here realized with the help of an automaton. The states in the automaton represent partitions of the time-series and the input symbols represent structures that cause a transition from one partition to some other partition. We also label the transitions with two parameters, the probability of occurrence of the transition and its expected duration.

Given the start and target states as a query, the automaton outlined above helps in determining the most probable structure to reach the target state from the start state along with its probability of occurrence and expected duration. Due to the start and target states being user-defined and dynamic, we name the automaton Dynamic Stochastic Automaton (DSA).

Definition 6.1: A **dynamic stochastic automaton (DSA)** is defined by a 7-tuple given by

$$F=(Q,I,Q_G,V,\delta,\delta_t,\delta_G) \tag{6.1}$$

where, $Q=$ Non-empty finite set of states (partitions)

$I=$ Non-empty finite set of input symbols, representative of primitive patterns (cluster centers) obtained in section 5

$Q_G=$ Set of next states (called a group) for a given state and a given input symbol: $Q_G: Q\times I\rightarrow P(Q)$, where $P(Q)$ is the power set of Q. It should be noted that Q_G has two representations. It can both be represented as a set of states or as a mapping from the Cartesian product of Q and I to the power set of Q, i.e., $P(Q)$.

$V=[0,1]$ is the valuation space

$\delta=$ Probabilistic transition function $\delta: Q\times I\times Q\rightarrow V$

$\delta_t=$ Temporal transition function $\delta: Q\times I\times Q\rightarrow \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers representing time required for the transition.

δ_G = Probabilistic transition function for groups: $\delta_G : Q \times I \times Q_G \rightarrow V$ where δ_G generates the probability of transition from a starting state to a group of next states on the occurrence of a given input symbol.

For a given pair of states $p, q \in Q$, the transition from p to q following the input symbol $a \in I$ is associated with a label of the form: $a / \delta(p,a,q)$, $\delta_t(p,a,q)$. The proposed DSA has the following properties:

Property 6.1: *It is required that for any given state q, the sum of the probability of transitions to all possible groups $G \in Q_G(q,x)$, $\forall x$ following all possible input symbols x is unity as given below*:

$$\sum_{x \in I} \sum_{G \in Q_G(q,x)} \delta_G(q,x,G) = 1 \tag{6.2}$$

Property 6.2: *The probability of transition $\delta(q,x,q')$ from a state q to a state q' following the input symbol x is the product of the transition probability $\delta_G(q,x,Q_G(q,x))$ from q to the group $Q_G(q,x)$ following the input symbol x and the conditional transition probability $\delta(q,x,q' \mid Q_G(q,x))$ from q to q' following x, given that the transition to the group $Q_G(q,x)$ has already occurred. This is formally given below*:

$$\delta(q,x,q') = \delta_G(q,x,Q_G(q,x)) \times \delta(q,x,q' \mid Q_G(q,x)) \tag{6.3}$$

Note 6.1: It should be noted that the proposed DSA differs from traditional stochastic automata [30] by the following counts. First, unlike traditional stochastic automata, the DSA has no fixed starting states and terminal states, as they are user-defined and thus dynamically chosen, justifying the name, dynamic stochastic automaton. Second, there is an additional term δ_t describing the time required for the transition to take place on occurrence of an input symbol $x \in I$ to a given state $s \in Q$. Third, we have introduced the concept of a group of next states $Q_G(q,x)$ from a given starting state $q \in Q$, following a given input symbol $x \in I$ in the DSA. The state transition probability from the starting state q to a target state q' following the input symbol x is hence, modified as given in Property 6.2. □

A. Construction of Dynamic Stochastic Automaton (DSA)

Construction of an automaton requires partitioning the dynamic range of the time-series into fewer partitions, rather than large number of partitions as undertaken during the segmentation phase. Apparently this looks a little tricky. However, the re-partitioning scheme has a fundamental basis to ensure that the partitions, when transformed to states in the DSA, will be utilized in most cases if they are relatively few. In case the partitions obtained during segmentation are used here, most of the states in the automaton remain non-utilized, as the vertical span of the temporal patterns obtained in clustering is large enough to cover several partitions obtained in the segmentation phase. After careful experiments, it is observed that the temporal segments obtained by clustering can cover up to 3 to 4 partitions if the dynamic range is equally partitioned into 7 to 10 partitions.

The probability of transition $\delta_G(s_i,a,Q_G(s_i,a))$ from state $s_i \in Q$ to the group of states $Q_G(s_i,a)$ following input symbol $a \in I$ is obtained as

$$\delta_G(s_i,a,Q_G(s_i,a)) = \frac{\sum_{s \in Q_G(s_i,a)} f(s_i,a,s)}{\sum_{s \in Q} \sum_{x \in I} f(s_i,x,s)} \tag{6.4}$$

where $f(s_i,a,s)$ denotes the frequency count or the number of transitions from partition s_i to partition s due to the occurrence of a temporal pattern belonging to cluster a (that acts as an input symbol).

Similarly, the conditional probability $\delta(s_i,a,s_j \mid Q_G(s_i,a))$ that a transition will occur from the state $s_i \in Q$ to the state $s_j \in Q_G(s_i,a)$ following the input symbol a, given that a transition to the group $Q_G(s_i,a)$ has already occurred, is obtained as

$$\delta(s_i,a,s_j \mid Q_G(s_i,a)) = \frac{f(s_i,a,s_j)}{\sum_{s \in Q_G(s_i,a)} f(s_i,a,s)} \tag{6.5}$$

Finally, the computation of the probability of a transition $\delta(s_i,a,s_j)$ from state s_i to state s_j due to input symbol a where $s_i, s_j \in Q$ and $a \in I$, is obtained, following Property 6.2, as

$$\delta(s_i,a,s_j) = \delta_G(s_i,a,Q_G(s_i,a)) \times \delta(s_i,a,s_j \mid Q_G(s_i,a)) \tag{6.6 (a)}$$

$$= \frac{\sum_{s \in Q_G(s_i,a)} f(s_i,a,s)}{\sum_{s \in Q} \sum_{x \in I} f(s_i,x,s)} \times \frac{f(s_i,a,s_j)}{\sum_{s \in Q_G(s_i,a)} f(s_i,a,s)} \tag{6.6(b)}$$

$$= \frac{f(s_i,a,s_j)}{\sum_{s \in Q} \sum_{x \in I} f(s_i,x,s)} \tag{6.6 (c)}$$

The computation of $\delta_t(s_i,a,s_j)$ is obtained here by measuring the duration of all possible transitions in the time-series with starting state s_i and terminating state s_j due to the occurrence of a temporal pattern belonging to cluster a. Let the possible temporal segments falling in cluster a that appear at partition s_i for transition to partition s_j be $a_1, a_2, ..., a_n$. Then, $\delta_t(s_i,a,s_j)$ is obtained as

$$\delta_t(s_i,a,s_j) = \frac{\sum_{k=1}^{n} \tau(s_i,a_k,s_j)}{n} \tag{6.7}$$

where $\tau(s_i,a_k,s_j)$ represents the temporal length of the time-segment a_k starting at partition s_i and ending at partition s_j. The following illustrative example explains the notations used in equations (6.1) to (6.6) and discusses the computation of transition probabilities in a DSA with 6 states (corresponding to 6 partitions in the time-series). The number of states chosen in this example is arbitrary and completely for illustrative purposes.

Example 6.1: Let there be a DSA as shown in Fig. 5. We demonstrate the calculation of transition probabilities from state S1 to all the other states in the automaton. The frequency counts (i.e., number of occurrences) of transitions from state S1 to the other states are given in Table I. The element in Table I corresponding to the transition from S1 to some state Sj is represented as a set of ordered pairs of the form (*x*,*y*), where *x* represents the input symbol and *y* represents the corresponding frequency count $f(1, x, j)$, of transitions from S1 to Sj following the input symbol *x*.

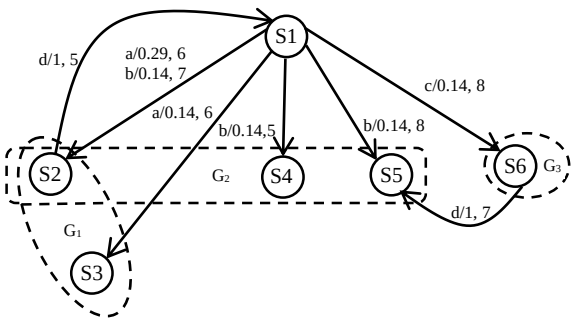


Fig. 5. A DSA to illustrate its parameters, the labels in each arc are of the form *"a/x,y"* where *a* is the input symbol, *x* is the probability of occurrence of the transition and *y* is the expected duration of the transition. The groups G_1, G_2 and G_3 corresponding to state S1 are shown using closed dotted curves.

TABLE I
FREQUENCY COUNTS OF TRANSITIONS FROM STATE S1 TO OTHER STATES IN THE DSA OF FIG. 5

To state / From state	S1	S2	S3	S4	S5	S6
S1	–	(a,2) (b,1)	(a,1)	(b,1)	(b,1)	(c,1)

Here, $Q = \{\text{S1, S2, S3, S4, S5, S6}\}$, $I = \{a,b,c,d\}$,

$Q_G(\text{S1},a) = \{\text{S2, S3}\} = G_1$,

$Q_G(\text{S1},b) = \{\text{S2, S4, S5}\} = G_2$,

$Q_G(\text{S1},c) = \{\text{S6}\} = G_3$.

Having determined the groups of next states G_1, G_2 and G_3, corresponding to state S1, we can compute the transition probability $\delta(\text{S1},a,\text{S2})$ following the simplified formula presented in equation (6.6 (c)) as follows:

$$\delta_G(\text{S1},a,\text{S2}) = \frac{f(1,a,2)}{f(1,a,2)+f(1,a,3)+f(1,b,2)+f(1,b,4)+f(1,b,5)+f(1,c,6)}$$

$$= \frac{2}{2+1+1+1+1+1} = \frac{2}{7} \approx 0.29.$$

Similarly, the other transition probabilities from state S1 are computed as $\delta(\text{S1},a,\text{S3}) \approx 0.14$, $\delta(\text{S1},b,\text{S2}) \approx 0.14$, $\delta(\text{S1},b,\text{S4}) \approx 0.14$, $\delta(\text{S1},b,\text{S5}) \approx 0.14$ and $\delta(\text{S1},c,\text{S6}) \approx 0.14$. Alternatively, the same transition probabilities can be computed following equations (6.4), (6.5) and (6.6 (a)), though at a greater computational cost. The detailed calculations corresponding to this approach of computing transition probabilities are given in the Appendix [45].

We also provide the values of the temporal transition function δ_t obtained from Fig. 5 as

$\delta_t(\text{S1},a,\text{S2}) = 6$, $\delta_t(\text{S1},a,\text{S3}) = 6$, $\delta_t(\text{S1},b,\text{S2}) = 7$,

$\delta_t(\text{S1},b,\text{S4}) = 5$, $\delta_t(\text{S1},b,\text{S5}) = 8$ and $\delta_t(\text{S1},c,\text{S6}) = 8$. □

The approach of constructing a DSA is summarized in Pseudo code 2.

Pseudo code 2: Computation of DSA

Input: A sequence $< a_1, a_2, ..., a_l >$ of l time-segments,
k, the number of states in the DSA.
Output: A DSA corresponding to the given temporal segments.

```
BEGIN
  Partition the time-series into k partitions z_i and construct a
  state s_i for each partition z_i for i = 1 to k ;
  FOR each temporal segment a_k, k = 1 to l DO
  BEGIN
  IF s_i is the starting state, s_j is the ending state of a_k and a
   is the corresponding structure THEN
      Insert a directed arc from state s_i to state s_j in the
      automaton and label it with δ(s_i, a, s_j) and δ_t(s_i, a, s_j) in
      order following (6.6(c)) and (6.7);
  END IF;
  END FOR;
END.
```

B. Forecasting Using the Dynamic Stochastic Automaton

Forecasting the most probable structure (MPS) at a given starting state to reach a target state is an important problem in economic time-series prediction. Such forecasting requires three inputs: i) a completely labeled DSA, ii) the current state obtained from the day of prediction, and iii) the target state. Occasionally, the users can provide a time-limit within which they expect to reach the target state. In case there exist multiple structures to reach a given target state, then the expected durations of the predicted structures are checked against a threshold before recommendation of the sequence to users.

In order to realize the above approach, the following steps are carried out.

1. For a given starting state s_1 and a given target state s_2, determine all possible structures p such that, $\delta(s_1, p, s_2)$ is maximized.
2. If multiple structures (MPS) are found to exist with equal probability of occurrence, report the MPS whose duration is closest to the duration given by the user in the query.

Example 8.1: Let there be a DSA as shown in Fig. 6.

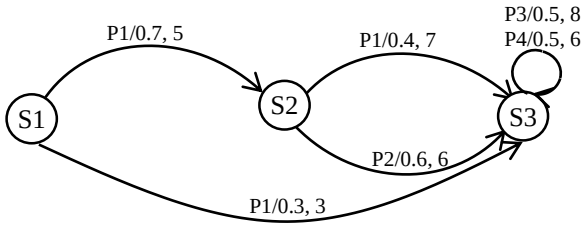


Fig. 6. Example of a time-labeled dynamic stochastic automaton

TABLE II
MOST PROBABLE STRUCTURE (MPS) AND ITS PROBABILITY OF OCCURRENCE AND EXPECTED DURATION FOR EACH PAIR OF STATES IN THE DSA OF FIG. 6

Target state / Start state	S1			S2			S3		
	MPS	δ	δ_t	MPS	δ	δ_t	MPS	δ	δ_t
S1	-	-	-	P1	0.7	5	P1	0.3	3
S2	-	-	-	-	-	-	P2	0.6	6
S3	-	-	-	-	-	-	P3/P4	0.5	8/6

In Table II, we show the most probable structure (MPS) along with its corresponding probability of occurrence δ and expected duration δ_t, for all possible pairs of states in the DSA shown in

Fig. 6.

C. Computational Complexity of DSA Construction

Construction of a DSA from a pre-segmented and labeled time-series requires a single iteration over all the labeled segments. For each segment, the algorithm requires its starting state (partition), its ending state (partition) and its corresponding structure. Processing of each segment takes constant or $O(1)$ time. Naturally, the construction of the DSA requires $O(l)$ time where l is the number of segments. A trivial data structure for storing the DSA in memory is a matrix that holds the transitions for each possible pair of states following all possible input symbols. Such a memory representation takes $O(k^2m)$ space, where k is the number of states and m is the number of structures (or input symbols). Clearly, this is the worst case space complexity. However, more compact ways of representing the DSA can be easily thought of. For instance, storing the information for each transition (i.e., its start state, target state, input symbol, probability of occurrence and expected duration) in a list can save a lot of memory space.

It is important to note here that the overall time-complexity for all the three steps: segmentation, clustering and DSA construction is $O(l^2)$ corresponding to the clustering step. The complexities of the other two steps being very small compared to clustering, can be safely ignored.

