

# GRAPH BASED FRAMEWORK FOR TIME SERIES PREDICTION

## A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

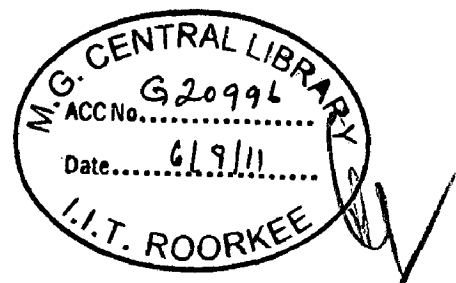
**MASTER OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

By

**VIVEK YADAV**



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE -247 667 (INDIA)  
JUNE, 2011

## CANDIDATE'S DECLARATION

---

I hereby declare that the work, which is being presented in the dissertation entitled "GRAPH BASED FRAMEWORK FOR TIME SERIES PREDICTION" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology in Computer Science and Engineering** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2010 to June 2011, under the guidance of **Dr. D. Toshniwal, Assistant Professor**, Department of Electronics and Computer Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: 17 June 2011

Place: Roorkee

  
(VIVEK YADAV)


---

## CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 17 June 2011

Place: Roorkee

  
(Dr. D. Toshniwal)

Assistant Professor

Department of Electronics and Computer Engineering

IIT Roorkee.

# ACKNOWLEDGEMENTS

---

First and foremost, I would like to extend my heartfelt gratitude to my guide and mentor **Dr. D. Toshniwal**, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for her invaluable advices, guidance, encouragement and for sharing her broad knowledge. Her wisdom, knowledge and commitment to the highest standards inspired and motivated me. She has been very generous in providing the necessary resources to carry out my research. She is an inspiring teacher, a great advisor, and most importantly a nice person.

I also wish to thank Kehar Singh for his valuable suggestions and timely help regarding the domain knowledge and datasets. I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample moral supports and valuable suggestions.

On a personal note, I owe everything to the Almighty and my parents. The support which I enjoyed from my father, mother and other family members provided me the mental support I needed.

**VIVEK YADAV**

# ABSTRACT

---

A time-series is a sequence of real numbers where each number represents a measured value with respect to time. Data mining when performed on time series data is called time series data mining. Time series data mining comprises of tasks like clustering, dimensionality reduction, association rule mining, classification and prediction. A major task of data mining with regard to time series is predicting the future values. The time series prediction can be useful in many ways such as planning, measuring the performance of a predicted value based on the past data against actual observed value.

Existing methods of time series prediction like fitting a function, use of least square method, exponential method and ARIMA method are statistical approaches which require experience and prior knowledge by the user. These methods also do not consider about evolving patterns in time series over periods of time. Thus there exists a major research challenge of dealing with evolving patterns in time series. We propose a new approach for time series prediction that is based on graph framework and takes care of evolving patterns in time series data. This framework overcomes the problem by extracting concealed patterns in time series using concept of graph and graph mining techniques. The motivation behind using graphs on time series data is that graphs can model each observation as vertex and represent the affect of variation in observations with respect to time in form of edges. We, thus propose to map time series data to graphs and use graph edit distance measure for computing similarity in time series data being represented as graphs. This similarity is being used for purpose of classification and prediction.

In the present work, the experiments to evaluate the graph based framework for time series prediction were done on two real world datasets which are - Airline Passenger data set (measures the number of passengers who travelled between Jan, 1949 to Dec, 1960 by air), and carbon di oxide concentration dataset measured in parts per million from May, 1974 to September, 1987. Our results show that the graph based framework for time series prediction is robust and can be used on diverse time series datasets.

# TABLE OF CONTENTS

<b>Candidate's Declaration and Certificate</b> .....	i
<b>Acknowledgements</b> .....	iii
<b>Abstract</b> .....	v
<b>Table of Contents</b> .....	vii
<b>List of Figures</b> .....	xi
<b>List of Tables</b> .....	xv
<b>Chapter 1 Introduction and Problem Statement</b> .....	<b>1</b>
1.1 Introduction.....	1
1.2 Motivation.....	2
1.3 Statement of the Problem.....	3
1.4 Organization of the Report.....	3
<b>Chapter 2 Background and Literature Review</b> .....	<b>5</b>
2.1 Time Series Data .....	5
2.2 Time Series Modeling.....	5
2.3 Time Series Concepts.....	6
2.3.1 Major Components for Characterizing Time Series .....	6
2.3.2 Decomposition of Time Series.....	6
2.3.3 Deseasonalized Time Series .....	6
2.4 Techniques for Time Series Prediction .....	7
2.4.1 Fitting a Function.....	8
2.4.2 Use Least Square Methods on Deseasonalized Time Series .....	8
2.4.3 ARIMA (Auto-Regressive Integrated Moving Average).....	9
2.4.4 Exponential Smoothing.....	11
2.4.5 Hybrid ARIMA with Neural Network Model .....	12

2.4.6	Support Vector Machines .....	13
2.5	Graph Concepts.....	14
2.5.1	Graph Matching .....	14
2.5.2	Graph Clustering.....	15
2.6	Research Gaps .....	16
<b>Chapter 3 Proposed Framework for Time Series Prediction .....</b>		<b>17</b>
3.1	Major steps in Graph Based Framework for Time series Prediction .....	18
3.1.1	Mapping Time Series Data as Graphs .....	18
3.1.2	Graph Clustering.....	20
3.1.3	Database.....	21
3.1.4	Graph Matching .....	21
3.1.5	Prediction Based on Best Graph Match .....	22
3.1.6	Calculate Error and Learning Based on Actual Recorded Value .....	23
3.2	Applying the Graph Based Framework in Conjunction with Existing Time Series Concepts.....	24
3.3	Flow Diagram of the Graph Based Framework for Time series Prediction..	24
<b>Chapter 4 Implementation of Proposed Framework .....</b>		<b>29</b>
4.1	Modules in Graph Based Framework.....	29
4.1.1	Generation of Graph on Time Series Data.....	29
4.1.2	Graph Clustering.....	31
4.1.3	Database.....	32
4.1.4	Graph Matching .....	33
4.1.5	Predicting Time Series using Graph Representation .....	34
4.1.6	Learning Approach in Graph Based Framework.....	36
4.2	The Experiment dataset.....	36

4.2.1	Airline Passenger Time series (APTS) Dataset .....	36
4.2.2	Dataset of Monthly Carbon di oxide Concentration .....	36
<b>Chapter 5 Results and Discussion .....</b>		<b>39</b>
5.1	Results for Airline Passenger Time Series dataset.....	39
5.2	Results for Monthly Carbon di oxide Concentration Dataset .....	45
5.3	Discussion .....	51
<b>Chapter 6 Conclusion and Future Work .....</b>		<b>53</b>
6.1	Conclusion.....	53
6.2	Suggestions for the Future Work .....	54
<b>REFERENCES .....</b>		<b>55</b>
<b>LIST OF PUBLICATIONS .....</b>		<b>59</b>





Figure 5.1 Represents Actual and Predicted Observations with respect to Time. Predictions are Made Using Graph Based Framework on Time Series Data with Seasonal Fluctuations Without Clustering.....	41
Figure 5.2 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.1. Average Percentage Error is= 7.05. ....	41
Figure 5.3 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data Without Clustering. ....	42
Figure 5.4 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.3. Average Percentage Error is= 5.91. ....	42
Figure 5.5 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Time Series Data with Seasonal Fluctuations and Clustering. ....	43
Figure 5.6 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.5. Average Percentage Error is= 6.63. ....	43
Figure 5.7 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data with Clustering. ....	44
Figure 5.8 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.7. Average Percentage Error is= 5.81. ....	44
Figure 5.9 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on time series with Seasonal Fluctuations Without Clustering. ....	46
Figure 5.10 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.9. Average Percentage Error is= 0.165. ....	47

Figure 5.11 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data Without Clustering. ....47

Figure 5.12 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.11. Average Percentage Error is= 0.163. ....48

Figure 5.13 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Time Series Data with Seasonal Factors with Clustering.....48

Figure 5.14 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.13. Average Percentage Error is= 0.168. ....49

Figure 5.15 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data with Clustering. ....49

Figure 5.16 Represents the Percentage Error Observed while Predicting Time Series for predictions shown in Figure 5.15. Average Percentage Error is 0.165.....50



## LIST OF TABLES

---

Table 3.1 Monthly Data of Airline Passenger Time Series Dataset ( <i>APTS</i> ) is Summarized Quarterly for Year 1949 and 1950.....	19
Table 4.1 Function Name and Class in which Java Code Implemented.....	32
Table 5.1 Average Percentage Error on Airline Passenger Dataset and Error Pattern in Different Scenarios and Reason for Error.....	40
Table 5.2 Average percentage Error on monthly carbon di oxide concentration dataset and Error pattern in different scenarios and reason for such percentage error. ....	46
Table 5.3 Accuracy Rate of Different variants of Graph Based Framework on Different Datasets. ....	50
Table 5.4 Accuracy of Least Square Method on Different Dataset, when Third Degree Polynomial was fitted. ....	50



## Chapter 1

# **Introduction and Problem Statement**

---

In the following sections, a brief introduction and motivation for undertaking the present study and the problem statement for the report has been included.

### **1.1 Introduction**

With the digital revolution, the data that is generated and stored from various business processes is increasing at a tremendous rate. The abundance of data, coupled with the need for powerful data analysis tool, has been described as a data rich but information poor situation [1]. Data Mining, better known as knowledge discovery from Data (KDD), can be described as obtaining possibly unseen and potentially useful information from large data. Data Mining may be better explained as processing data using sophisticated searching capabilities and statistical algorithms, to determine patterns and correlations which are concealed in large preexisting databases [2]. This process of data mining has a great application in the business world, for example, the mined knowledge/information can be used to increase revenue, cut costs or make certain strategic decisions.