The time and space complexities for segmentation, clustering and DSA construction algorithms have been summarized in Table III. Also, the CPU runtime in milliseconds is mentioned, for carrying out segmentation of a time-series of 5474 data points as well as clustering of the obtained segments and construction of the corresponding DSA. The algorithms are executed on an Intel Core i5 processor with a CPU speed of 2.30 GHz using MATLAB as the programming environment.

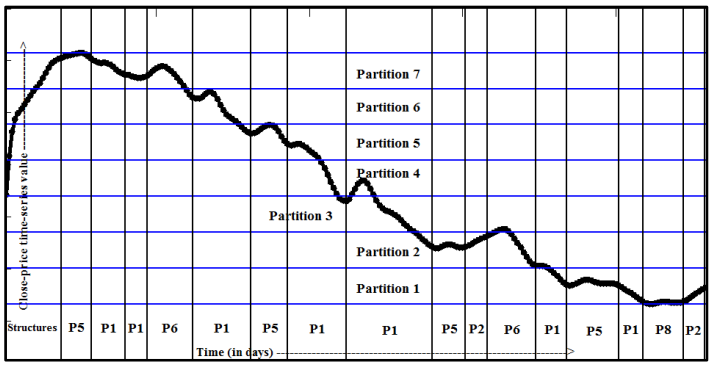
TABLE III
COMPUTATIONAL AND RUNTIME COMPLEXITIES OF SEGMENTATION, CLUSTERING AND DSA CONSTRUCTION ALGORITHMS

Algorithm	Time complexity	Space complexity	CPU Runtime (milliseconds)
SSNS Segmentation	$O(n \log z)$	$O(n+z)$	1872.0
Multi-layered DBSCAN Clustering	$O(l^2)$	$O(l^2)$	3868.8
DSA Construction	$O(l)$	$O(k^2m)$	847.4

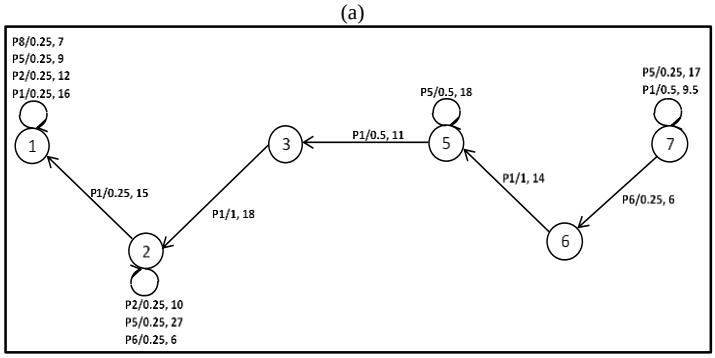
VII. PREDICTION EXPERIMENTS AND RESULTS

This section presents an experiment to examine the success of the proposed model in forecasting three well-known economic time-series: i) TAIEX [32], ii) NASDAQ [33] and iii) DJIA [34] for each year in the period 1990-1999. Some statistical details regarding the above mentioned three time-series are given in Section II of the Appendix [45]. The time-series for every year (in 1990-1999) is divided into two disjoint parts. The proposed model is trained for the period 1st January to 31st October (10 months) and is tested on the period 1st November to 31st December (2 months). The experiments involve the following steps for each time-series (TAIEX, NASDAQ and DJIA):

Step 1. Extraction of structures from a time-series: The time-series for the entire period 1990-1999, is first segmented and



(a)



(b)

Fig. 7. Construction of a DSA from a pre-segmented and labeled time-series: (a) the TAIEX 1990 close-price time-series for the period January to October is segmented and labeled into its corresponding structures, (b) the DSA constructed from the pre-segmented labeled time-series.

the segments are then clustered to identify the structures in the series. The only required parameter to be set in this step is the threshold number of data points T (refer to Pseudo Code A.2 of the Appendix [45]) below which clustering will no longer continue. The value of T is set as 10% of the total number of data points for clustering (i.e., total number of temporal segments). The recurrent structures extracted from TAIEX, NASDAQ and DJIA, in the form of 10-dimensional vectors have been plotted and are illustrated in Fig. A.5, Fig. A.6 and Fig. A.7 of the Appendix [45].

Step 2. Construction of DSA: For each year in the period 1990 to 1999, a DSA is constructed from the training span, i.e., from January to October. First, the training phase time-series is partitioned and segmented. Next, all the obtained segments are labelled with their respective structures. Such a labelled time-series is used to construct the DSA. In our experiments, we set the number of states in the DSA as seven (thereby setting k = 7 in Pseudo Code 2). The number of states should ideally be between 7 and 10. A higher number leads to most of the states remaining unused. Hence, we partition the training period time-series into seven equi-spaced partitions, namely, Extremely Low (1), Very Low (2), Low (3), Medium (4), High (5), Very High (6) and Extremely High (7). Construction of the DSA from a pre-segmented and labelled training period TAIEX series for the year 1990 is shown in Fig. 7. The DSAs generated for all the years corresponding to the three time-series are given in Fig. A.8, Fig. A.9 and Fig. A.10 of the Appendix [45].

Step 3. Prediction of the Most Probable Structure (MPS): During the prediction phase, we consider the time-series for the months November and December of each year. Given the start and target states as input queries, the DSA can predict the MPS along with its corresponding probability and expected duration. Hence, for the test phase series, we predict the MPS on the days which

correspond to segment boundaries. The start state is the partition corresponding to the current segment boundary and the target state is the partition for the next segment boundary. Any data point lying in between two segment boundaries falls within the predicted MPS and hence, the prediction for all such data points yields the same MPS as that obtained from the last encountered segment boundary. The predicted structures in the test-period for each year in the span 1990-1999 for all three time-series, TAIEX, NASDAQ and DJIA are summarized in Tables A.IV, A.V and A.VI respectively of the Appendix [45].

Once we identify the MPS between a pair of given states, we can extrapolate the sequence of data points lying on the predicted structure. This can be done in two steps: i) de-normalize the cluster center corresponding to the MPS such that the range of the de-normalized structure lies within the start and target prices, ii) sample the de-normalized cluster center at a daily interval to obtain the complete sequence of data points between the start and the target states. This extrapolation technique enables us to predict the daily time-series data points lying on a structure. It also helps us in building a comparison framework with other existing prediction models based on the well-known RMSE metric. Time-series plots of the predicted data points along with the corresponding actual data points in the test phase are illustrated in Fig. A.11, A.12 and A.13 of the Appendix [45] for TAIEX, NASDAQ and DJIA respectively. We have also reported the prediction RMSE values in Tables A.IV, A.V and A.VI of the Appendix [45]. In this regard, it should be noted that for a very small number of test-phase segments we cannot make any predictions. For instance, the first segment in the test-phase time-series for TAIEX in the year 1990, starts from partition 1 and ends at partition 2. As seen from Fig. 7(b), there is no arc from state 1 to state 2 in the DSA corresponding to TAIEX, 1990. Hence, we cannot possibly make any valid prediction in this case. In such scenarios the predicted structures are marked as "N.A." in Tables A.IV, A.V and A.VI of the Appendix [45].

Step 4. Evaluation: In order to evaluate the performance of the proposed model for structure prediction, we use three well-known metrics: Precision (*P*), Recall (*R*) and Accuracy (*A*). For calculating these metrics we first construct a confusion matrix **C**, where the element $\mathbf{C}_{i,j}$ corresponding to the i^{th} row and j^{th} column is equal to the number of times that structure *j*, is predicted as structure *i*. The confusion matrices obtained for TAIEX, NASDAQ and DJIA are presented in Table A.VII, A.IX, and A.XI of the Appendix [45] respectively. Given a confusion matrix **C**, the precision *P* and recall *R* for structure *i* are computed as follows:

$$P = \frac{\mathbf{C}_{i,i}}{\sum_{\forall j} \mathbf{C}_{i,j}} \quad \text{and} \quad R = \frac{\mathbf{C}_{i,i}}{\sum_{\forall j} \mathbf{C}_{j,i}}. \tag{7.1}$$

The overall precision and recall are computed as the mean of the precision and recall measures of individual structures. The accuracy *A* from a given confusion matrix is computed as follows:

$$A = \frac{\sum_{\forall i} \mathbf{C}_{i,i}}{\sum_{\forall i}\sum_{\forall j} \mathbf{C}_{i,j}}. \tag{7.2}$$

The evaluation measures obtained from the confusion matrices for TAIEX, NASDAQ and DJIA are given in Tables A.VIII, A.X and A.XII of the Appendix [45] respectively. In Table IV, we present the overall evaluation measures obtained from the three time-series. We have also performed a comparison of prediction performances of our model with other econometric time-series models [9-14] on the TAIEX data set, based on the RMSE metric. The RMSE values obtained from the above mentioned prediction models for each year in the period 1990 to 1999 are given in Table A.XIII of the Appendix [45] and are plotted in Fig. 8.

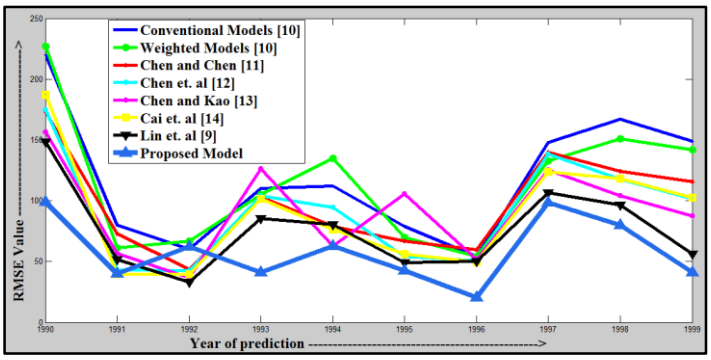


Fig. 8. RMSE errors obtained from different time-series models by prediction on the TAIEX time-series for the years 1990 – 1990.

TABLE IV
EVALUATION METRICS FOR ASSESSING THE PERFORMANCE OF THE PROPOSED MODEL ON TAIEX, NASDAQ AND DJIA TIME-SERIES

Evaluation measure / Time-Series	Precision	Recall	Accuracy	RMSE
TAIEX	60.4%	76.3%	67.86%	58.7914
NASDAQ	92.3%	83.6%	82.93%	4.6037
DJIA	82.6%	86%	77.3%	21.5961

It is evident from Table IV, that the precision, recall and accuracy of structure prediction, averaged over the three time-series TAIEX, NASDAQ and DJIA are very encouraging with values of 78.43% precision, 81.97% recall and 76.03% overall accuracy. Furthermore, it can also be observed from Fig. 8 that the prediction RMSE values on TAIEX for the years 1990-1999 are quite low compared to other existing models in the literature. With an average RMSE of approximately 58.79 (on the TAIEX), the proposed method outperforms other existing models by a relatively large margin.

Thus, the experimental results indicate that the proposed model might aid in real world stock market prediction. Given the current stock price and a target price that we want to achieve, the model can be used to predict the most probable structure along with the approximate time to reach the target value. Furthermore, the extrapolation technique discussed in Step 3 above, can be applied to obtain the sequence of time-series data points lying on the predicted structure.

VIII. PERFORMANCE ANALYSIS

In this section, we compare the performance of the proposed time-series segmentation and clustering algorithms with traditional algorithms available in the literature. For segmentation, the SSNS algorithm is compared with the Top-down [20], Bottom-up [21], Sliding Window (SW) [19] and Feasible Space Window (FSW) [22] algorithms. In order to compare the performance of different segmentation algorithms, we define a metric called match-score. Two segment boundaries from different segmentation algorithms are said to match if they are placed at the same time-instant. The total number of matched segment boundaries provides the match-

score of the outputs of the two algorithms. We compute the match-score of outputs obtained from different segmentation algorithms with respect to the output obtained from a hand-crafted ground-truth segmentation of the same time-series. For this analysis, the above mentioned segmentation algorithms are executed on a sample time-series of 150 data points taken from the TAIEX close-price series. The results are summarized in Table V, where, for each algorithm, we report the percentage of total segments matched with the hand-crafted ground-truth segments. The outputs of the above mentioned algorithms could not be shown here due to lack of space and are given in [41] for convenience. It is observed that with a match-score percentage of 76.5%, the proposed SSNS algorithm clearly outperforms all other competitive algorithms.

In order to analyse the performance of the proposed extension of DBSCAN, we compare it with the traditional k-means [42], fuzzy c-means [43] and DBSCAN [31] algorithms, using two performance indices, the *sum of squared intra-cluster distance* (SSID) and *sum of inter-cluster distances* (SICD) [44]. Let there be m clusters with cluster centers $(Z_1, Z_2, ..., Z_m)$. Then the SSID index is computed as follows,

$$SSID = \sum_{i=1}^{m} \sum_{X \in S_i} \left\| X - Z_i \right\| \tag{8.1}$$

where, S_i is the set of points in the i^{th} cluster and $\left\| X - Z_i \right\|$ is the Euclidean distance of the point X from its corresponding cluster center Z_i. Clearly, a low value of the SSID index indicates a good performance of the clustering algorithm.

The SICD index is computed between cluster centers as follows,

$$SICD = \sum_{i \neq j} \left\| Z_i - Z_j \right\| \tag{8.2}$$

where, $i, j \in \{1, 2, .., m\}$. The SICD index provides an estimation of inter-cluster separation. Naturally, a higher value of the SICD index indicates a well-spaced separation between individual clusters and thus a better performance of the clustering algorithm. The algorithms are executed on the TAIEX close-price time-series for the period 1990-2000 with the following input parameters:

K-means [42]: The desired number of clusters is set as 9, as we obtain 9 clusters (see Fig. 4) from the data set by applying the proposed multilevel DBSCAN. The algorithm is executed for a maximum of 1000 iterations.

Fuzzy c-means [43]: The desired number of clusters is set as 9 for the same reason as mentioned above for K-means algorithm. The algorithm is executed for a maximum of 1000 iterations.

DBSCAN [31]: The eps (ε) parameter is set based on the highest density clusters, following the steps mentioned in Section V.B. Naturally, the ε value is small. A higher value of ε results in all the data points being grouped into a single cluster. The value of the minimum number of points m in the $\varepsilon-$neighborhood of a core point is intuitively set as 5.

Multilevel DBSCAN: The parameters are same as that mentioned in Step 1 of the Experiments section (Section VII).

The obtained SSID and SICD measures for each clustering algorithm are summarized in Table VI. With the lowest SSID value of 10.6215 and the highest SICD value of 1293.2, the proposed multilevel DBSCAN outperforms the other traditional clustering algorithms.