A time series is a sequence of real numbers; each number represents the value of the system under study obtained over repeated measurements of time. Time series data arises in many real-world applications. Efficient discovery of knowledge through Time series data mining can be helpful in several domains such as:

- Stock Market Analysis.
- Economic and sales Forecasting.
- Budgetary analysis.
- Inventory studies.
- Observation of natural phenomenon (such as wind, atmosphere, temperature).
- Studying growth patterns of diseases.

Mining Time series data has been a focused theme in data mining research for over three decade. Abundant literature has been dedicated to this research area and

tremendous progress has been made. There are two main goals of data mining on time series data:

1. Identifying the nature of the phenomenon represented by the sequence of observations.
2. Forecasting (predicting future values of the time series variable).

Both of these goals require that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can extrapolate the identified pattern to predict future observations in time series. Thus, time series prediction is one of the major tasks in data mining with regard to time series data. The time series prediction can be useful in planning and measuring the performance of a predicted value based on the past data against actual observed value for the variable under study.

Graphs are the natural way in modeling systems with structural information, such as circuits, chemical compounds, protein structures, biological networks and social networks. Graph mining refers to process of extracting previously unknown potentially useful knowledge from structured data that can be represented as graphs. Graph Mining includes techniques such as frequent pattern mining, clustering [3, 4], matching [3, 5-8] and classification for graph data [3]. In recent times a lot of progress has been done in the field of graph mining due to the greater expressive power associated with graphs and their diverse application domains [3].

## **1.2 Motivation**

In time series prediction, a basic assumption is made that is some aspect of past pattern are likely to continue in future in future [9]. Under this assumption time series prediction is assumed to be based on past values of the variable under study. But changes are inevitable; in time series also the nature of pattern in time series may change over period of time. So the motivation is to propose an approach that can make predictions despite dynamic changes in time series. The idea behind creation of graph over time series is to utilize two facts.

1. Graph is a data structure that is well suited to model a pattern, thus graph can capture the tacit historical pattern present in time series.
2. Similarity between graphs can be calculated to know the similar patterns and their order of their occurrence.

Thus, graph is created with the motivation to store a pattern over time series and make prediction based on similarity of observed pattern from historical data.

### **1.3 Statement of the Problem**

The problem statement for the present work can be stated as follows:

*“To cluster and classify time series data using graph mining techniques.”*

The above problem statement can be divided into the following sub problems:

- *To propose a generic framework for time series prediction.*
- *To map time series data to graphs.*
- *To cluster the similar time series graphs.*
- *To classify patterns and predict future values using graph representation of time series.*
- *To evaluate the effectiveness of proposed algorithms using real world datasets.*

There are two assumptions:

- Time series is univariate.
- Time series data consists of real value attributes only.

### **1.4 Organization of the Report**

This dissertation report comprises of six chapters including this chapter that introduces the topic and states the problem. The rest of the report is organized as follows.

Chapter 2 gives the background of time series data; discuss some of the well-known concepts in time series and brief literature review of existing techniques for time



series predictions. In our framework we use Graph Matching and Graph Clustering, hence a brief review about them is presented, the including research gaps.

Chapter 3 describes the framework designed for time series framework, the major components, their functionality and need in the system.

Chapter 4 gives the implementation details of the proposed framework, details of experiments performed and description of dataset used.

Chapter 5 discusses the results that were obtained using graph bases framework for predicting the time series on real world datasets.

Chapter 6 concludes the dissertation work and gives suggestions for future work.

## Chapter 2

### Background and Literature Review

---

#### 2.1 Time Series Data

A time series is a sequence of real numbers; each number represents the value of the system under study obtained over repeated measurements of time. Mostly these values or events are collected at equally spaced, discrete time intervals (e.g., hourly, daily, weekly, monthly, yearly etc.). When there is only one variable upon which observations with respect to time are made, is called univariate time series. Time series data constitute of  $\{Y_1, Y_2, Y_3 \dots Y_i\}$  values, where each  $Y_i$  represent the value of variable under study at time  $i$ . The successive observations in time series are statistically dependent on time and time series modeling is concerned with techniques for analysis of such dependencies.

#### 2.2 Time Series Modeling

Time series modeling is advantageous, as it can be used more easily for forecasting purposes since the historical sequences of observations upon study on main variable are readily available as they are recorded in form of past observations and can be purchased or gathered from published secondary sources. In time series modeling, the prediction of values for future periods is based on the pattern of past values of the variable under study, but the model does not generally account for explanatory variable which may have affected the system. There are two reasons for resorting to such models as described in [10] :

1. The system may not be understood, and even if it is understood it may be extremely difficult to measure the cause and effect relationship of parameters affecting the time series.
2. The main concern may be only to predict the next value and not to explicitly know why it was observed.

## 2.3 Time Series Concepts

In this section the major concepts regarding time series are presented in subsequent sections.

### 2.3.1 Major Components for Characterizing Time Series

There are four major components that characterize the time series as described in [1]:

1. **Trend component**- these indicate the general direction in which a time series data is moving over a long interval of time, denoted by  $T$ .
2. **Cyclic component**- these refer to the cycles, that is, the long-term oscillations about a trend line or curve, which may or may not be periodic, denoted by  $C$ .
3. **Seasonal component**- these are systematic or calendar related, denoted by  $S$ .
4. **Random component**- these characterize the sporadic motion of time series due to random or chance events, denoted by  $I$ .

### 2.3.2 Decomposition of Time Series

The time series variable  $Y$  at the time  $t$  can be modeled in terms of the four major components Trend ( $T$ ), Cyclic ( $C$ ), Seasonal ( $S$ ) and Random ( $I$ ) with respect to time  $t$ . There are two types of decomposition method based technique they use:

1. **Multiplicative Model**: where time series modeled as product of four components that characterize time series,  $Y_t = T_t \times C_t \times S_t \times I_t$  used in [11].
2. **Additive Model**: where time series modeled as product of four components that characterize time series,  $Y_t = T_t + C_t + S_t + I_t$  used in [12].

### 2.3.3 Deseasonalized Time Series

In time series analysis there is an important notion of deseasonalizing in the time series [13, 14]. It makes the assumption that if the time series represents a seasonal pattern of  $L$  periods, then by taking moving average  $M_t$  of  $L$  periods, we would get the mean value for the year. This would be free of seasonality and contain little randomness (owing to averaging). Thus  $M_t = T_t \times C_t$ . To determine the seasonal component, one would simply divide the original series by the moving average that is,

$Y_t / M_t = (T_t \times C_t \times S_t \times R_t) / (T_t \times C_t) = S_t \times R_t$ . Taking average over months eliminates randomness and yields seasonality component  $S_t$  which are also called seasonal index. Deseasonalized  $Y_t$  time series can be computed by  $Y_t / S_t$ . In Deseasonalized time series, the data does not have systematic or calendar related influences. Seasonal fluctuations conceal both the true underlying movement of the series as well as certain non-seasonal characteristics that may be area of interest.

To detect seasonal patterns, use correlation between the  $i^{th}$  element of the series and  $(i - k)^{th}$  element (where  $k$  is referred to as the lag) using autocorrelation analysis discussed in [13] (Correlation examines if there is an association between two variables and if so, to what extent). Let  $\langle y_1, y_2 \dots y_N \rangle$  be the time series. Apply Eqn. 2.1 the Pearson coefficient; the two attributes in the equation respectively refer to the two random variables representing the time series viewed with lag  $k$ . These times series are  $X = \langle y_1, y_2 \dots y_{N-k} \rangle$  and  $Y = \langle y_{k+1}, y_{k+2} \dots y_N \rangle$ . A zero value indicates that there is no correlation relationship. A positive value indicates a positive correlation, that is, both variables increase together. A negative value indicates a negative correlation, that is, one variable increases as the other decreases. The higher the positive (or negative) value is the greater is the positive (or negative) correlation relationship.

$$R_{X,Y} = \frac{\sum_{i=1}^N [(x_i - \text{mean}(X)) \times (y_i - \text{mean}(Y))]}{N \times \sigma_X \times \sigma_Y} \quad (2.1)$$

## 2.4 Techniques for Time Series Prediction

To predict the time series the trend component needs to be identified. If the time series data contain considerable error, then the first step in the process of trend identification is smoothing.

**Smoothing:** Smoothing always involves some form of local averaging of data such that the nonsystematic components of individual observations cancel each other out. The common techniques are:

1. *Moving average smoothing* which replaces each element of the series by either the simple or weighted average of  $n$  surrounding elements, where  $n$  is the width of the smoothing "window" [15, 16].

## 2. Medians can be used instead of means.

The main advantage of median as compared to moving average smoothing is that its results are less biased by outliers (within the smoothing window). Thus, if there are outliers in the data (e.g., due to measurement errors), median smoothing typically produces smoother or at least more "reliable" curves than moving average based on the same window width. The main disadvantage of median smoothing is that in the absence of clear outliers it may produce more "jagged" curves than moving average and it does not allow for weighting [17]. Hence, Moving average smoothing is commonly used as a preprocessing step.

### 2.4.1 Fitting a Function

Use regression analysis to fit a function to time series. Many monotonous time series data can be adequately approximated by a linear function, or if there is a clear monotonous nonlinear component, the data first need to be transformed to remove the nonlinearity. However, a logarithmic, exponential, or polynomial function can also be used. The curve is extrapolated to predict the values. However, pure regression analysis cannot capture all four movements described above that occur in real-world applications [17].

### 2.4.2 Use Least Square Methods on Deseasonalized Time Series

Use least square method of linear regression [13], where we consider the best-fitting curve  $C$  as the *least squares curve*, that is, the curve having the minimum of  $\sum_{i=1}^n d_i^2$ , where the *deviation* or *error* described in [13, 18],  $d_i$ , is the difference between the value  $y_i$  (value from deseasonalized time series) of a point  $(x_i, y_i)$  and the corresponding value as determined from the curve  $C$ .