In order to compare the relative performance of the proposed extension of DBSCAN algorithm with the original DBSCAN, we execute both the algorithms for clustering the temporal segments obtained from the TAIEX series for the period 1990-2000. The experimental results reveal that the number of clusters detected by the DBSCAN algorithm is two, representing the bullish (near-linear rise) and bearish (near-linear fall) temporal patterns as shown in Fig. A.14 of the Appendix [45]. On the other hand, the proposed multi-layered extension of DBSCAN is able to identify nine temporal patterns (as shown in Fig. 4). Also the cluster centers produced are automatically arranged in decreasing order of density, where the density of the i^{th} cluster is defined is follows,

$$Density = \frac{R_i}{D_i} \tag{9.3}$$

In (9.3), R_i is the number of data points in the i^{th} cluster and D_i is the maximum intra-cluster distance between any two points in the i^{th} cluster. The densities of the clusters obtained on the said data set are summarized in Table A.XIV of the Appendix [45]. The results obtained from the above experiment provide conclusive proof of the advantage of the proposed extension of DBSCAN. It is able to segregate clusters of variable densities and successfully detect temporal patterns of lower density in the data set. Since DBSCAN is not able to adapt its ε parameter for clusters of variable densities, it is able to identify only the two clusters of maximum density (bullish and bearish patterns). Also, for relatively larger values of ε, DBSCAN groups all the data points into a single cluster.

TABLE V
COMPARISON OF TIME-SERIES SEGMENTATION ALGORITHMS BY MATCHING OF GENERATED SEGMENTS WITH HAND-CRAFTED GROUND-TRUTH SEGMENTS

Algorithm	Number of matched segment boundaries (m)	Number of segment boundaries obtained (S)	Percentage of segment boundaries matched $(m / S) \times 100$
Top-down [20]	7	21	33.3
Bottom-up [21]	7	19	36.8
SW [19]	11	26	42.3
FSW [22]	9	18	50.0
SSNS	**13**	**17**	**76.5**

TABLE VI
COMPARISON OF EXISTING CLUSTERING ALGORITHMS WITH THE PROPOSED TECHNIQUE BASED ON THE SSID AND SICD METRICS

Algorithm / Performance index	K-means [42]	Fuzzy c-means [43]	DBSCAN [31]	**Multilevel DBSCAN**
SSID	15.8917	11.2476	12.5371	**10.6215**
SICD	1186.9	1147.4	1222.2	**1293.2**

IX. CONCLUSIONS

The merit of the paper lies in the overall formulation of knowledge acquisition from a time-series along with the merits of individual algorithms used in the knowledge acquisition process. The segmentation algorithm is used to isolate time-blocks of similar structural characteristics, i.e., positive, negative or near-zero slopes from the rest. The clustering algorithm proposed here is an extension of the well-known DBSCAN [31] algorithm with an aim

to hierarchically cluster temporal segments based on their data-density. DBSCAN is used as the base clustering algorithm for its advantage of not requiring the number of clusters as an input parameter. The merit of the extension lies in multi-level hierarchical clustering, where each level is used to cluster points of maximum data-density available in that level. The proposed extension is advantageous as it successfully captures clusters of lower data-density as well.

The merit of the dynamic stochastic automaton lies in its inherent structure for knowledge representation which includes the information about both the probability of occurrence as well as the expected duration of each transition. These parameters have been successfully applied to predict the most probable structure (MPS) between two given states within a finite time-limit.

Extensive experiments undertaken on three well-known economic time-series: TAIEX, NASDAQ and DJIA, indicate promising results with an average precision of 78.43%, an average recall of 81.97% and an average accuracy 76.03% for structure prediction. The proposed model can thus be successfully used for trading and investment in stock market.

REFERENCES

[1] O. Barnea, A. R. Solow, and L. Stone, "On Fitting a Model to a Population Time Series with Missing Values," *Israel Journal of Ecology and Evolution*, vol. 52, pp. 1-10, 2006.

[2] S.M. Chen, and J.R. Hwang, "Temperature Prediction Using Fuzzy Time Series," *IEEE Trans. Syst., Man, Cybern., Part-B*, vol. 30, no. 2, pp. 263-275, Apr. 2000.

[3] C.L. Wu, and K.W. Chau, "Prediction of Rainfall Time Series Using Modular Soft Computing Methods," *Elsevier, Engineering Applications of Artificial Intelligence*, vol. 26, pp. 997-1007, 2013.

[4] A. Jalil, and M. Idrees, "Modeling The Impact Of Education On The Economic Growth: Evidence From Aggregated And Disaggregated Time Series Data Of Pakistan," *Elsevier, Econ. Model*, vol. 31, pp. 383-388, 2013.

[5] M. Small and C.K. Tse, "Detecting Determinism in Time Series: The Method of Surrogate Data," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 5, pp. 663-672, May 2003.

[6] B. Abraham and J. Ledolter, *Statistical Methods for Forecasting*, John Wiley and Sons, Inc., Hoboken, NJ, USA, 2008.

[7] D.T. Mirikitani and N. Nikolaev, "Recursive Bayesian Recurrent Neural Networks for Time-Series Modeling," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 262-274, Feb. 2010.

[8] J.T. Connor, R.D. Martin, and L.E. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 240-254, Mar. 1994.

[9] X. Lin, Z. Yang and Y. Song, "Short-term Stock Price Prediction Based on Echo State Networks," *Elsevier, Expert Systems with Applications*, vol. 36, no. 3, pp. 7313-7317, Apr. 2009.

[10] H.K. Yu, "Weighted Fuzzy Time Series Models for TAIEX Forecasting," *Elsevier, Physica A*, vol. 349, no. 3-4, pp. 609-624, Apr. 2005.

[11] S.M. Chen, C.D. Chen, "TAIEX Forecasting Based on Fuzzy Time Series and Fuzzy Variation Groups," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 1, pp. 1-12, Jan. 2011.

[12] S.M. Chen, H.P. Chu, and T.W. Sheu, "TAIEX Forecasting Using Fuzzy Time Series and Automatically Generated Weights of Multiple Factors," *IEEE Trans. Syst. Man Cybernet. Part A Syst. Hum.*, vol. 42, no. 6, pp. 1485–1495, Apr. 2012.

[13] S.M. Chen, and P.Y. Kao, "TAIEX Forecasting Based on Fuzzy Time Series, Particle Swarm Optimization Techniques and Support Vector Machines," *Elsevier, Inform. Sci.* vol. 247, pp. 62-71, Oct. 2013.

[14] Q. Cai, D. Zhang, W. Zheng and S.C.H Leung, "A New Fuzzy Time Series Forecasting Model Combined with Ant Colony Optimization and Auto-regression," *Elservier, Knowledge-Based Systems*, vol. 74, pp. 61-68, Jan. 2015.

[15] M. Han, J. Xi, S. Xu and F.L. Yin, "Prediction of Chaotic Time Series Based on the Recurrent Predictor Neural Network," *IEEE Trans. on Signal Processing*, vol. 52, no. 12, pp. 3409-3416, Dec. 2004.

[16] Jan G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *Elsevier, International Journal of Forecasting*, vol. 22, pp. 443-473, 2006.

[17] P. Esling and C. Agon, "Multiobjective Time Series Matching for Audio Classification and Retrieval," *IEEE Trans. on Audio Speech and Language Processing*, vol. 21, no. 10, pp. 2057-2072, May 2013.

[18] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," *Data Mining in Time Series Databases,* second ed. World Scientific, 2003.

[19] U. Appel and A.V. Brandt, "Adaptive Sequential Segmentation of Piecewise Stationary Time Series," *Information Science*, vol. 29, no. 1, pp. 27-56, 1983.

[20] H. Shatkay and S.B. Zdonik, "Approximate Queries and Representations for Large Data Sequences," Technical Report CS-95-03, Dept. of Computer Science, Brown Univ., 1995.

[21] E. Keogh and P. Smyth, "A Probabilistic Approach to Fast Pattern Matching in Time Series Databases," *Proc. ACM SIGKDD '97*, pp. 20-24, 1997.

[22] X. Liu, Z. Lin, and H. Wang, "Novel Online Methods for Time Series Segmentation," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 12, pp. 1616-1626, Dec. 2008.

[23] G.F. Bryant and S.R. Duncan, "A Solution to the Segmentation Problem Based on Dynamic Programming," *Proc. Third IEEE Conf. Conf. Control Applications (CCA '94)*, pp. 1391-1396, 1994.

[24] J. Abonyi, B. Feil, S. Nemeth, and P. Arva, "Modified Gath-Geva clustering for fuzzy segmentation of multivariate time-series," *Elsevier, Fuzzy Sets and Systems*, vol. 149, no. 1, pp. 39-56, Jan. 2005.

[25] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations," *IEEE Trans. Pattern Analysis, Machine Intelligence*, vol. 32, no. 12, pp. 2232-2245, Dec. 2010.

[26] F.L. Chung, T.C. Fu, V. Ng, and R.W.P. Luk, "An Evolutionary Approach to Pattern-Based Time Series Segmentation," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 5, pp. 471-489, Oct. 2004.

[27] Y. Zhao and S. Zhang, "Generalized Dimension Reduction Framework for Recent-Biased Time Series Analysis," *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 2, pp. 231-244, Feb. 2006.

[28] T. Warren Liao, "Clustering of time series data – a survey," *Elsevier, Pattern Recognition*, vol. 38, no. 11, pp. 1857-1874, Nov. 2005.

[29] R. Xu and D. Wunsch II, "Survey of Clustering Algorithms," *IEEE Trans. Neural Networks*, vol. 16, no. 3, May 2005.

[30] E.R. Dougherty, and C.R. Giardina, *Mathematical Methods for Artificial Intelligence and Autonomous Systems*, Prentice-Hall international editions publisher, the University of Michigan, USA, 1988.

[31] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. KDD-96*, pp. 226-231, 1996.

[32] TAIEX [Online]. Available: http://www.twse.com.tw/en/products/indices/tsec/taiex.php

[33] NASDAQ [Online]. Available: http://www.nasdaq.com

[34] DJIA [Online]. Available: https://www.google.com/finance?cid=983582

[35] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, New York, USA, 1997.

[36] G. Das, K.I. Lin, H. Mannila, G. Renganathan and P. Smyth, "Rule Discovery From Time Series," in *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, pp. 16-22, 1998.

[37] E. Keogh and J. Lin, "Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research," *Knowledge and Information Systems*, vol. 8, no. 2, pp. 154-177, Aug. 2005.

[38] Z. Xiong, R. Chen, Y. Zhang, and X. Zhang, "Multi-density DBSCAN Algorithm Based on Density Levels Partitioning," *Journal of Information & Computational Science*, vol. 9, no. 10, pp. 2739-2749, 2012.

[39] R. D. Edwards, J. Magee, and W.H.C. Bassetti, "The Dow Theory," in Technical Analysis of Stock Trends, 8th ed., 2000 N.W. Corporate Blvd., Boca Raton, Florida: CRC Press LLC, 2001, pp. 21-22.

[40] J. Lundberg, "Lifting The Crown-Citation Z-score", *Elsevier, Journal of Informetrics*, vol. 1, pp. 145-154, 2007.

[41] http://computationalintelligence.net/tnnls/segmentation_performance.pdf

[42] E.W. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.

[43] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York and London, 1987.

[44] J.C. Bezdek and N.R. Pal, "Some New Indexes of Cluster Validity," *IEEE Trans. Syst. Man Cybern., Part-B*, vol. 28, no. 3, pp. 301–315, 1998.

[45] http://computationalintelligence.net/tnnls/appendix.pdf

Appendix to the paper
Learning Regularity in an Economic Time-Series for Structure Prediction

D. Bhattacharya, J. Mukhoti, A. Konar, *Senior Member, IEEE* and A. Saha

This Appendix includes additional information about the paper, containing algorithms, illustrative examples, detailed experimental results and other required information, which could not be included in the main manuscript for space limitation. It includes a total of 10 sections. The Appendix is also uploaded on the authors' website for convenience of the readers. The web-link is given in [45] of the main paper.

I. Traditional DBSCAN Algorithm

In this section, we present the pseudo code of the well-known DBSCAN algorithm in Pseudo Code A.1. The pseudo code could not be included in the main manuscript due to lack of space.

Pseudo code A.1: DBSCAN clustering

Input: A data set D, of q unlabeled data points, $\{P_1, P_2, ..., P_q\}$ in k-dimensional space, and user-defined parameters ε and m, representing respectively, the neighbourhood radius of a point and minimum number of points around a core point.
Output: A q-dimensional vector $\vec{C} = [c_1, c_2, ..., c_q]$, where c_i denotes the cluster number (representative of cluster identity) of the point P_i $\forall i \in \{1, 2, ..., q\}$.

```
BEGIN
Initialize cluster_count ← 1, and declare all points in D as unprocessed; Initialize set N ← ∅;
FOR each unprocessed point P_i, in D do
BEGIN
  Mark P_i as processed;
  FOR each point p in D, if p is directly density reachable from P_i, save p in N;
  IF (|N| < m) THEN ignore P_i as noise;
  ELSE do
  BEGIN
    Mark P_i as a core point and assign cluster_count as cluster number c_i to P_i;
    FOR each unprocessed point P_j in N, do
    BEGIN
      Mark P_j as processed; Initialize set M ← ∅;
      FOR each point s in D, if s is directly density reachable from P_j, save s in M;
      IF (|M| ≥ m) THEN N ← N ∪ M;
      N ← N − {P_j};
      Assign cluster_count as cluster number c_j to P_j;
    END FOR;
  END IF;
Increment cluster_count by one; Reinitialize N ← ∅;
END FOR;
END.
```

II. EXTENDED MULTILEVEL DBSCAN ALGORITHM

In this section, we present the pseudo code of the proposed extended multilevel DBSCAN algorithm in Pseudo Code A.2. The pseudo code could not be included in the main manuscript due to lack of space.

Pseudo code A.2: Multilevel DBSCAN Clustering

Input: An array of l 10-dimensional points, $P=[p_1, p_2, ..., p_l]$ and an empirically chosen threshold value T.

Output: A vector $\vec{C}=[c_1, c_2, ..., c_m]$ of m cluster centers of 10-dimensions each.

```
BEGIN MAIN
  D ← compute_distance_matrix(P);
  C ← multilevel_DBSCAN(P,D,T);
END MAIN
Procedure multilevel_DBSCAN(P,D,T):
BEGIN
  q ← num_points(P); /*compute point count in array P */
  IF ( q < T ) THEN EXIT; /*Stopping criterion of recursion */
  ε ← compute_epsilon(D); /*compute the current ε */
  Initialize PIT as an array of q pointers all set to NULL;
  Initialize P' as an empty array; /* P' will store outliers */
  FOR each point p_i in P do /*Construction of PIT */
  BEGIN
    Store the index j of all points p_j lying in the ε - neighborhood of p_i in the i^th row of the PIT;
  END FOR;
  FOR each point p_i in P do /*Removal of outliers*/
  BEGIN
    /* outlier(p_i) is computed following equation 5.2 (in main manuscript) */
    IF outlier(p_i) = true, mark p_i as outlier and store it in P';
    ELSE mark p_i as non-outlier;
  END FOR;
  Cluster the non-outlier points using modified-DBSCAN and store corresponding cluster centers (mean of the data
   points in a cluster) in C⃗;
  Call multilevel_DBSCAN(P',D,T);
END.
```

III. Computation of Transition Probabilities in a Dynamic Stochastic Automaton

In this section, we illustrate the calculation of state transition probabilities in the DSA shown in Fig. 5 of the paper. For convenience, we have given the same figure here as Fig. A.1. The state transition probabilities are computed here following Property 6.2 in the paper, i.e., the transition probability $\delta(\text{S1}, a, \text{S2})$ from state S1 to state S2 following the input symbol *a*, is equal to product of the transition probability $\delta_G(\text{S1}, a, G_1)$ from state S1 to the group G_1 (the group of next states obtained when the input symbol *a* occurs at state S1) and the conditional probability $\delta(\text{S1}, a, \text{S2} \mid G_1)$ that a transition will occur from state S1 to state S2 following symbol *a*, given that a transition to some state in group G_1 has already occurred.