The use of regression analysis, require knowledge about the mathematical model of the process. Finding the mathematical model for the real world data is hard. Since, in real-life patterns of the data are unclear, and each observation involves considerable and unusual error. In regression analysis, if the error terms are not independent (*auto correlated*), the efficiency of the least-square parameter estimates is adversely affected and the standard error estimates are biased.

### 2.4.3 ARIMA (Auto-Regressive Integrated Moving Average)

The ARIMA methodology was developed by Box and Jenkins in [15] allows us to deal with time series in which patterns of the data are unclear and each observation involves considerable error; it has gained enormous popularity in many areas and research practice confirms its power and flexibility [19, 20]. Introduction to ARIMA methods are discussed in [17, 21].

#### Type of Processes in ARIMA

1. **Autoregressive Process:** Most time series consist of elements that are serially dependent. The coefficient or a set of coefficients that describe consecutive elements of the series from specific, time-lagged (previous) elements. This can be summarized in the Eqn 2.2:

$$x_t = \zeta + \phi_1 \times x_{(t-1)} + \phi_2 \times x_{(t-2)} + \phi_3 \times x_{(t-3)} + \dots + \varepsilon \quad (2.2)$$

where  $\zeta$  is a constant (intercept),  $\phi_1, \phi_2, \phi_3$  are the autoregressive model parameters and  $\varepsilon$  is random shock.

**Stationarity Requirement:** Note that an autoregressive process will only be stable if the parameters are within a certain range; for example, if there is only one autoregressive parameter then it must fall within the interval of  $-1 < \phi < 1$ . Otherwise, past effects would accumulate and the values of successive  $x_t$ 's would move towards infinity, that is, the series would not be stationary. If there is more than one autoregressive parameter, similar (general) restrictions on the parameter values is defined in [15]. In Time Series analysis, a stationary series has a constant mean, variance, and autocorrelation through time.

2. **Moving Average Process:** Independent from the autoregressive process, each element in the series can also be affected by the past error (or random shock) that cannot be accounted for by the autoregressive component, that is given by Eqn 2.3.

$$x_t = \mu + \varepsilon_t - \theta_1 * \varepsilon_{(t-1)} - \theta_2 * \varepsilon_{(t-2)} - \theta_3 * \varepsilon_{(t-3)} - \dots \quad (2.3)$$

where  $\mu$  is a constant, and  $\theta_1, \theta_2, \theta_3$  are the moving average model parameters.

**Invertible Requirement:** there is a "duality" between the moving average process and the autoregressive process [15], that is, the moving average equation above can be rewritten (*inverted*) into an autoregressive form (of infinite order). Moving average parameters follow condition, of being *invertible*.

## ARIMA METHODOLOGY

- **Autoregressive Moving Average Model.** The general model introduced by Box and Jenkins [15] includes autoregressive as well as moving average parameters, and explicitly includes differencing in the formulation of the model. Specifically, the three types of parameters in the model are: the autoregressive parameters ( $p$ ), the number of differencing passes ( $d$ ), and moving average parameters ( $q$ ). In the notation introduced by Box and Jenkins, models are summarized as ARIMA ( $p, d, q$ ); so, for example, a model described as (0, 1, 2) means that it contains 0 (zero) autoregressive ( $p$ ) parameters and 2 moving average ( $q$ ) parameters which were computed for the series after it was differenced once.
- **Identification.** As mentioned earlier, the input series for ARIMA needs to be stationary, that is, it should have a constant mean, variance, and autocorrelation through time. Therefore, usually the series first needs to be differenced until it is stationary (this also often requires log transforming the data to stabilize the variance). The number of times the series needs to be differenced to achieve stationarity is reflected in the  $d$  parameter (discussed in the previous paragraph). In order to determine the necessary level of differencing, use plot of the data and autocorrelogram. At this stage (which is usually called *Identification* phase) the autoregressive ( $p$ ) and moving average ( $q$ ) parameters are necessary to yield an effective but still *parsimonious* model of the process (*parsimonious* means that it has the fewest parameters and greatest number of degrees of freedom among all models that fit the data) needs to be decided. The major tools used in the identification phase are plots of the series, use of auto correlation.

- **Estimation and Forecasting.** At the next step (*Estimation*), the parameters are estimated (using function minimization procedures, such as minimization of sum of Least squares. The estimates of the parameters are used in the last stage (*Forecasting*) to calculate new values of the series (beyond those included in the input data set) and confidence intervals for those predicted values. The estimation process is performed on transformed (differenced) data; before the forecasts are generated, the series needs to be *integrated* (integration is the inverse of differencing) so that the forecasts are expressed in values compatible with the input data. This automatic integration feature is represented by the letter I in the name of the methodology (ARIMA = Auto-Regressive Integrated Moving Average).

ARIMA (Auto-regressive Integrated Moving Average) methodology is powerful and flexible technique that can deal with unclear patterns in data, error in the data. The Auto-regressive procedure estimates and forecasts linear regression models for time series data when the despite there is autocorrelation among errors or not. But, ARIMA is a complex technique; it is not easy to use, it requires a great deal of experience, as the decision regarding parameters is not straightforward and it requires good deal of experimentation to come up with model and parameters that best describes the underlying system [22]. The approximation of ARIMA models to complex nonlinear process may not be adequate.

#### 2.4.4 Exponential Smoothing

Exponential smoothing has become very popular as a forecasting method for a wide variety of time series data. Gardner proposed a "unified" classification of exponential smoothing methods in [23]. Exponential smoothing methods are also discussed in [24].

A simple and pragmatic model for a time series would be to consider each observation as consisting of a constant ( $b$ ) and an error component  $\varepsilon$  (epsilon), that is:  $X = b + \varepsilon_t$ . The constant  $b$  is relatively stable in each segment of the series, but may change slowly over time. To isolate the true value of  $b$ , and thus the systematic or predictable part of the series, is to compute a kind of moving average, where the current and immediately preceding ("younger") observations are assigned greater weight than the



respective older observations. Simple exponential smoothing accomplishes exactly such weighting, where exponentially smaller weights are assigned to older observations. The specific formula for simple exponential smoothing is:

$$S_t = \alpha * X_t + (1-\alpha) * S_{t-1} \quad (2.4)$$

When applied recursively to each successive observation in the series, each new smoothed value (forecast) is computed as the weighted average of the current observation and the previous smoothed observation; the previous smoothed observation was computed in turn from the previous observed value and the smoothed value before the previous observation, and so on. Thus, in effect, each smoothed value is the weighted average of the previous observations, where the weights decrease exponentially depending on the value of parameter  $\alpha$  (alpha). If  $\alpha$  is equal to 1 (one) then the previous observations are ignored entirely; if  $\alpha$  is equal to 0 (zero), then the current observation is ignored entirely, and the smoothed value consists entirely of the previous smoothed value (which in turn is computed from the smoothed observation before it, and so on; thus all smoothed values will be equal to the initial smoothed value  $S_0$ ). Values of  $\alpha$  in-between will produce intermediate results.

The theoretical properties of (simple and complex) exponential smoothing is discussed in [23, 25]. The method has gained popularity mostly because of its usefulness as a forecasting tool. For example, empirical research by Makridakis *et al.* in [26], has shown simple exponential smoothing to be the best choice for one-period-ahead forecasting, from among 24 other time series methods and using a variety of accuracy measures. Thus, regardless of the theoretical model for the process underlying the observed time series, simple exponential smoothing will often produce quite accurate forecasts.

Choosing the best value for the parameter  $\alpha$ : Gardner *et al.* in [23] concludes that it is best to estimate an optimum  $\alpha$  from the data, rather than to "guess" and set an artificially low value.

#### **2.4.5 Hybrid ARIMA with Neural Network Model**

Zang *et al.* in [27] presented the idea of combining Artificial Neural Network (ANN) and ARIMA model to Time series. The approximation of ARIMA models to complex

nonlinear process may not be adequate. Using ANN for linear problems does not always yield good result. Hence Zang *et al.* proposed hybrid methodology. It had two steps.

1. Where an ARIMA model was used to analyze the linear part of the problem.
2. Since ARIMA model cannot capture the nonlinear structure of data, residuals of linear problem will contains information about non linearity. Let  $e_t$  denote the residual at time  $t$  from linear model. By modeling residuals using ANNs, nonlinear relationship is discovered. With  $n$  input nodes the ANN model for residuals will be

$$e_t = f(e_{t-1}, e_{t-2}, e_{t-3} \dots, e_{t-n}) + \varepsilon_t \quad (2.5)$$

Where  $f$  is the nonlinear function determined by neural network,  $\varepsilon_t$  is the random error. In this model if  $f$  is not appropriate the error term is not necessarily random. Hence correct model identification is critical. Correct model identification in ANN is not easy and requires expertise and experience.

#### **2.4.6 Support Vector Machines**

Support Vector Machines (SVMs) have originally been used for classification purposes but their principles can be extended easily to the task of regression and time series prediction [28]. SVMs are linear learning machines which means that a linear function  $f(x) = wx + b$  is always used to solve the regression problem. These advantages of SVM from the specific formulation of a (convex) objective function with constraints, which is solved using Lagrange Multipliers and has the characteristics that:

- A global optimal solution exists which will be found,
- The result is a general solution avoiding overtraining,
- The solution is sparse and only a limited set of training points contribute to this solution.
- Nonlinear solutions can be calculated efficiently due to the usage of inner products.

## 2.5 Graph Concepts

A *labeled graph*  $G$  is a five element tuple  $G = \{V, E, \Sigma_V, \Sigma_E, l\}$  where  $V$  is a set of vertices and  $E \subseteq V \times V$  is a set of directed edges.  $\Sigma_V$  and  $\Sigma_E$  are the sets of vertex labels and edge labels respectively. The labeling function  $l$  defines the mappings  $V \rightarrow \Sigma_V$  and  $E \rightarrow \Sigma_E$ .