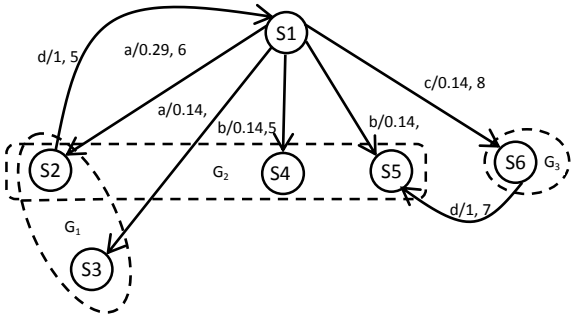


Fig. A.1. A DSA to illustrate its parameters, the labels in each arc are of the form *"a/x,y"* where *a* is the input symbol, *x* is the probability of occurrence of the transition and *y* is the expected duration of the transition. The groups G_1, G_2 and G_3 corresponding to state S1 are shown using closed dotted curves.

Here, $Q = \{\text{S1, S2, S3, S4, S5, S6}\}$, $I = \{a, b, c, d\}$,

$Q_G(\text{S1}, a) = \{\text{S2, S3}\} = G_1$,

$Q_G(\text{S1}, b) = \{\text{S2, S4, S5}\} = G_2$,

$Q_G(\text{S1}, c) = \{\text{S6}\} = G_3$.

Having determined the groups of next states G_1, G_2 and G_3, corresponding to state S1, we can compute the transition probability $\delta_G(\text{S1}, a, G_1)$ following equation (6.4) as follows:

$$\delta_G(\text{S1}, a, G_1) = \frac{f(1,a,2) + f(1,a,3)}{(f(1,a,2) + f(1,a,3)) + (f(1,b,2) + f(1,b,4) + f(1,b,5)) + (f(1,c,6))} = \frac{2+1}{(2+1)+(1+1+1)+(1)} = \frac{3}{7}.$$

Similarly, we compute $\delta_G(\text{S1}, b, G_2) = \frac{3}{7}$ and $\delta_G(\text{S1}, c, G_3) = \frac{1}{7}$. Next, we calculate the conditional probability of transition $\delta(\text{S1}, a, \text{S2} \mid G_1)$ from state S1 to state S2, due to the input symbol *a*, given that the transition to group G_1 has already occurred. This is computed following equation (6.5) as follows:

$$\delta(\text{S1}, a, \text{S2} \mid G_1) = \frac{f(1,a,2)}{f(1,a,2) + f(1,a,3)} = \frac{2}{2+1} = \frac{2}{3}.$$

In a similar fashion, we compute $\delta(\text{S1}, a, \text{S3} \mid G_1) = \frac{1}{3}$, $\delta(\text{S1}, b, \text{S2} \mid G_2) = \frac{1}{3}$, $\delta(\text{S1}, b, \text{S4} \mid G_2) = \frac{1}{3}$, $\delta(\text{S1}, b, \text{S5} \mid G_2) = \frac{1}{3}$ and $\delta(\text{S1}, c, \text{S6} \mid G_3) = 1$. Finally, we can compute the probability of transition $\delta(\text{S1}, a, \text{S2})$ from state S1 to state S2 due to input symbol *a* (from equation (6.6 (a)) as:

$$\delta(\text{S1}, a, \text{S2}) = \delta_G(\text{S1}, a, G_1) \times \delta(\text{S1}, a, \text{S2} \mid G_1) = \frac{3}{7} \times \frac{2}{3} = \frac{2}{7} \approx 0.29.$$

Similarly, the other transition probabilities from state S1 are computed as $\delta(\text{S1}, a, \text{S3}) \approx 0.14$, $\delta(\text{S1}, b, \text{S2}) \approx 0.14$, $\delta(\text{S1}, b, \text{S4}) \approx 0.14$, $\delta(\text{S1}, b, \text{S5}) \approx 0.14$ and $\delta(\text{S1}, c, \text{S6}) \approx 0.14$. It should be noted that the same transition probabilities are obtained following the mathematically simplified formula in equation (6.6 (c)) albeit at a much lesser computational cost.

IV. Details of Experimental Time-Series: TAIEX, NASDAQ and DJIA for the Period 1990-1999

In this section, we present some statistical details of the data sets on which we carry out our experiments. The experiments are carried out on the daily close-price values of three well-known economic time-series: TAIEX, NASDAQ and DJIA for each year in the period 1990-1999. For every year, we train our model from the months January to October (10 months) and test the model on the months November and December (2 months). Tables A.I, A.II and A.III contain the observation frequencies (i.e., number of time-series data

points) in each of the seven partitions of the dynamic range of a time-series along with other statistical measures like maximum, minimum, mean, median and standard deviation values of the series for each year in the period 1990-1999 for the economic time-series TAIEX, NASDAQ and DJIA respectively. In addition, we have also presented the time-series plots for the entire period 1990-1999 for TAIEX, NASDAQ and DJIA in Fig. A.2, Fig. A.3 and Fig. A.4 respectively.

TABLE A.I
OBSERVATION FREQUENCIES AND STATISTICAL MEASURES FOR TAIEX ECONOMIC CLOSE-PRICE TIME-SERIES FOR THE YEARS 1990-1999

Measures / Year	Observation Frequencies							Statistical Descriptive Measures				
	Partition 1	**Partition 2**	**Partition 3**	**Partition 4**	**Partition 5**	**Partition 6**	**Partition 7**	**Maximum**	**Minimum**	**Mean**	**Median**	**Standard Deviation**
1990	73	67	25	12	26	30	48	12495.34	2560.47	6749.13	5402.64	3246.24
1991	6	20	86	68	35	37	34	6305.22	3316.26	4928.83	4815.92	653.11
1992	36	67	24	44	57	41	15	5391.63	3327.67	4271.63	4370.30	546.04
1993	20	66	113	69	11	9	2	6070.56	3135.50	4214.78	4136.46	468.33
1994	30	20	65	36	45	49	41	7183.75	5194.63	6252.96	6257.77	524.72
1995	55	75	32	42	11	65	6	7051.49	4503.37	5543.75	5423.17	687.94
1996	53	15	4	27	77	75	37	6982.81	4690.22	6003.72	6175.87	685.65
1997	26	36	69	71	28	25	31	10116.84	6820.35	8410.56	8267.58	802.48
1998	19	61	45	49	31	28	38	9277.09	6251.38	7727.68	7653.51	810.12
1999	9	23	18	26	89	76	25	8608.91	5474.79	7426.69	7589.34	695.87

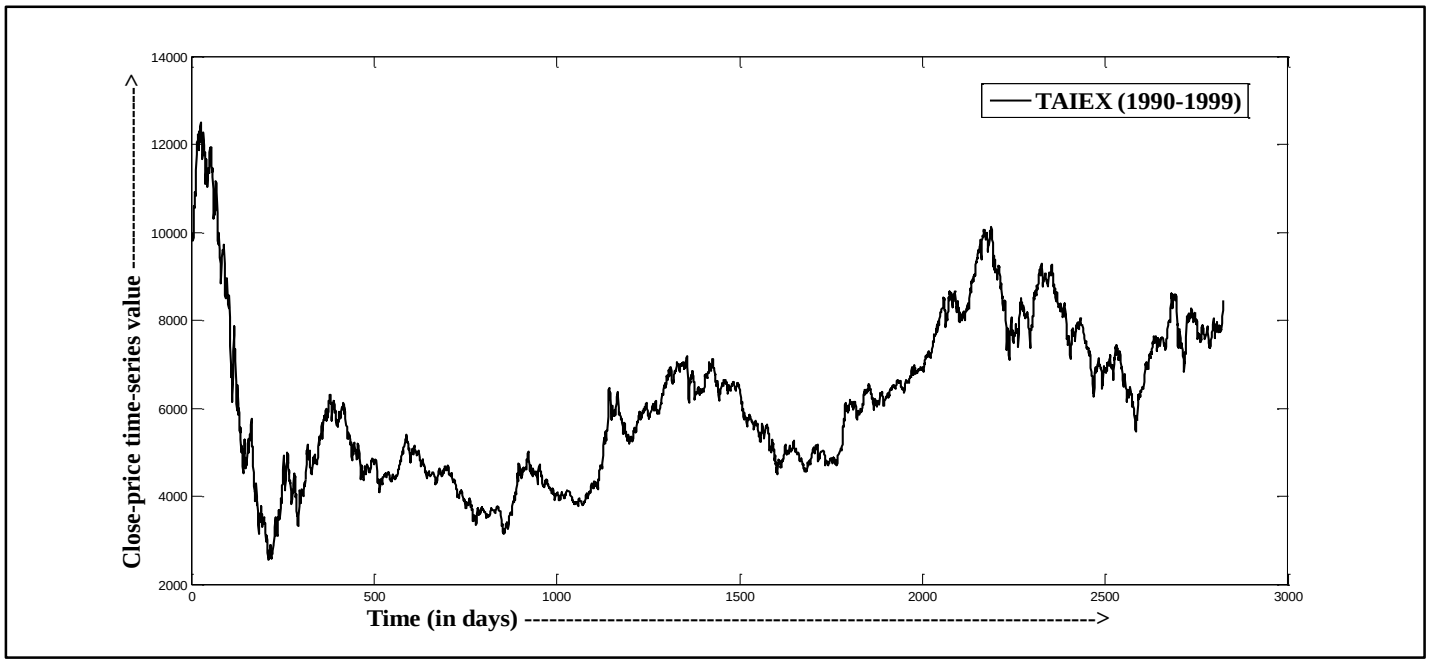


Fig. A.2. Time-series plot of TAIEX economic close-price time-series for the period 1990-1999

TABLE A.II
OBSERVATION FREQUENCIES AND STATISTICAL MEASURES FOR NASDAQ ECONOMIC CLOSE-PRICE TIME-SERIES FOR THE YEARS 1990-1999

Measures / Year	Observation Frequencies							Statistical Descriptive Measures				
	Partition 1	**Partition 2**	**Partition 3**	**Partition 4**	**Partition 5**	**Partition 6**	**Partition 7**	**Maximum**	**Minimum**	**Mean**	**Median**	**Standard Deviation**
1990	26	29	37	6	39	66	50	469.60	325.40	409.18	425.20	43.04
1991	16	7	18	44	94	67	7	586.34	355.80	491.69	497.55	45.90
1992	31	81	40	31	43	14	14	676.95	547.84	599.26	588.48	31.55
1993	18	44	75	19	27	40	30	787.42	645.87	715.16	704.09	37.21
1994	14	40	47	36	51	28	36	803.93	693.79	751.65	750.14	27.83
1995	33	46	30	14	11	44	74	1069.79	743.58	925.19	939.48	109.04
1996	14	23	61	35	40	56	25	1316.27	988.57	1164.95	1162.08	81.01
1997	32	49	36	19	40	51	26	1745.85	1201.00	1469.48	1455.61	154.51
1998	8	33	45	91	47	19	9	2192.69	1419.12	1794.91	1797.89	147.65
1999	69	89	53	10	12	11	8	4069.31	2208.05	2728.15	2602.43	405.14

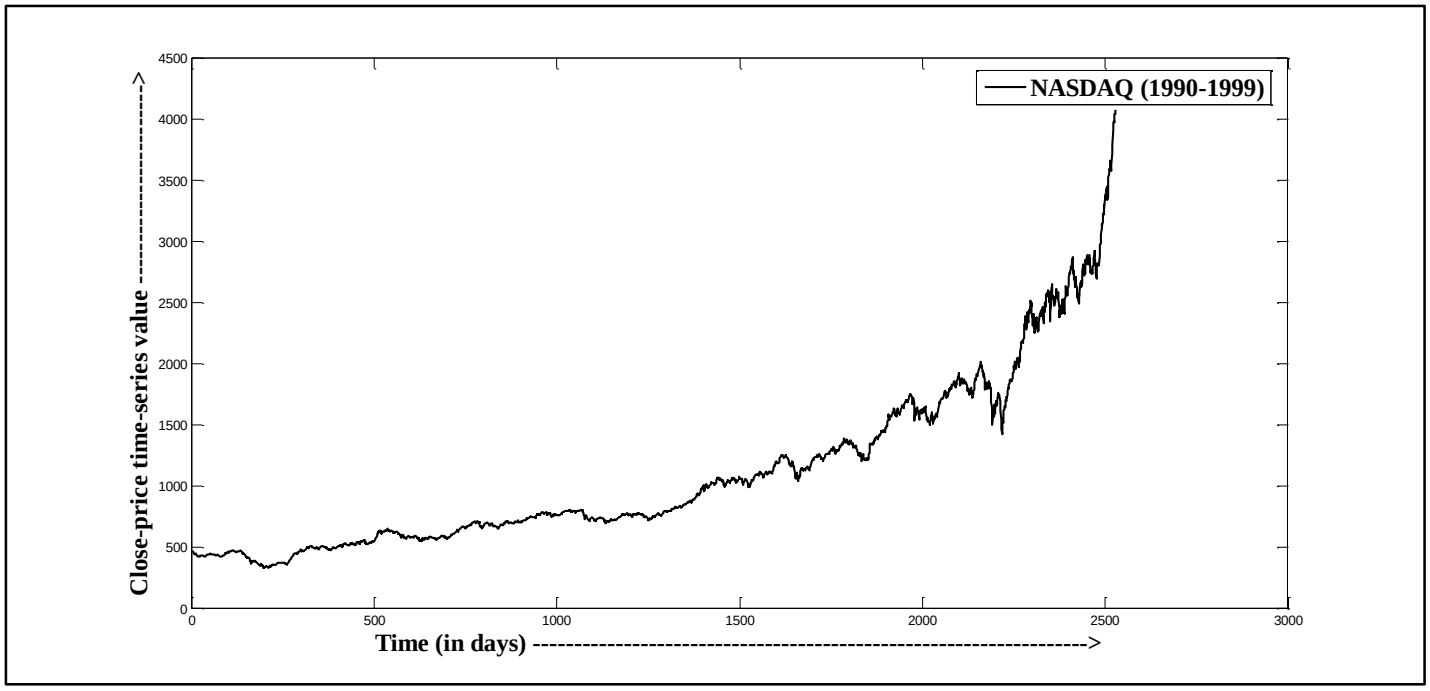


Fig. A.3. Time-series plot of NASDAQ economic close-price time-series for the period 1990-1999

TABLE A.III
OBSERVATION FREQUENCIES AND STATISTICAL MEASURES FOR DJIA ECONOMIC CLOSE-PRICE TIME-SERIES FOR THE YEARS 1990-1999

Measures / Year	Observation Frequencies							Statistical Descriptive Measures				
	Partition 1	**Partition 2**	**Partition 3**	**Partition 4**	**Partition 5**	**Partition 6**	**Partition 7**	**Maximum**	**Minimum**	**Mean**	**Median**	**Standard Deviation**
1990	18	32	65	50	31	36	21	2999.75	2365.10	2679.45	2659.13	148.73
1991	9	11	3	13	109	101	7	3168.83	2470.30	2929.04	2949.50	125.01
1992	7	26	55	66	47	30	23	3413.21	3136.58	3284.08	3279.44	59.83
1993	23	18	56	53	46	39	18	3794.33	3241.95	3524.92	3528.29	130.99
1994	11	37	47	42	57	44	15	3978.36	3593.35	3794.58	3798.16	87.77
1995	48	27	24	40	77	10	29	5216.47	3832.08	4499.39	4585.84	383.03
1996	18	21	108	49	25	11	29	6560.9	5032.24	5743.89	5666.88	352.04
1997	19	45	27	24	36	66	39	8259.30	6391.69	7443.15	7636.62	525.54
1998	23	27	14	40	40	76	37	9374.27	7539.06	8631.94	8796.07	487.21
1999	35	15	19	28	62	71	28	11497.12	9120.67	10482.61	10659.43	645.52

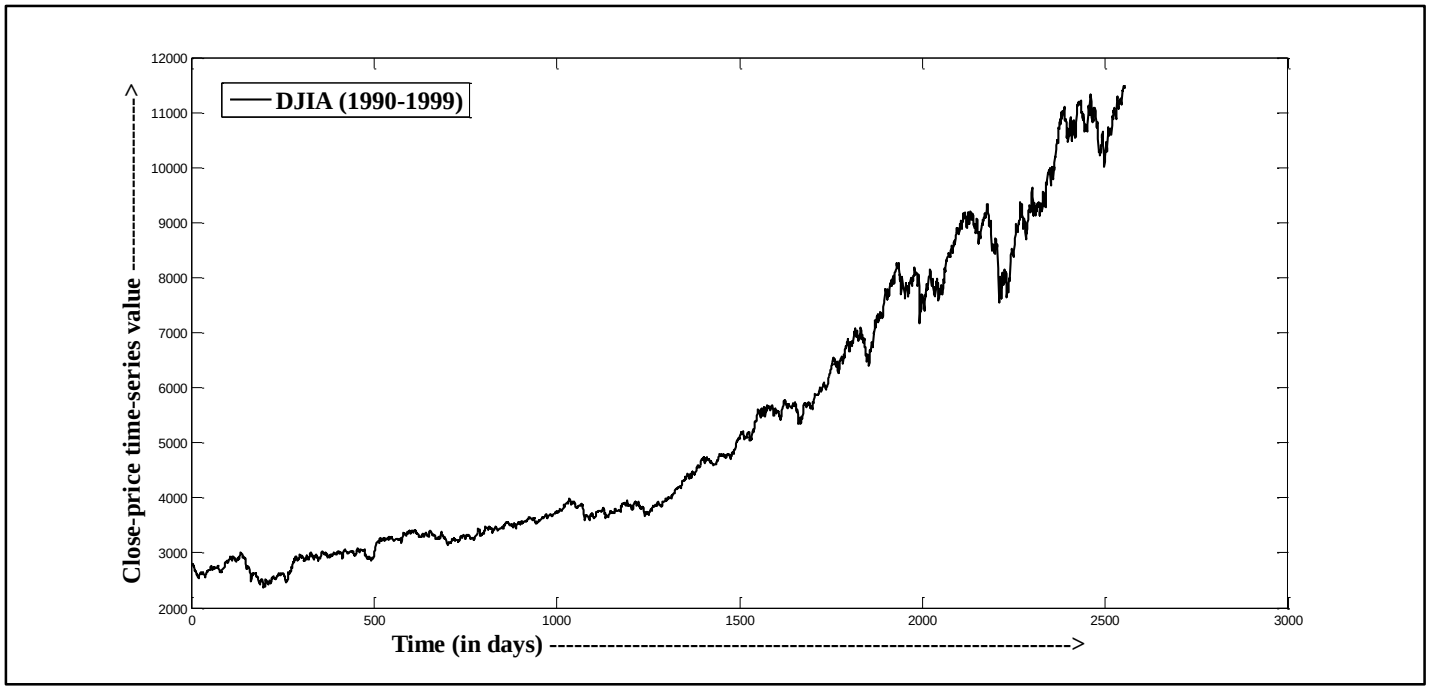


Fig. A.4. Time-series plot of DJIA economic close-price time-series for the period 1990-1999

V. Regularly Occurring Structures Extracted From The Three Daily Close-Price Time-Series: TAIEX, NASDAQ and DJIA

In this section, we present the structures extracted from the daily close-price values of three economic time-series: TAIEX, NASDAQ and DJIA. It should be noted that the structures are extracted from the time-series corresponding to the entire period 1990 to 1999. The obtained structures for TAIEX, NASDAQ and DJIA are illustrated in Fig. A.5, Fig. A.6 and Fig. A.7 respectively.

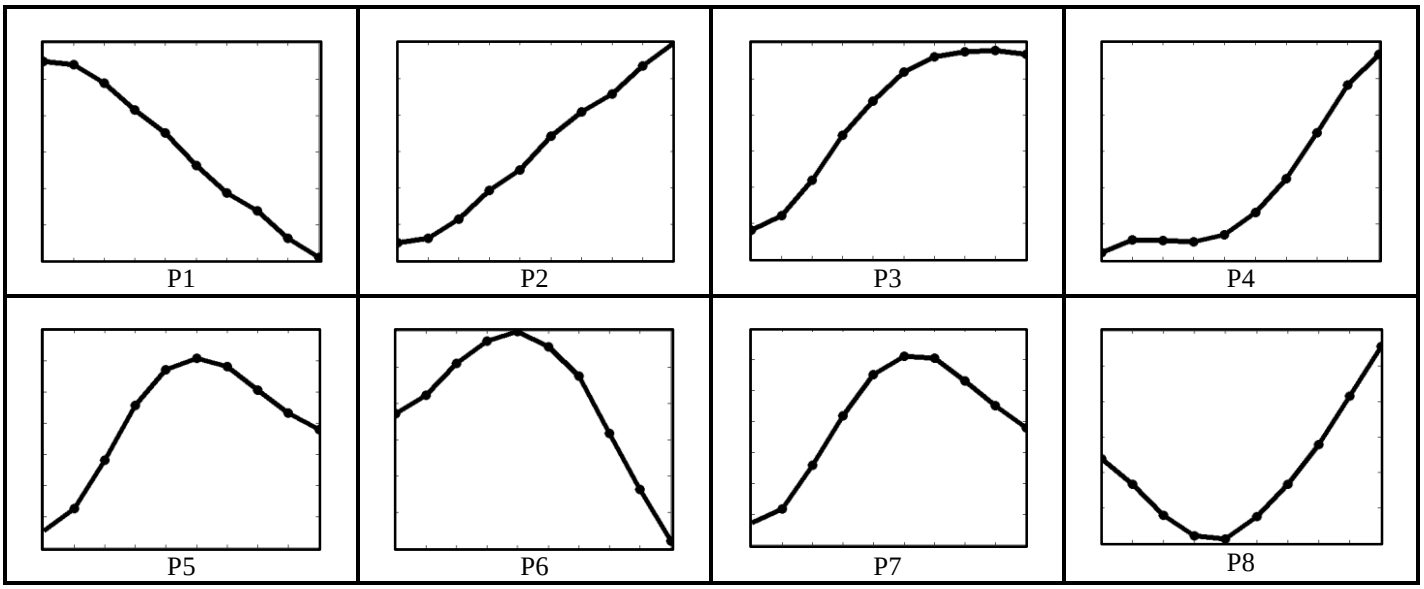


Fig. A.5. The structures (recurrent patterns) obtained from the TAIEX close-price economic time-series of the years 1990-1999. There are eight patterns P1 to P8.

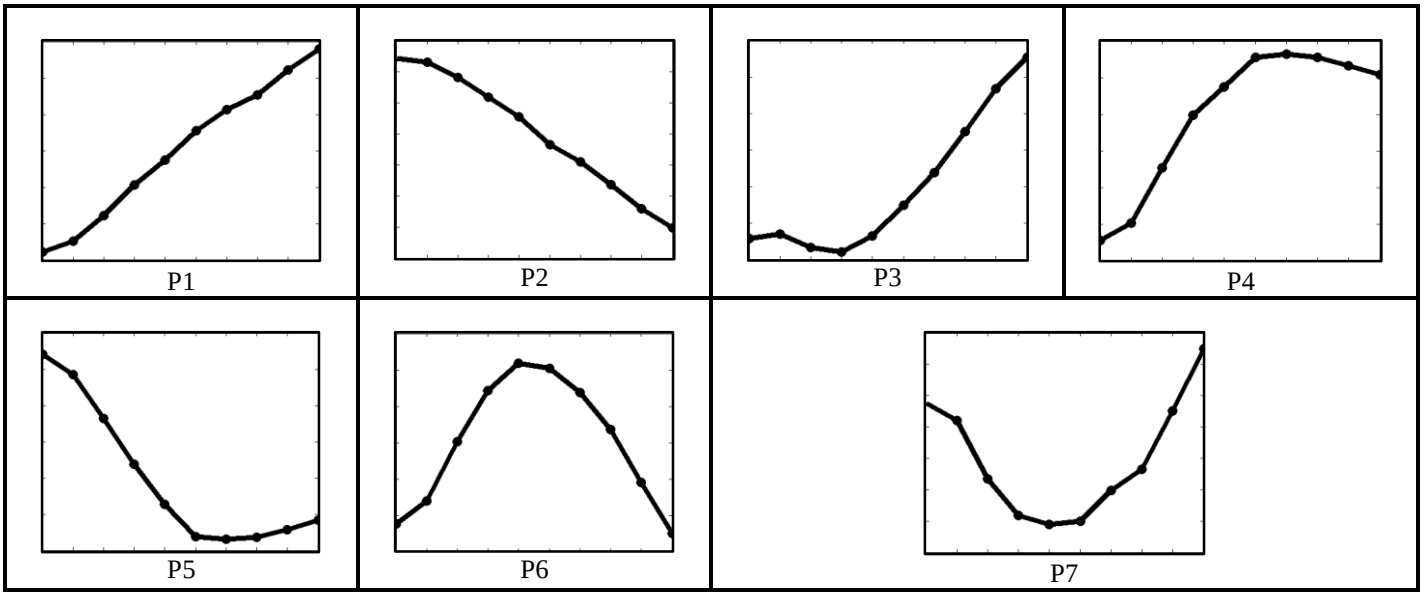


Fig. A.6. The structures (recurrent patterns) obtained from the NASDAQ close-price economic time-series of the years 1990-1999. There are seven patterns P1 to P7.

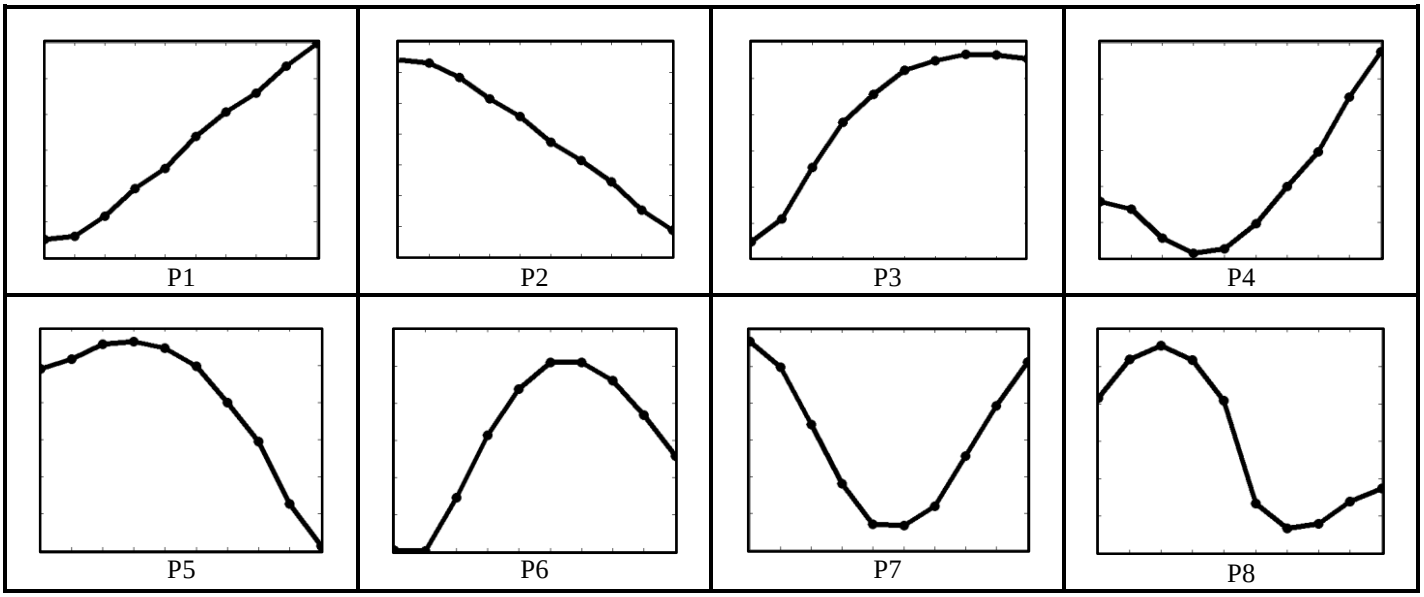


Fig. A.7. The structures (recurrent patterns) obtained from the DJIA close-price economic time-series of the years 1990-1999. There are eight patterns P1 to P8.

VI. Prediction Results Obtained from the for the years 1990 to 1999

In this section, we present the prediction results obtained from the test phase (November and December) of each year in the period 1990-1999 for all the three time-series, TAIEX, NASDAQ and DJIA. The corresponding DSAs are obtained from the training phase (January to October) time-series of each year in the period 1990-1999. The set of 10 DSAs for TAIEX, NASDAQ and DJIA are illustrated in Fig. A.8, Fig. A.9 and Fig. A.10 respectively.

The test phase structure prediction results for TAIEX, NASDAQ and DJIA are summarized in Tables A.IV, A.V and A.VI respectively. Also, the predicted sequence of data points along with their actual counterparts are plotted in Fig. A.11, A.12 and A.13 respectively.