Two concepts from graph mining are incorporated in graph based framework for time series analysis which is graph matching and graph clustering. They are reviewed in subsequent section.

### 2.5.1 Graph Matching

The process of evaluating the similarity/proximity among graphs is referred the task of graph matching.

Graph matching can be done in two ways as described in [5, 8]

- Exact Graph Matching
- Inexact Graph Matching

In Exact Graph Matching [29] for a match to be successful, it is required that there should be strict correspondence between two graphs or at least amongst their sub-graphs. Inexact Graph Matching approach, allows matching in-between completely non-identical graphs. That is, in-exact matching algorithms are endowed with a certain tolerance to errors and noise, enabling them to detect similarities in a more general way than the exact matching approach. Therefore, inexact graph matching is also referred to as *error-tolerant graph matching*.

There are various ways to do Inexact graph matching such as Spectral Method [30] Graph Edit Distance [7, 31] and Graph kernels[32]. One of the most popular approach for graph matching is graph edit distance which we have used in our framework is reviewed.

**Graph Edit Distance:** Graph edit distance offers an intuitive way to integrate error-tolerance into the graph matching process and this technique of matching is applicable

to virtually all types of graphs. The key idea in this approach is to model structural variation by edit operations reflecting modifications in structure and labeling. A standard set of edit operations is given by insertions, deletions, and substitutions of both nodes and edges. Given two graphs, the source graph  $g_1$  and the target graph  $g_2$ , the idea of graph edit distance is to perform any of the operations of inserting, deleting and substituting on edges and vertex of  $g_2$  to transform into  $g_1$ . A sequence of edit operations  $ed_1, ed_2 \dots ed_k$  that transform  $g_1$  into  $g_2$  is called an edit path between  $g_1$  and  $g_2$ . The orientation/mapping with minimum cost gives the most similarity. Hence based on the metric of graph edit distance two graphs can be compared.

**Definition 2.2.1 (Graph Edit Distance)** Let  $g_1 = \{V_1, E_1, \Sigma_{V_1}, \Sigma_{E_1}, l_1\}$  be the source and  $g_2 = \{V_2, E_2, \Sigma_{V_2}, \Sigma_{E_2}, l_2\}$  the target graph. The graph edit distance between  $g_1$  and  $g_2$  is defined by

$$d(g_1, g_2) = \min_{(e_1, \dots, e_k) \in \gamma(g_1, g_2)} \sum_{i=1}^k c(e_i) \quad 2.6$$

where  $\gamma(g_1, g_2)$  denotes the set of edit paths transforming  $g_1$  into  $g_2$ , and  $c$  denotes the cost function measuring the strength  $c(e)$  of edit operation  $e$ .

### 2.5.2 Graph Clustering

In Graph clustering attempts to cluster the different graphs based on overall structural behavior. In this case, we have a (possibly large) number of graphs which need to be clustered based on their underlying structural behavior. The major challenges in graph clustering algorithm as compared to many of the conventional algorithms discussed in [33] are:

1. Most of the underlying classical algorithms typically use some form of distance function in order to measure similarity. Therefore, we need appropriate measures in order to define similarity (or distances) between structural objects.
2. Many of the classical algorithms (such as k-means) use *representative objects* such as centroids in critical intermediate steps. While this is straightforward in the case of multi-dimensional objects, it is much more challenging in the case

of graph objects. Therefore, appropriate methods need to be designed in order to create representative objects.

As discussed in [34, 35]. The graph edit distance can be used to measure the similarity between graphs. Using technique of Graph embedding in vector space the graph is mapped to vector space. In vector space the mean or median can be computed easily which forms the cluster prototype. As discussed in [36] the cluster prototype can be mapped from vector space to graph space

## **2.6 Research Gaps**

- Traditionally, all of the existing time series prediction methods use static modeling that is they do not consider about change in underlying pattern. However, this is not the case as new patterns evolve in time series over periods of time and incorporating changes in existing methods is difficult. Hence, there is a scope for developing new time series prediction technique to deal with evolving patterns in time series.
- Most of the existing research is based on parametric method. Such methods require some prior knowledge about the data. But, in real world, we may not have prior information about the data. Hence, there is a need to work on non-parametric methods for time series prediction.

## Chapter 3

### Proposed Framework for Time Series Prediction

We consider with the related issues of Time series prediction which uses mainly statistical techniques which require domain knowledge and are complex in dealing with evolving patterns in time series. We propose a model for time series prediction, which is built to address the major issue of predicting the future values, as new patterns evolve in time series over periods of time. The proposed framework uses techniques derived from graph mining. It is easy to use and consists of easy learning strategy. The major steps in the proposed framework, for graph based time series prediction are as shown in Figure 3.1.

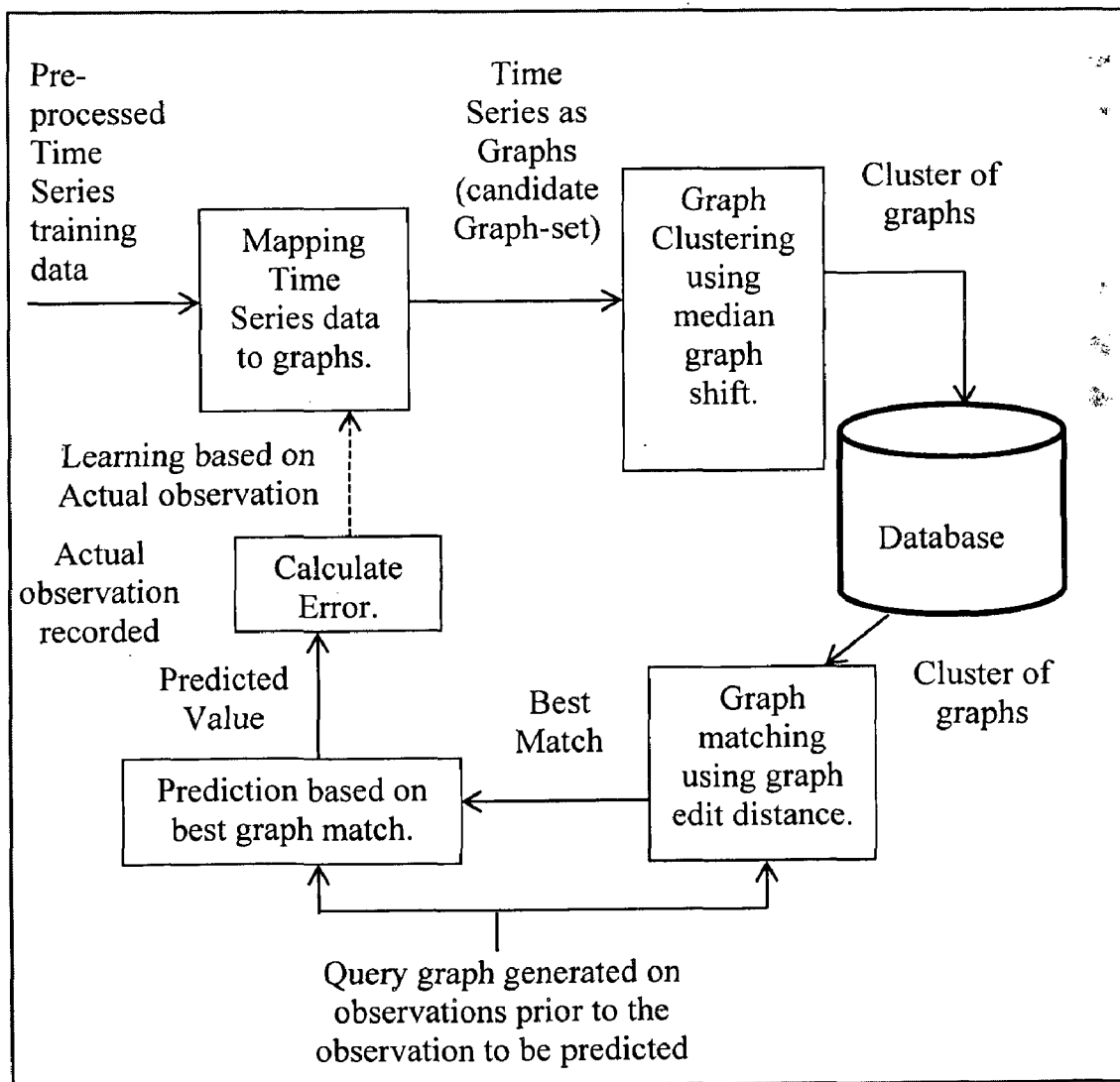


Figure 3.1 Graph Based Framework for Time series Prediction

## 3.1 Major steps in Graph Based Framework for Time series

### Prediction

The major steps in Graph based framework for time series prediction are as shown in Figure 3.1 are:

- Mapping Time series Data as Graphs.
- Graph Clustering.
- Database.
- Graph Matching.
- Prediction based on best graph match.
- Calculate Error and Learning based on Actual recorded value.

These steps are discussed in detail, describing their functionality and their need in the framework in the subsequent section.

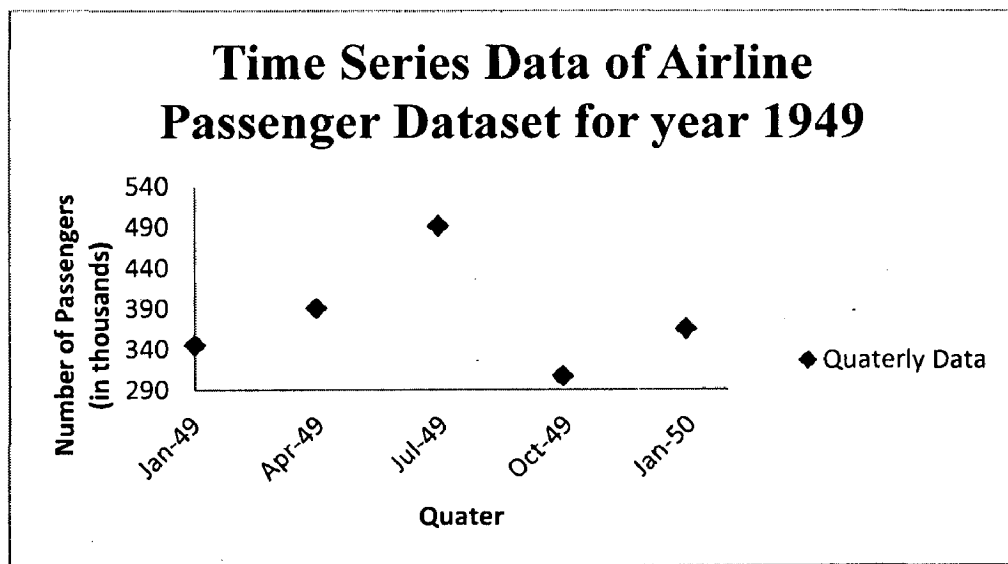
#### 3.1.1 Mapping Time Series Data as Graphs

The graph on time series is created to capture the pattern in time series data. The graph on time series is defined as a labeled graph, a *labeled graph*  $G$  which is a five element tuple  $G = \{V, E, \Sigma_V, \Sigma_E, l\}$  where  $V$  is a set of vertices and  $E \subseteq V \times V$  is a set of directed edges.  $\Sigma_V$  and  $\Sigma_E$  are the sets of vertex labels and edge labels respectively. The labeling function  $l$  defines the mappings  $V \rightarrow \Sigma_V$  and  $E \rightarrow \Sigma_E$ .

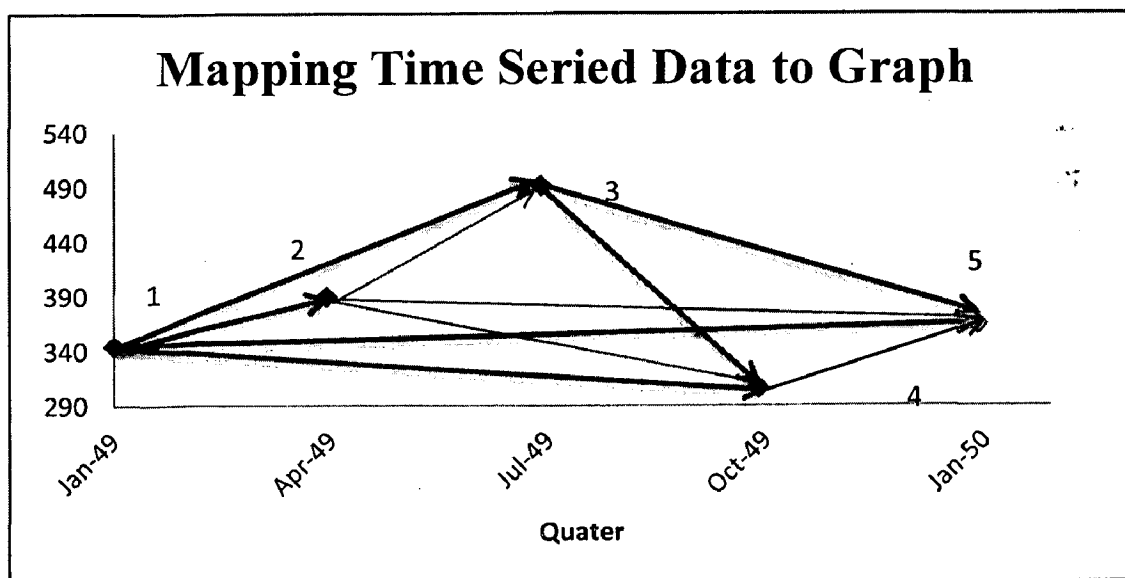
To model time series data as graph, each observation is represented as vertex and variation between observations are represented as edges. The number of vertices in graph is equal to time interval over which a pattern has to be stored. The edges are created to take into account the effect of each observation on other. Since the past values will affect the future values, hence the edges are directed from observation representing current vertex to vertices corresponding to successive observations in time series data. Edge measures the change in angle with the horizontal. In Table 3.1 the time series of airline passenger dataset is summarized quarterly for year 1949 and deseasonalized time series as generated as discussed in Section 2.2.1. Figure 3.2 represents the time series data and Figure 3.3 represents the corresponding graph; which is generated on the time series data.

**Table 3.1 Monthly Data of Airline Passenger Time Series Dataset (APTS) is Summarized Quarterly for Year 1949 and 1950.**

Year	Quarter	Number of Passenger	Seasonal Factor	Deseasonalized Time series
1949	Q1	362	0.95	344.85
1949	Q2	385	1.013	390.06
1949	Q3	432	1.14	491.11
1949	Q4	341	0.89	306.00
	Average	380		
1950	Q1	382	0.95	363.91



**Figure 3.2 Representing Deseasonalized Time Series Data of Airline Passenger Data Set for year 1949 with respect to Time.**



**Figure 3.3 Representing Time Series Data as Graph.**



To \ From	1	2	3	4	5
1	0	1.55	1.56	-1.49	1.36
2	0	0	1.56	-1.55	-1.46
3	0	0	0	-1.57	-1.56
4	0	0	0	0	1.55
5	0	0	0	0	0

**Figure 3.4 Adjacency Matrix Representation of Graph for Figure 3.3.**

In Figure 3.3 vertices are labeled with numerals and edges store change in angle between vertices with horizontal in radians. In Figure 3.3 the edge angles are not marked only their direction is shown. Figure 3.4 shows the Adjacency Matrix Representation described in [37], of generated Graph on time series.

The graphs generated are classified into two categories depending upon the type of observation they represent. We define a notion of pattern period. A pattern is a noticeable structure in time series and pattern period is the time it takes to form a pattern. It is a user specified parameter and represents the time interval after which a pattern in time series is formed. The observations which form a pattern denote vertices of *pattern set*. Predictor observations are observations that are added to pattern, knowingly in order to make predictions. The observations which are added to pattern are denoted as vertices of *predictor set*. If the graph has vertices corresponding to observations that form only the pattern it is called '*past-pattern graph*'. If the graph has vertices corresponding to pattern and predictor observations then graph is called '*knowledge graph*'. Predictor observations form the basis to make predictions in the model e.g. Suppose a pattern is assumed to be over a year, hence in Figure 3.3 Vertex  $\{1, 2, 3, 4\} \in \text{Pattern Vertex set}$  and Vertex  $\{5\} \in \text{Predictor Vertex set}$  and graph in Figure 3.4 represents a '*knowledge graph*'.

### 3.1.2 Graph Clustering

Graph Clustering in the proposed framework is an optional step. Graph clustering is done in the framework to group similar graphs together so that the model has the following advantages.

1. The cluster prototype can be more general and represent pattern of multiple graphs.
2. Search time decreases drastically as instead of searching for similarity across all graphs in database, only the similarity with cluster prototype needs to be computed at time of prediction.
3. Adds the model the capability to handle outlier data, since smoothing of data due to averaging.

The criteria for clustering two graphs in the same cluster is edges from pattern vertex to that of predictor vertex are observed in both graphs, if the percentage difference in outgoing edges between both graphs observed from each vertex is within a threshold  $t$  the two graphs are clustered and cluster prototype is updated accordingly. Cluster prototype is the calculated as mean of two graphs using technique proposed by [34].

### 3.1.3 Database

Database stores the Cluster of '*Knowledge graphs*'. Knowledge graphs have vertices corresponding to pattern and predictor observations. The Graph database is partitioned into  $d$ -set, where  $d$  is the number of vertices in pattern. Each generated knowledge graph is added to cluster corresponding to the predictor vertex information it represents. The graphs are partitioned with the motivation to ease the search since while making prediction for a particular time interval the model will query all clusters of graphs that have been observed in that period, while searching for most similar graph in graph database. Since the graphs are already portioned, search space is reduced.

### 3.1.4 Graph Matching

Graph Matching is required in the framework to quantify the similarity between graphs generated on Time series data. Graph Matching techniques are discussed in Section 2.4.1, where existing graph Matching techniques have been discussed. Graph Matching is done while making the prediction in time series. Graph matching is done between '*past-pattern graph*' which is generated using time series observations prior to the observation to be predicted and with each of the cluster prototypes considering only vertices in pattern set between both graphs. We have used the Graph Edit

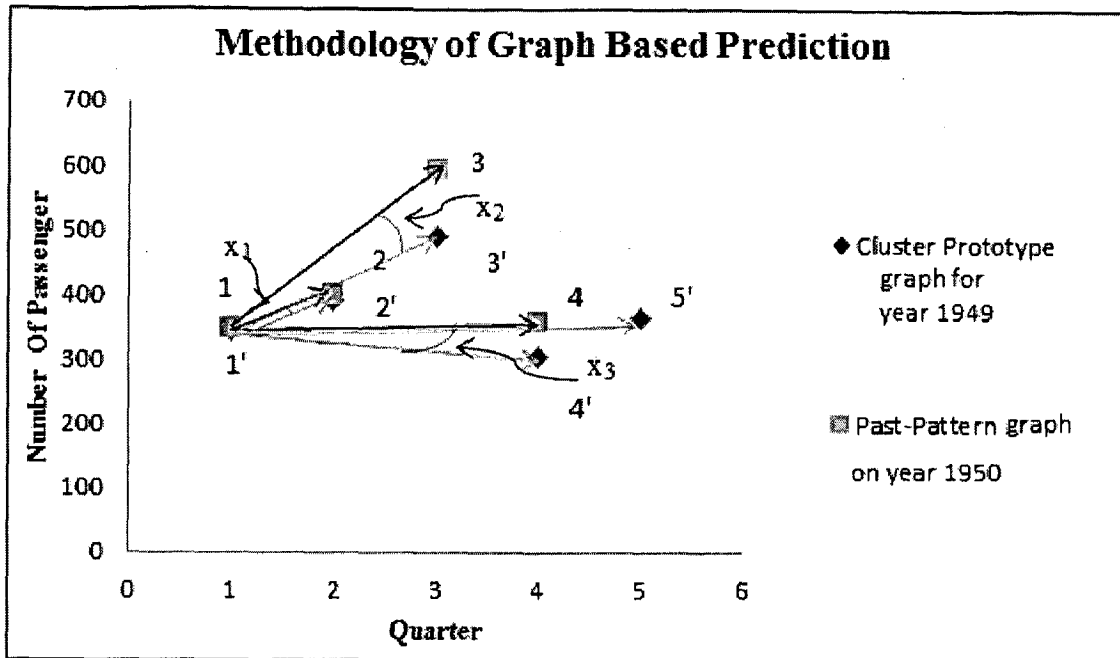
Distance as metric to measure similarity among graphs as discussed in section 2.5.1 and [7], since the graphs on Time series are planner in nature.