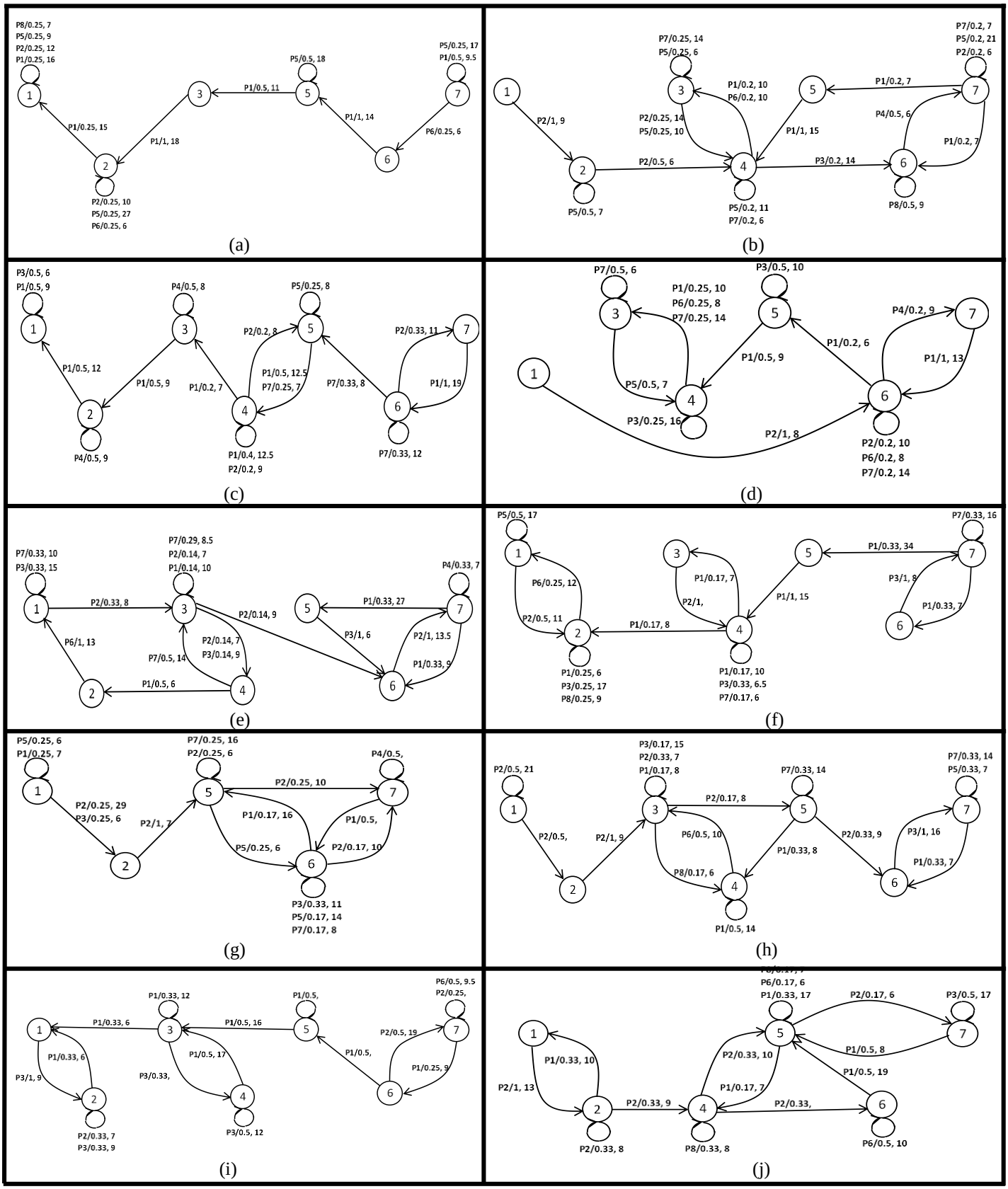


Fig. A.8. The Dynamic Stochastic Automata (DSA) obtained from the TAIEX close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

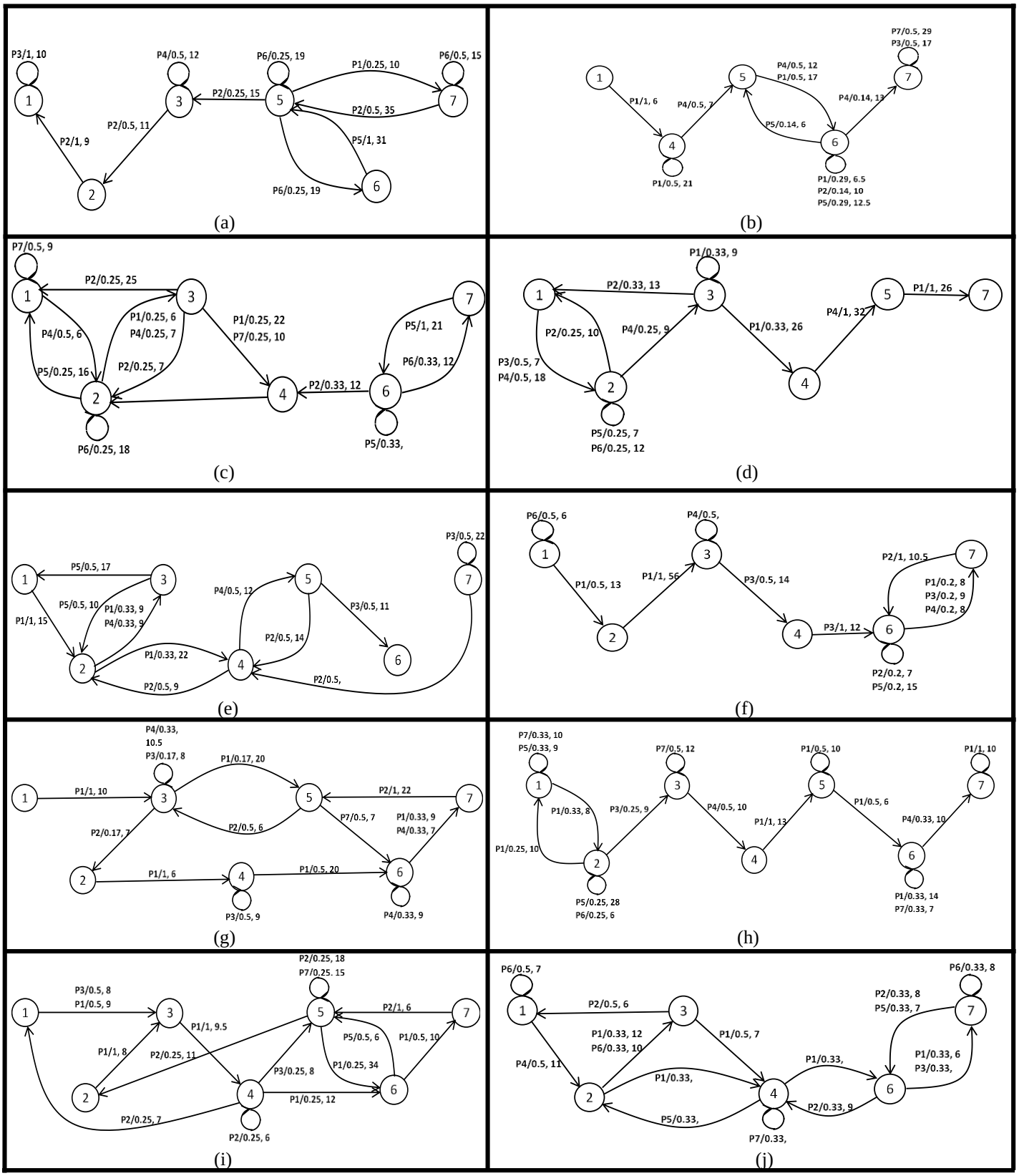


Fig. A.9. The Dynamic Stochastic Automata (DSA) obtained from the NASDAQ close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

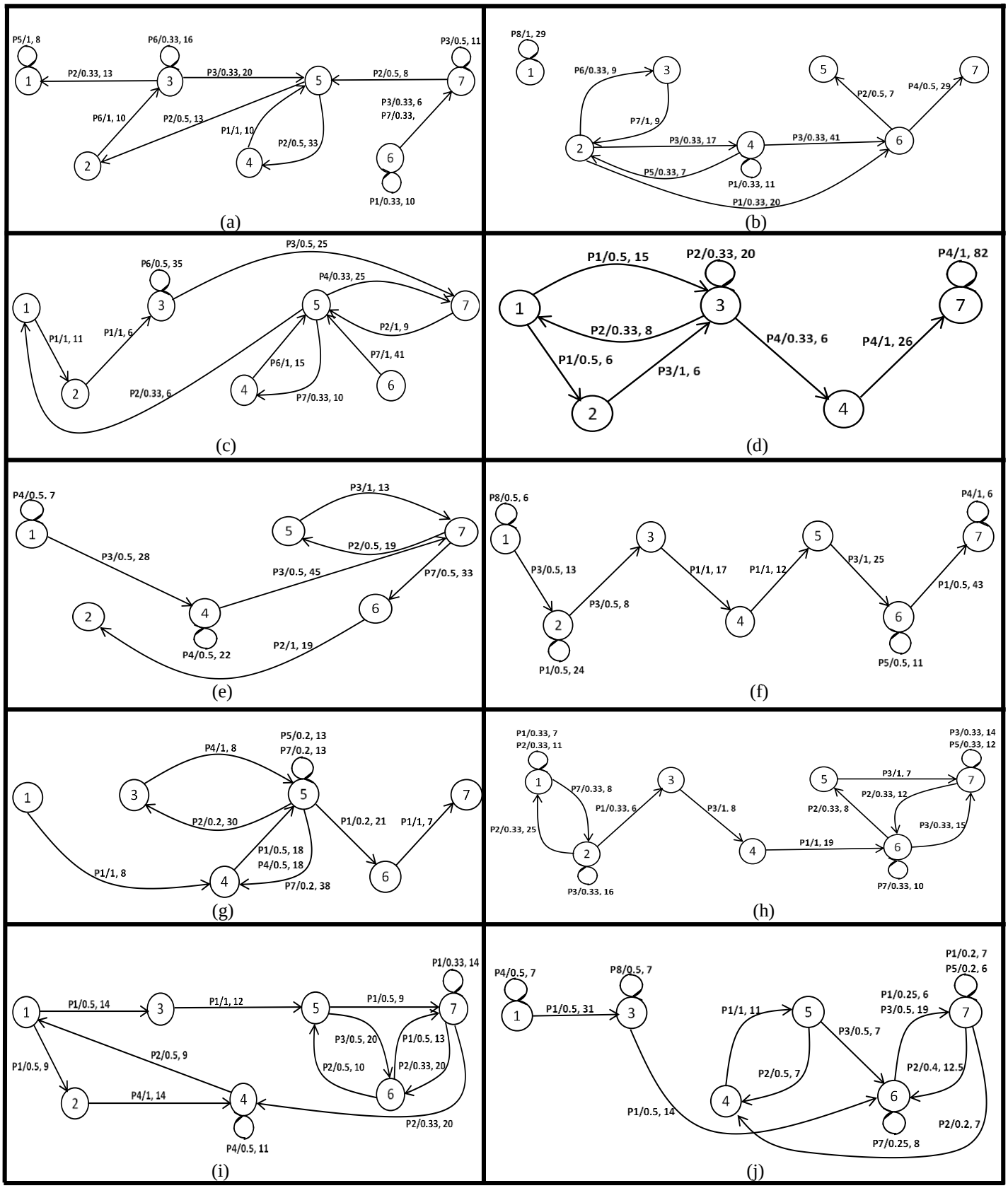


Fig. A.10. The Dynamic Stochastic Automata (DSA) obtained from the DJIA close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

TABLE A.IV
STRUCTURE PREDICTION RESULTS OBTAINED FROM TAIEX CLOSE-PRICE TIME-SERIES

Year of experiment	Segment number	Start partition (state)	Target partition (state)	Predicted pattern	Actual pattern	RMSE
1990	1	1	2	N.A.	P2	98.8786
	2	2	2	P5	P1	
	3	**2**	**2**	**P5**	**P5**	
	4	**2**	**2**	**P5**	**P5**	
	5	**2**	**1**	**P1**	**P1**	
1991	**6**	**3**	**3**	**P5**	**P5**	40.1389
	7	**3**	**3**	**P5**	**P5**	
	8	**3**	**3**	**P5**	**P5**	
	9	**3**	**3**	**P5**	**P5**	
	10	3	3	P5	P1	
	11	3	3	P5	P2	
	12	3	3	P5	P1	
	13	**3**	**3**	**P5**	**P5**	
1992	14	1	1	P1	P4	62.1855
	15	**1**	**1**	**P1**	**P1**	
	16	1	1	P1	P2	
	17	**2**	**2**	**P4**	**P4**	
	18	**2**	**1**	**P1**	**P1**	
1993	19	3	3	P7	P2	41.2490
	20	**3**	**4**	**P5**	**P5**	
	21	**4**	**4**	**P3**	**P3**	
	22	**5**	**4**	**P1**	**P1**	
	23	4	5	N.A.	P2	
	24	**5**	**5**	**P3**	**P3**	
	25	**5**	**5**	**P3**	**P3**	
	26	4	7	N.A.	P2	
1994	27	5	5	N.A.	P2	62.7116
	28	5	5	N.A.	P7	
	29	**5**	**6**	**P3**	**P3**	
	30	6	6	N.A.	P5	
	31	**6**	**7**	**P2**	**P2**	
	32	**7**	**7**	**P4**	**P4**	
1995	**33**	**1**	**1**	**P5**	**P5**	42.7068
	34	1	1	P5	P1	
	35	**1**	**1**	**P5**	**P5**	
	36	1	1	P5	P2	
	37	**1**	**1**	**P5**	**P5**	
	38	**1**	**2**	**P2**	**P2**	
	39	2	2	P1	P2	
1996	**40**	**7**	**7**	**P4**	**P4**	20.3955
	41	**7**	**7**	**P4**	**P4**	
	42	**7**	**7**	**P4**	**P4**	
	43	**7**	**7**	**P4**	**P4**	
	44	7	7	P4	P1	
	45	**7**	**7**	**P4**	**P4**	
	46	7	7	P4	P2	
	47	7	7	P4	P5	
1997	48	2	2	N.A.	P1	98.3012
	49	**1**	**1**	**P2**	**P2**	
	50	**1**	**2**	**P2**	**P2**	
	51	2	2	N.A.	P1	
	52	**2**	**3**	**P2**	**P2**	
	53	3	3	P2	P5	
	54	3	3	P2	P1	
1998	**55**	**2**	**2**	**P3**	**P3**	80.0654
	56	**3**	**3**	**P1**	**P1**	
	57	3	2	N.A.	P1	
	58	**2**	**2**	**P3**	**P3**	
	59	**2**	**1**	**P1**	**P1**	
1999	60	5	6	N.A.	P2	41.2819
	61	**6**	**6**	**P6**	**P6**	
	62	6	6	P6	P1	
	63	6	6	P6	P2	
	64	**6**	**6**	**P6**	**P6**	
	65	**6**	**6**	**P6**	**P6**	
	66	6	6	P6	P2	
					Average RMSE	**58.7914**