### 3.1.5 Prediction Based on Best Graph Match

For predicting the values in time series we require a graph  $g_1$  '*past-pattern graph*' which is generated using time series observations prior to observation to be predicted and most similar *cluster prototype*  $g_2$  is selected using graph matching as discussed in Section 3.1.3.

- Now  $g_1$  is having only the vertices corresponding to observations which form pattern and  $g_2$  is having vertices corresponding to both pattern and predictor observations.
- To make prediction  $g_1$  uses the pattern knowledge of  $g_2$ , which is stored in form of graph.
- Every vertex of  $g_1$  takes into account the difference of all the out-going edges between itself and its corresponding vertex in  $g_2$  in a weighted average manner (that is edge difference of vertices closer to the vertex in the predictor set are given more weight) and then adding this difference to the edge between the corresponding vertex in  $g_2$  pattern set and predictor set. This is the angle between vertex in  $g_1$  and the predicted observation.
- In this manner each vertex of  $g_1$  predicts the angle between itself and the predicted value, and the predicted value is the average of value predicted by each vertex. The edge difference is calculated in weighted average manner, which is a technique to incorporate exponential smoothing in graph based framework for time series prediction.

As shown in Figure 3.5 the methodology for predicting the future value in the time series for vertex 1 in graph  $g_1$  (*Past-pattern graph* for year 1950) which uses  $g_2$  (stored *Cluster prototype graph* for year 1949 in graph Database). In Figure 3.5 edges are shown from vertex 1 in  $g_1$  and vertex 1' in  $g_2$  for simplicity. Vertex 1 in  $g_1$  predicts the angle between itself and vertex 5 (observation to be predicted). Vertex 1 of  $g_1$  calculates the difference in angle between all outgoing edges between itself and its corresponding vertex in  $g_2$  considering only pattern set.



**Figure 3.5 Representing Graph on Time Series Data, showing only Edges from Vertex 1 and 1' in Past-pattern Graph for year 1950 and Cluster Prototype Graph for year 1949.**

Let  $x_1$  be the difference in angle between vertex 1 and vertex 2 and 2' as shown in Figure 3.5. Similarly  $x_2$  is the difference between edges from vertex 1 and vertex 3 and 3' and  $x_3$  is the difference between edges from vertex 4 and 4'. Weighted average of difference in angle is calculated as  $(x_1 + 2 \times x_2 + 3 \times x_3) / 6$ . Vertex 1 adds this weighted average of difference to edge 1' and 5'. This is the predicted angle between 1 and 5 where 5 is the vertex predicted by vertex 1. Similarly each node predicts the angle between itself and vertex in the predictor and the predicted value is the average of all these values. In this way time-series is predicted using the graph based framework.

### 3.1.6 Calculate Error and Learning Based on Actual Recorded Value

The error at each observation is calculated using the Eqn. 3.1. The Learning approach in graph based framework is 'learning over experience'. As the actual observation is recorded, Knowledge graph is generated to capture the pattern information associated with this observation and it is added to appropriate cluster using method as discussed in Section 3.1.2 in graph database.

$$\text{Percentage Error} = \frac{|(\text{predicted} - \text{Actual})|}{\text{Actual}} \times 100 \quad (3.1)$$

Graph clustering is useful as number of graphs in database does not necessarily increase linearly with every observation as similar graphs will be clustered rather on different observed patterns.

### **3.2 Applying the Graph Based Framework in Conjunction with Existing Time Series Concepts**

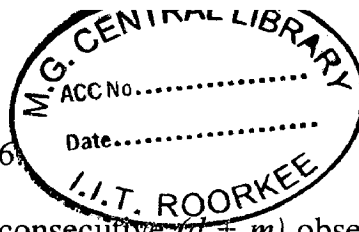
As discussed in section 2.2, the time series variable  $Y$  at time  $t$  (that is  $Y_t$ ) can be modeled as product of Trend component at time  $t$  ( $T_t$ ), Cyclic component at time  $t$  ( $C_t$ ), Seasonal factor at time  $t$  ( $S_t$ ) and Random component at time  $t$  ( $I_t$ ) that is  $Y_t = T_t \times C_t \times S_t \times I_t$ . The time series can be deseasonalized using the concept of Moving average as discussed in section 2.3.2. Thus deseasonalized time series  $Y_t$  can be computed which has trend component and cyclic component only that is  $Y_t = T_t \times C_t$ . Graphs generated on deseasonalized time series will represent pattern which is free from seasonal variation. Hence in Graph based framework for time series analysis the deseasonalized time series can also be used.

The concept of exponential smoothing is incorporated in prediction step as discussed in Section 3.1.4. The concept of exponential smoothing while predicting gives more weight to recent observations than older observations which is discussed in detail in section 2.2.2. The same concept is incorporated at prediction step where more weight is given to edges towards vertex closer to the predictor. Thus taking difference of edges in a weighted manner is an approach to incorporate exponential smoothing in graph based framework.

Thus existing time series concepts can be easily added to graph based framework for time series prediction.

### **3.3 Flow Diagram of the Graph Based Framework for Time series Prediction**

The flow diagram of complete approach is broken in two phases as shown in Figure 3.6 and Figure 3.7. The first phase is training phase, where model learns and the second phase is prediction phase, where model predicts the time series.



**Training Step** as shown in Figure 3.6.

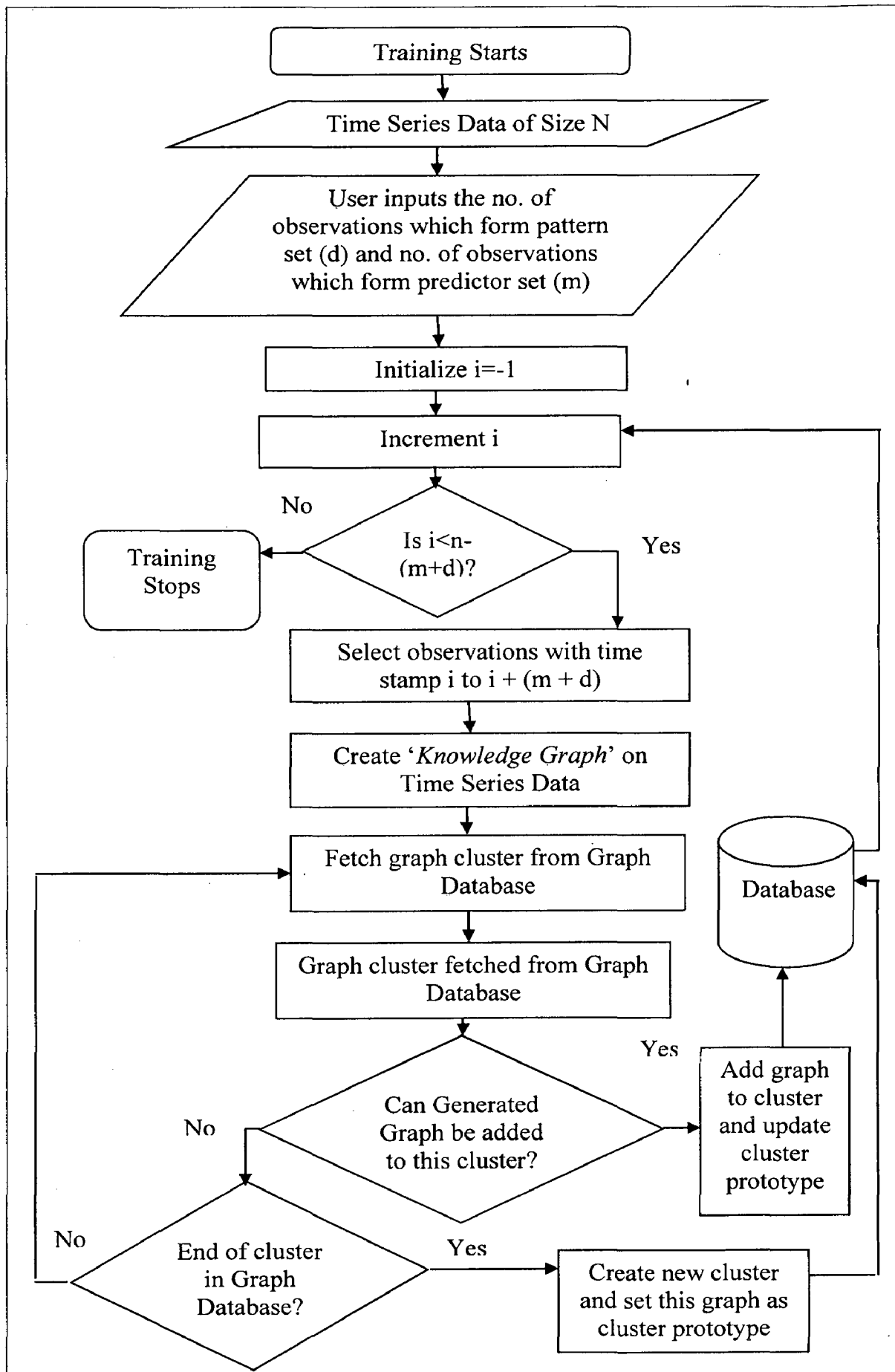
1. The model takes each of the consecutive  $(d + m)$  observations, where  $d$  is the number of observation which represents a pattern and  $m$  is the observation in the predictor set.
2. *Knowledge graph* is generated on each of the  $(d + m)$  observations.
3. Generated graph is added to the  $k^{th}$  set in 'graph database', where  $k$  is the  $(m \bmod (d))$  in the pattern set. 'Graph Database' is having cluster of graphs, where each cluster prototype represents mean of all graphs in its cluster.
4. All existing cluster prototype graphs are scanned to find the cluster to which knowledge graph generated in step 2 can be added using the criteria and concept defined in Section 3.1.2. Once such a cluster is found the knowledge graph is added and cluster prototype is updated.
5. If none of the cluster prototypes meet the defined criteria new graph cluster is created and Knowledge graph generated at step 2 is made cluster prototype.

**Prediction Step** as shown in Figure 3.7:

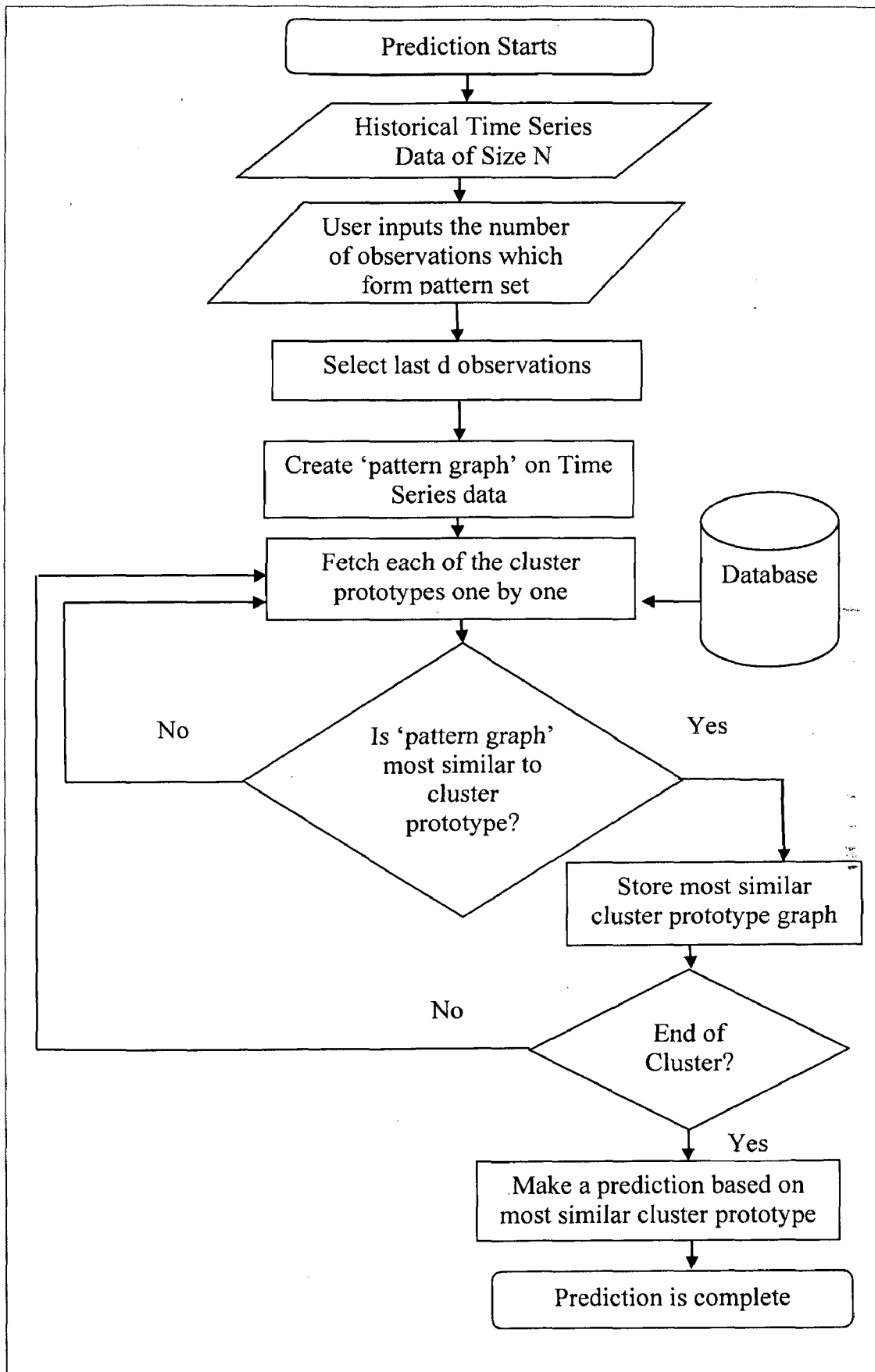
1. The model takes last  $d$  observations, where  $d$  is the number of observation which represents a pattern.
2. *Past-pattern graph* is generated on  $d$  observations.
3. Form graph database each *cluster prototype* is scanned to find the cluster prototype that is most similar to *past-pattern graph*. Graph edit distance is used as metric to calculate similarity amongst graphs, considering only vertices that are in pattern set in cluster prototype.
4. Make prediction using proposed approach as discussed in section 3.1.5 which makes the prediction using graph representation of time series.

Note

- The *Database* shown in Figure 3.7 is populated in Figure 3.6.
- For learning based on Actual observation again training routine is applied but only for new generated knowledge graph.



**Figure 3.6 Flow Diagram of the Training Phase in Graph Based Framework for Time series Prediction.**



**Figure 3.7 Flow Diagram of the Prediction Phase in Graph Based Framework for Time series Prediction.**





## Chapter 4

### **Implementation of the Proposed Framework**

---

The project has been implemented in Java programming language using the NetBeans 6.9.1 Integrated Development Environment (IDE). The NetBeans IDE is written in Java and runs everywhere where a JVM (Java Virtual Machine) is installed, including Windows, Mac OS, Linux, and Solaris. A JDK (Java Development Kit) is required for Java development functionality. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. The IDE is available as a free download from internet.

This Java based Time series Prediction application which uses graph based framework has been implemented on Windows XP Operating System. It consists of six important modules which are described in the succeeding subsections and the functions that are used to implement it. Basic Modules in the framework are:

- Mapping of time series data as graphs.
- Graph Clustering.
- Database.
- Graph Matching.
- Predicting time series using graph representation.
- Calculate error and learning based on Actual Recorded value.

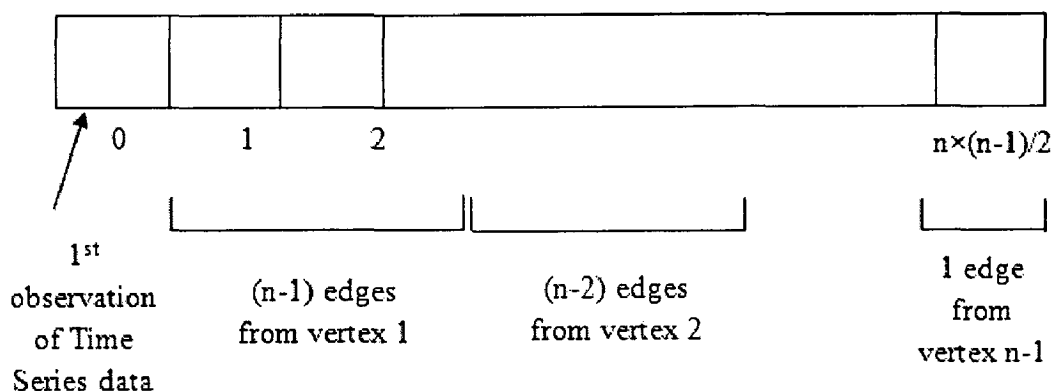
#### **4.1 Modules in Graph Based Framework**

The modules in graph based framework are:

##### **4.1.1 Generation of Graph on Time Series Data**

As discussed in section 3.1.1, the approach for creating graph on time series. Fig 3.4 shows the adjacency matrix representation of graph created in Figure 3.3. Adjacency matrix is  $n \times n$  two-Dimensional array, where  $n$  is the number of vertices. We notice

that in the graphs generated on time series data the entries are only in the upper triangular matrix.



**Figure 4.1 One Dimensional Array Representation of Upper Triangular Matrix.**

**Algorithm: Create\_Graph** Creates a graph by setting values in *g* (class member which is one dimensional array, implementation of upper triangular matrix) using the *Data* array which contains Time series observations.

Input:

- Reference to an array *Data* that has recorded time series observations.
- *Start*- Starting index of *Data* array over which graph has to be generated.
- *End*- Last index of *Data* array over which graph has to be generated.

Output: class member *g* is having graph on time series.

Method:

1. At location 0 in *g* set `data[start]`
2. for (`i=start` and `L=0`; Loop until `i<end`; Increment `i` & `L`)
3. for (initialize `j=i + 1` & `M=L + 1`; Loop until `j<end`; Increment `j` & `M`)
4. `g.setElementsIJ (L, M, Math.atan ((data[j] - data[i]) / (10 * (M - L))))`; // angle between vertex `j` & `i` is computed in radians. Division by 10 to have greater angular range.