TABLE A.V
STRUCTURE PREDICTION RESULTS OBTAINED FROM NASDAQ CLOSE-PRICE TIME-SERIES

Year of experiment	Segment number	Start partition (state)	Target partition (state)	Predicted pattern	Actual pattern	RMSE
1990	**1**	**5**	**5**	**P6**	**P6**	1.7529
	2	**5**	**5**	**P6**	**P6**	
	3	**5**	**7**	**P1**	**P1**	
	4	**7**	**7**	**P6**	**P6**	
1991	**5**	**7**	**7**	**P3**	**P3**	6.6067
	6	7	7	P3	P6	
	7	7	7	P3	P2	
	8	**7**	**7**	**P3**	**P3**	
	9	**7**	**7**	**P3**	**P3**	
	10	**7**	**7**	**P3**	**P3**	
	11	**7**	**7**	**P3**	**P3**	
	12	**7**	**7**	**P3**	**P3**	
1992	13	1	3	N.A.	P1	1.7859
	14	3	3	N.A.	P4	
	15	**3**	**2**	**P2**	**P2**	
	16	**2**	**2**	**P6**	**P6**	
1993	17	3	3	P1	P2	3.7424
	18	**2**	**1**	**P2**	**P2**	
	19	**1**	**2**	**P4**	**P4**	
	20	**2**	**2**	**P5**	**P5**	
1994	21	5	5	N.A.	P1	1.5146
	22	5	5	N.A.	P4	
	23	5	2	N.A.	P2	
	24	**2**	**3**	**P1**	**P1**	
1995	**25**	**7**	**6**	**P2**	**P2**	2.5885
	26	**6**	**6**	**P2**	**P2**	
	27	**6**	**7**	**P1**	**P1**	
	28	7	7	N.A.	P6	
	29	**7**	**6**	**P2**	**P2**	
1996	**30**	**4**	**4**	**P3**	**P3**	3.4207
	31	**4**	**4**	**P3**	**P3**	
	32	**4**	**4**	**P3**	**P3**	
	33	4	4	P3	P4	
	34	4	4	P3	P2	
	35	**4**	**4**	**P3**	**P3**	
	36	4	4	P3	P2	
1997	**37**	**6**	**6**	**P1**	**P1**	3.7377
	38	**6**	**6**	**P1**	**P1**	
	39	**6**	**6**	**P1**	**P1**	
	40	**6**	**6**	**P1**	**P1**	
	41	6	5	N.A.	P2	
1998	**42**	**3**	**4**	**P1**	**P1**	10.5144
	43	**4**	**4**	**P2**	**P2**	
	44	**4**	**4**	**P2**	**P2**	
	45	4	4	P2	P1	
1999	**46**	**3**	**4**	**P1**	**P1**	10.3728
	47	**4**	**4**	**P7**	**P7**	
	48	**4**	**6**	**P1**	**P1**	
					Average RMSE	**4.6037**

TABLE A.VI
STRUCTURE PREDICTION RESULTS OBTAINED FROM DJIA CLOSE-PRICE TIME-SERIES

Year of experiment	Segment number	Start partition (state)	Target partition (state)	Predicted pattern	Actual pattern	RMSE
1990	1	1	2	N.A.	P1	11.8403
	2	2	2	N.A.	P2	
	3	2	3	P6	P1	
	4	**3**	**3**	**P6**	**P6**	
	5	**3**	**3**	**P6**	**P6**	
	6	**3**	**3**	**P6**	**P6**	
1991	7	5	5	N.A.	P2	16.7944
	8	4	2	P5	P2	
	9	2	2	N.A.	P8	
	10	**2**	**4**	**P3**	**P3**	
1992	**11**	**3**	**3**	**P6**	**P6**	10.8323
	12	4	4	N.A.	P4	
	13	4	5	P6	P1	
	14	5	5	N.A.	P6	
	15	**5**	**1**	**P2**	**P2**	
1993	16	7	7	P4	P1	12.9804
	17	**7**	**7**	**P4**	**P4**	
	18	7	7	P4	P1	
	19	**7**	**7**	**P4**	**P4**	
	20	**7**	**7**	**P4**	**P4**	
1994	**21**	**4**	**4**	**P4**	**P4**	17.5341
	22	**4**	**4**	**P4**	**P4**	
	23	**4**	**7**	**P3**	**P3**	
	24	7	7	N.A.	P7	
	25	**7**	**5**	**P2**	**P2**	
1995	**26**	**7**	**7**	**P4**	**P4**	21.5288
	27	**7**	**7**	**P4**	**P4**	
	28	**7**	**7**	**P4**	**P4**	
	29	7	7	P4	P1	
	30	7	7	P4	P2	
	31	**7**	**7**	**P4**	**P4**	
	32	**7**	**7**	**P4**	**P4**	
1996	33	5	5	P7	P1	13.7326
	34	**5**	**5**	**P7**	**P7**	
	35	5	5	P7	P2	
	36	**5**	**5**	**P7**	**P7**	
	37	**5**	**6**	**P1**	**P1**	
	38	6	6	N.A.	P3	
1997	39	5	5	N.A.	P8	62.0058
	40	5	5	N.A.	P1	
	41	**5**	**7**	**P3**	**P3**	
	42	**7**	**7**	**P3**	**P3**	
	43	**7**	**6**	**P2**	**P2**	
1998	**44**	**6**	**7**	**P1**	**P1**	37.1841
	45	**7**	**7**	**P1**	**P1**	
	46	**7**	**6**	**P2**	**P2**	
	47	**6**	**5**	**P2**	**P2**	
	48	**6**	**7**	**P1**	**P1**	
1999	49	5	7	N.A.	P1	11.5280
	50	7	7	P1	P2	
	51	**7**	**7**	**P1**	**P1**	
	52	**7**	**7**	**P1**	**P1**	
	53	**7**	**7**	**P1**	**P1**	
	54	**7**	**7**	**P1**	**P1**	
	55	**7**	**7**	**P1**	**P1**	
					Average RMSE	**21.5961**

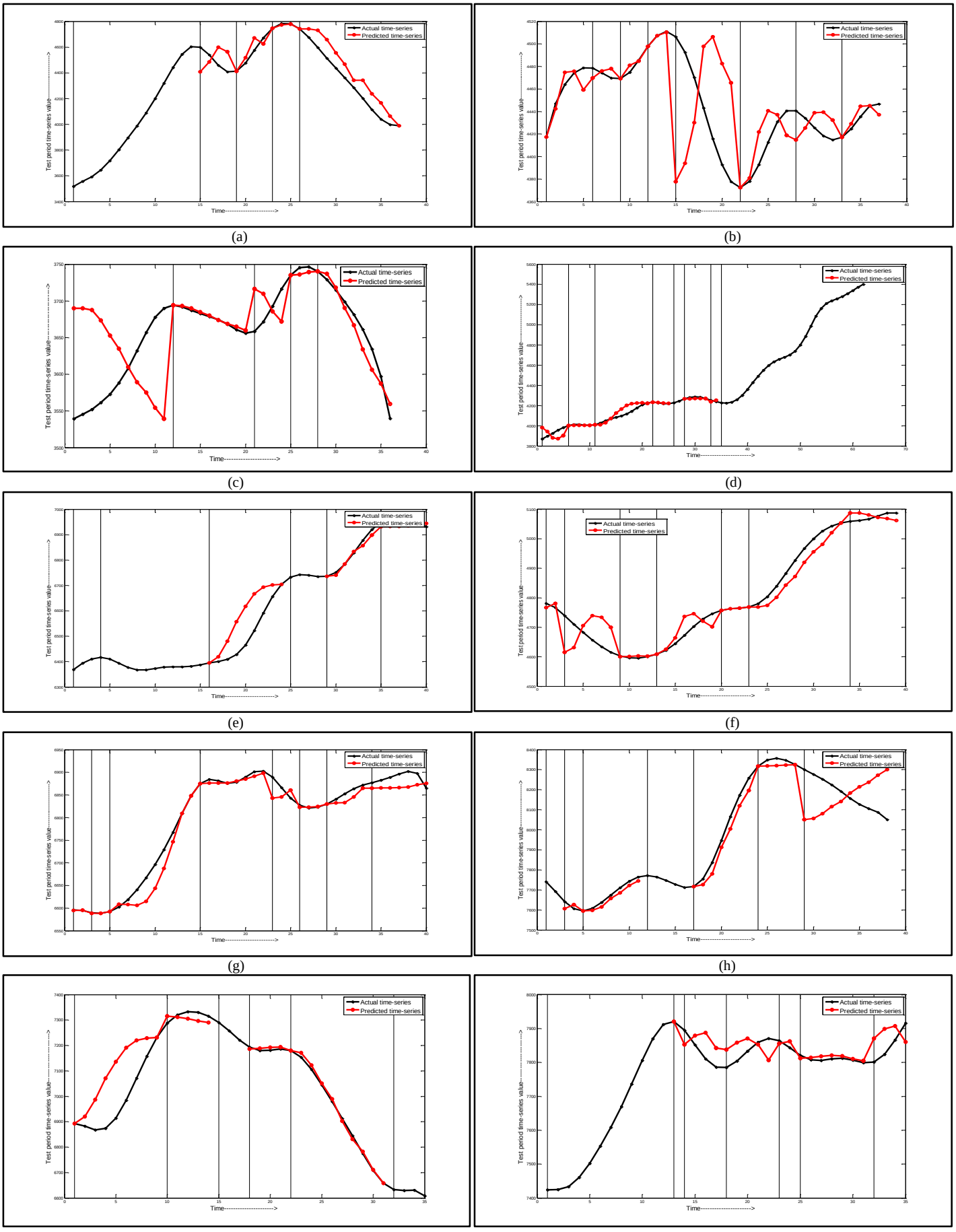


Fig. A.11. Prediction of test period (November and December) TAIEX close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

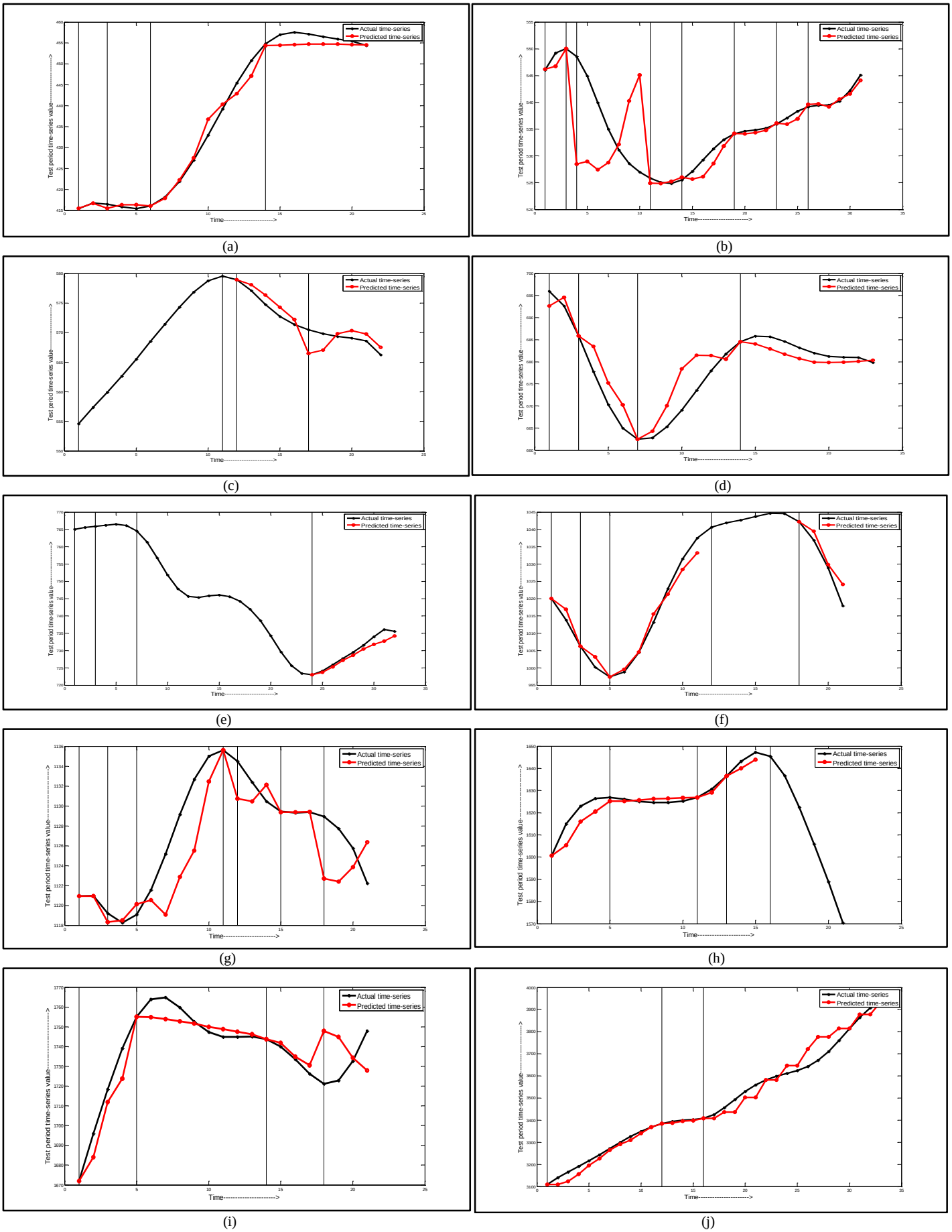
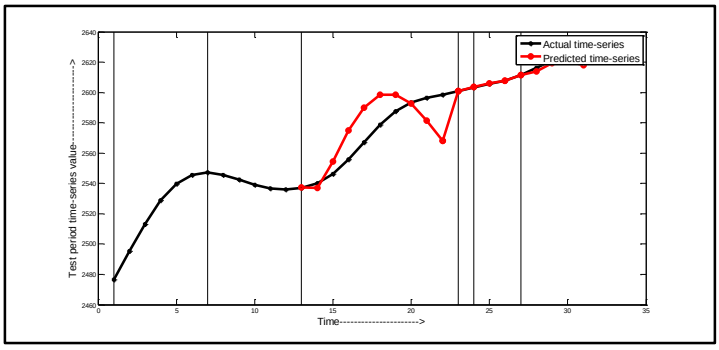
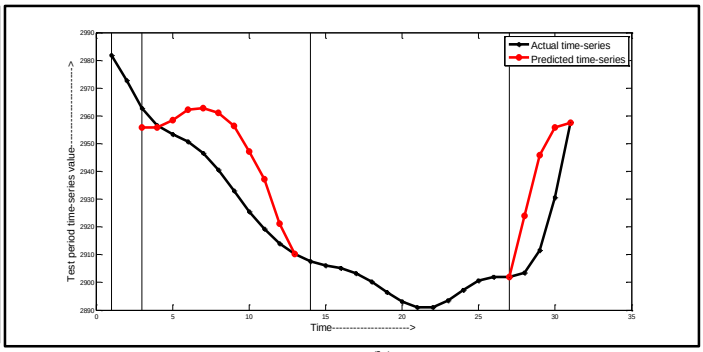
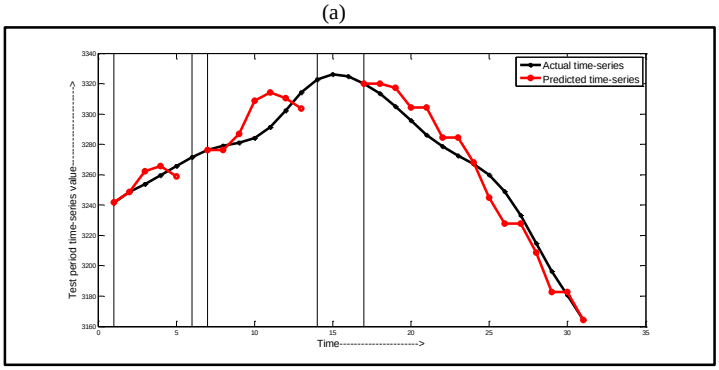
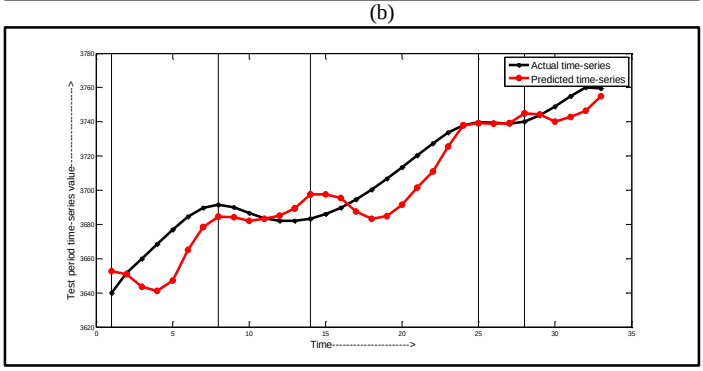


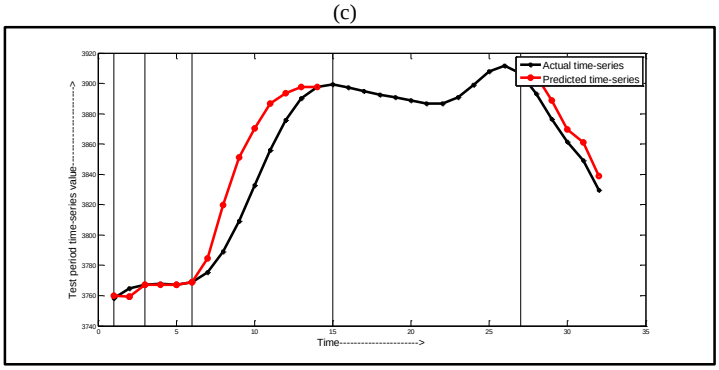
Fig. A.12. Prediction of test period (November and December) NASDAQ close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

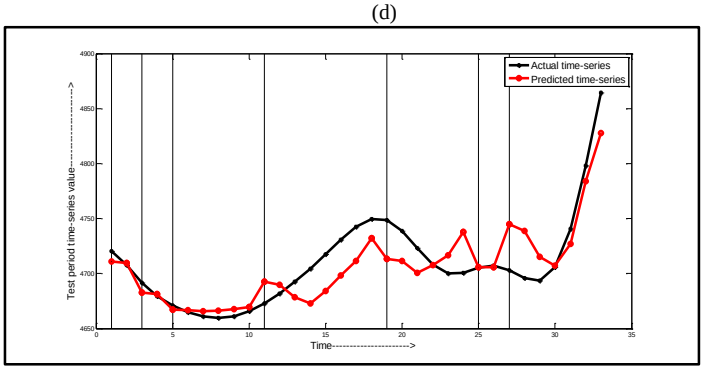


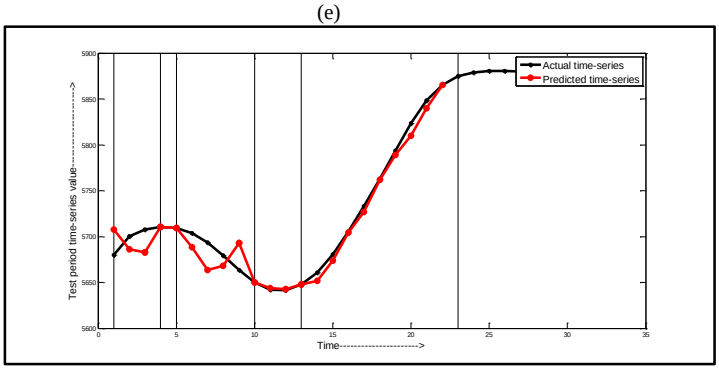


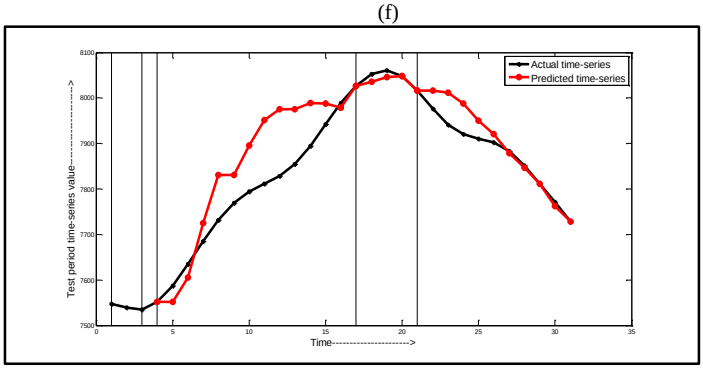


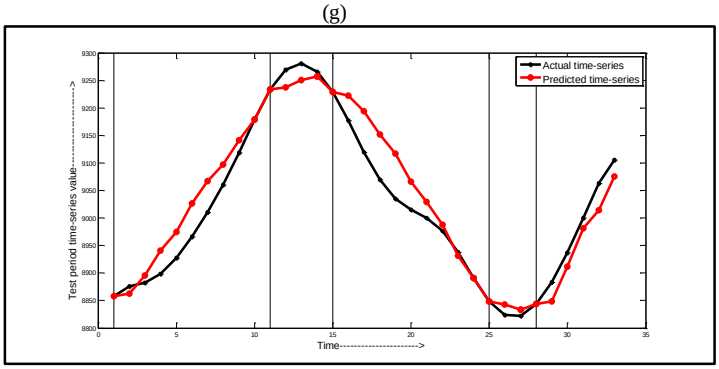












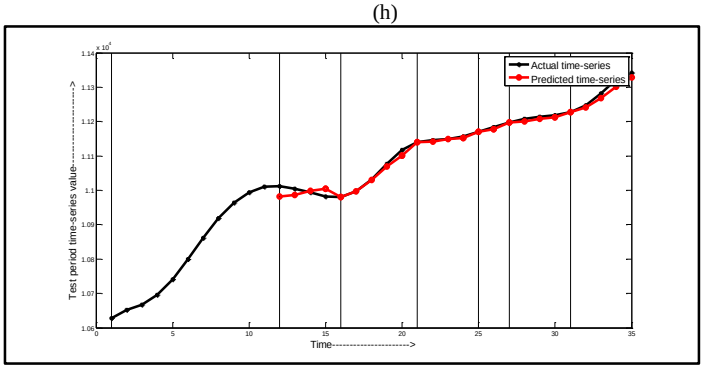


Fig. A.13. Prediction of test period (November and December) DJIA close price economic time-series of the years: (a) 1990, (b) 1991, (c) 1992, (d) 1993, (e) 1994, (f) 1995, (g) 1996, (h) 1997, (i) 1998 and (j) 1999.

VII. Evaluation of the Proposed Model For Structure Prediction

In this section, we analyse the structure prediction results obtained from the previous section and evaluate the performance of the proposed model for structure prediction in real life scenarios. The confusion matrices corresponding to TAIEX, NASDAQ and DJIA are presented in Table A.VII, Table A.IX, and Table A.XI. The precision, recall and accuracy measures computed from the above mentioned matrices are given in Tables A.VIII, A.X and A.XII.

TABLE A.VII
CONFUSION MATRIX FOR STRUCTURE PREDICTION ON TAIEX (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Actual structure / Predicted structure	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**Row sum**
P1	6	2	0	1	0	0	0	0	**9**
P2	1	5	0	0	1	0	0	0	**7**
P3	0	0	6	0	0	0	0	0	**6**
P4	1	1	0	7	1	0	0	0	**10**
P5	4	2	0	0	11	0	0	0	**17**
P6	1	2	0	0	0	3	0	0	**6**
P7	0	1	0	0	0	0	0	0	**1**
P8	0	0	0	0	0	0	0	0	**0**
Column Sum	**13**	**13**	**6**	**8**	**13**	**3**	**0**	**0**	**Total = 56**

TABLE A.VIII
EVALUATION METRICS FOR STRUCTURE PREDICTION ON TAIEX (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Structure / Evaluation metric	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**Macro-averaged evaluation metric**
Precision	0.67	0.71	1	0.7	0.65	0.5	0	-	**0.604 (60.4%)**
Recall	0.46	0.39	1	0.88	0.85	1	-	-	**0.763 (76.3%)**
Accuracy	**0.6786 (67.86%)**								

TABLE A.IX
CONFUSION MATRIX FOR STRUCTURE PREDICTION ON NASDAQ (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Actual structure / Predicted structure	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**Row sum**
P1	10	1	0	0	0	0	0	**11**
P2	1	7	0	0	0	0	0	**8**
P3	0	3	10	1	0	1	0	**15**
P4	0	0	0	1	0	0	0	**1**
P5	0	0	0	0	1	0	0	**1**
P6	0	0	0	0	0	4	0	**4**
P7	0	0	0	0	0	0	1	**1**
Column Sum	**11**	**11**	**10**	**2**	**1**	**5**	**1**	**Total = 41**

TABLE A.X
EVALUATION METRICS FOR STRUCTURE PREDICTION ON NASDAQ (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Structure / Evaluation metric	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**Macro-averaged evaluation metric**
Precision	0.91	0.88	0.67	1	1	1	1	**0.923 (92.3%)**
Recall	0.91	0.64	1	0.5	1	0.8	1	**0.836 (83.6%)**
Accuracy	**0.8293 (82.93%)**							

TABLE A.XI
CONFUSION MATRIX FOR STRUCTURE PREDICTION ON DJIA (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Actual structure / Predicted structure	P1	P2	P3	P4	P5	P6	P7	P8	Row sum
P1	9	1	0	0	0	0	0	0	**10**
P2	0	5	0	0	0	0	0	0	**5**
P3	0	0	4	0	0	0	0	0	**4**
P4	3	1	0	10	0	0	0	0	**14**
P5	0	1	0	0	0	0	0	0	**1**
P6	2	0	0	0	0	4	0	0	**6**
P7	1	1	0	0	0	0	2	0	**4**
P8	0	0	0	0	0	0	0	0	**0**
Column Sum	**15**	**9**	**4**	**10**	**0**	**4**	**2**	**0**	**Total = 44**

TABLE A.XII
EVALUATION METRICS FOR STRUCTURE PREDICTION ON DJIA (1990-1999) ECONOMIC CLOSE-PRICE TIME-SERIES

Structure / Evaluation metric	P1	P2	P3	P4	P5	P6	P7	P8	Macro-averaged evaluation metric
Precision	0.9	1	1	0.71	1	0.67	0.5	-	**0.826 (82.6%)**
Recall	0.6	0.56	1	1	-	1	1	-	**0.86 (86%)**
Accuracy	**0.7727 (77.27%)**								

VIII. COMPARISON OF THE PROPOSED MODEL WITH OTHER ECONOMETRIC TIME-SERIES PREDICTION MODELS ON TAIEX SERIES

In this section, we present a comparison of the proposed model with other existing econometric time-series prediction models. The comparison is performed on the TAIEX time-series for the period 1990 – 1999 with the well-known RMSE, as the metric of choice. The RMSE values obtained in each year for all of the above mentioned models are given in Table A.XIII. The lowest RMSE values for each year are shown in bold.

TABLE A.XIII
COMPARISON OF PROPOSED MODEL WITH EXISTING MODELS ON TAIEX (1990-1999) TIME-SERIES BASED ON THE RMSE METRIC

Years / Models	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	Average
1.Conventional models [10]	220	80	60	110	112	79	54	148	167	149	117.9
2.Weighted models [10]	227	61	67	105	135	70	54	133	151	142	114.5
3.Chen and Chen [11]	172.89	72.87	43.44	103.21	78.63	66.66	59.75	139.68	124.44	115.47	97.70
4. Chen *et. al* [12]	174.62	43.22	42.66	104.17	94.6	54.24	50.5	138.51	117.87	101.33	92.17
5. Chen and Kao [13]	156.47	56.50	36.45	126.45	**62.57**	105.52	51.50	125.33	104.12	87.63	91.25
6. Cai *et. al* [14]	187.10	**39.58**	39.37	101.80	76.32	56.05	49.45	123.98	118.41	102.34	89.44
7. Lin *et. al* [9]	148.32	51.73	**32.81**	85.42	80.24	49.13	50.21	106.58	96.41	56.35	75.72
8. Proposed method	**98.88**	40.14	62.19	**41.25**	62.71	**42.71**	**20.40**	**98.30**	**80.07**	**41.28**	**58.79**

IX. ADDITIONAL RESULTS ON PERFORMANCE ANALYSIS AND COMPARISON OF PROPOSED MULTILEVEL DBSCAN WITH TRADITIONAL DBSCAN

In this section, we present some additional results on the comparison of the proposed multilevel DBSCAN with the original DBSCAN algorithm. These results could not be included in the original paper due to the lack of space. In Fig. A.14, we present the cluster center patterns obtained by the traditional DBSCAN algorithm when executed on the TAIEX, 1990-2000 close-price time-series. In Table A. XIV, we present the cluster densities obtained by our proposed clustering technique along with that of original DBSCAN on the same data set (i.e., TAIEX 1990-2000 close-price time-series). It is observed that the proposed clustering method automatically generates clusters in decreasing order of cluster density.

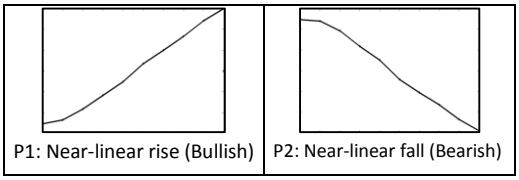


Fig. A.14. Cluster patterns generated by the traditional DBSCAN algorithm when executed on the TAIEX, 1990-2000 close-price time-series

TABLE A.XIV
CLUSTER DENSITIES OF THE PATTERNS OBTAINED FROM DBSCAN AND MULTI-LAYERED DBSCAN WHEN EXECUTED ON TEMPORAL SEGMENTS OBTAINED FROM TAIEX (1990-2000)

DBSCAN (Pattern : Density of pattern)	Multi-layered DBSCAN (Pattern : Density of pattern)
P1: 17.8137	P1: 31.4546
	P2: 29.8410
	P3: 14.5017
	P4: 11.6547
	P5: 7.1405
P2: 16.4261	P6: 6.0197
	P7: 4.2563
	P8: 3.4682
	P9: 3.2773