**Figure 4.2 Algorithm of Create Graph method, which Maps Time Series Data on Graphs**

Hence we map this two dimensional matrix to one dimensional array of size  $n \times (n - 1) / 2 + 1$  as shown in Figure 4.1. At location 0 the 1<sup>st</sup> observation of time series data used in graph generation is stored and in subsequent location edges of vertices are stored. Hence we store the upper triangular matrix only and use mapping function to map elements of *matrix* ( $i, j$ ), (where  $i$  is the row specifier and  $j$  is the column specifier) to one dimensional array. Thus using mapping of triangular matrix to one-dimensional array for internal representation of graph requires use  $n \times (n - 1) / 2 + 1$  space only instead of  $n^2$  required with adjacency matrix representation of graph.

In the proposed work we have used pattern set over a year and an observation in predictor set. Thus, the *knowledge graph* has 13 vertices and the *past-pattern graph* has 12 vertices. Algorithm of *create graph* as shown in Figure 4.2. It illustrates the procedure to Creates Graph on Time series Data. We have used trigonometric concepts to find the angle which represents edge between vertex  $i$  and vertex  $j$ . The *setElements()* is a mapping function that maps two dimensional matrix having only upper triangular matrix to one dimensional array.

#### 4.1.2 Graph Clustering

The existing graph clustering techniques have been discussed in section 2.4.2 and in section 3.1.2 the graph clustering criteria for time series data and its need in proposed framework is presented. The threshold parameter is defined as  $t=.05$ . To implement graph clustering, three functions are implemented and as shown in Table 4.1 their prototype and in which class they are implemented is summarized. The functionality of each function is discussed below:

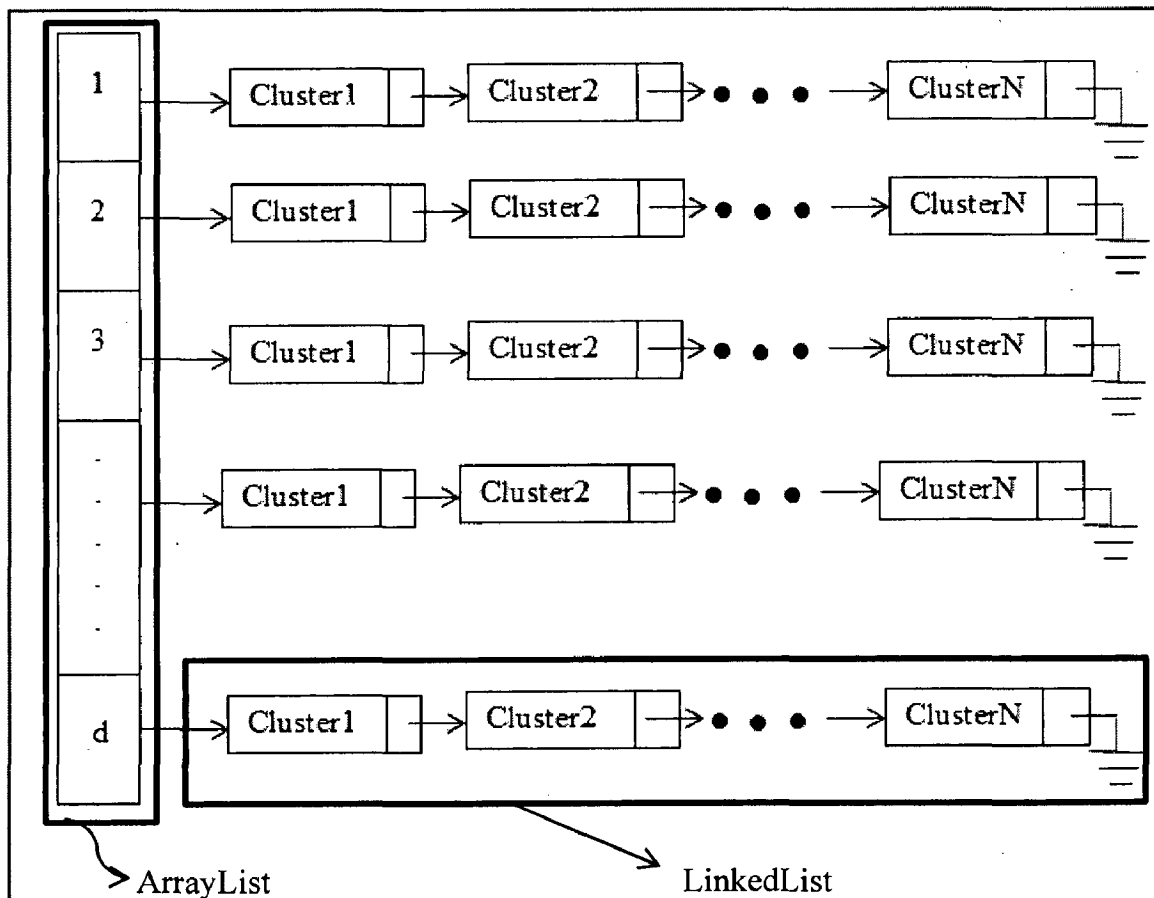
1. *Function 1*: It checks each of the cluster prototypes and finds that can generated graph can be added to this cluster using criteria defined in section 3.1.2, if it can be added it return true else false is returned.
2. *Function 2*: It adds graph to selected cluster and updates its cluster prototype.
3. *Function 3*: It traverses each of the existing clusters in the *Graph database* and finds that, can generated *Knowledge-graph* be added to the particular cluster using *function 1*. If *function 1* returns true, graph is added to cluster using *function 2* else other cluster is checked with *function 1* until end of cluster is

reached. If graph cannot be added to any of the existing clusters new cluster is created and this graph is set as its prototype.

**Table 4.1 Function Name and Class in which Java Code Implemented**

Function	Implemented in Class	Function Name in code and parameters
Function 1	Class Utility	<i>public static boolean canAddedToThisCluster(Cluster Cl, GraphOnTSDData g, int d)</i>
Function 2	Class Cluster	<i>public void ClusterAdd(GraphOnTSDData obj)</i>
Function 3	Class Utility	<i>public static void ScanClusterToAddG(LinkedList&lt;Cluster&gt; ll, GraphOnTSDData gg, GraphDb db, int i)</i>

### 4.1.3 Database



**Figure 4.3 Memory Representation of Database.**

As discussed in section 3.1.3 graph database stores the cluster of 'knowledge graphs'. Graph database is partitioned into  $d$ -set, where  $d$  is the number of observation that

form a pattern. To implement this graph database we have used a generic ArrayList which has reference to object of type generic as shown in Figure 4.3.

LinkedList. LinkedList and ArrayList are classes that are defined java.util package. Size of ArrayList is equal to the number of observations which represent a pattern. We have kept the size of ArrayList to be 12 (corresponding to number of months). Each ArrayList contains all the clusters of patterns observed in that time interval. The clusters are in LinkList. The LinkList contains list of type cluster. So each cluster contains the prototype of the cluster and all the graphs that are in the cluster.

#### 4.1.4 Graph Matching

The existing graph matching techniques are discussed in section 2.4.1. Role of graph matching in proposed framework is presented in section 3.1.4. In the proposed framework for graph matching, graph edit distance technique is used to compute proximity between graphs.

The key idea of Graph-edit Distance approach is to model structural variation in graphs by edit operations reflecting modifications in structure and labeling. While calculating graph edit distance for time series Graph, only substitutions of edges (change in edges to make graph to be matched similar to the reference graph) and a summation of cost incurred with each edit operation is calculated. The graph with least edit cost is most similar & selected as a graph that will form the basis, of the prediction.

Code to implement graph edit distance is in *utility* class and the function has prototype *public static double Graphdiff (GraphDS tobematched, GraphDS matchedAgainst, int d)*. It is a static method that returns the edit cost to transform one graph to another and it takes three arguments, the graph to be matched, the graph that is match against the graph that is reference (base graph) and the number of vertices to be considered while calculating graph edit distance (this is the number of vertices in pattern graph). In our implementation we have assumed pattern to be over a year, hence number of vertices in pattern is 12.

While doing edge substitutions to transform *tobemached* to *matchedAgainst* there are three types of edge substitutions as mentioned below:

1. *Normal substitution*, that is substitution of edge between vertex i and vertex j.
2. *Forward substitution* that is when we transform edge between vertex i and vertex j then all edges going out of vertex j also get transformed.
3. *Backward substitution* that is when we transform edge from vertex i to vertex j then all edges towards vertex i get transformed.

We have given different weights to cost associated with different type of edge substitution. *Normal substitution* is given weight 1, *Forward substitution* is given weight 1.25 and *backward substitution* is given weight 1.5.

#### **4.1.5 Predicting Time Series using Graph Representation**

The approach for predicting the time series using graph representation of time series data has already been discussed in section 3.1.5. In Figure 3.5 the methodology for graph based prediction is illustrated for vertex 1. The algorithm for predicting time series using graph based framework is shown in Figure 4.4.

As shown in Figure 4.4 each vertex in current graph uses two main steps to make prediction

1. Each vertex in Current graph observes the difference between all outgoing edges with respect to its corresponding vertex in *baseDb* graph and the net difference is calculated using weighted average as explained in section 3.1.4.
2. Each Node in Current graph uses the edge between predictor using graph database and add the difference obtained in step 1.

Thus in this way each vertex predicts the angle between itself and future observation. The predicted observation is average of value predicted by each vertex as shown in Figure 4.3.

**Algorithm: PredictNext:** Predicts the future value in time series

Input:

- *Current*: Graph having only pattern vertices.
- *baseDb*: Graph having pattern and predictor vertices.
- *d*: Number of vertices in pattern.

Output: The predicted value using graph based framework.

Auxiliary storage: double array `angles[d]` and `predicted[d]`.

Method:

1. for each edge from source *i*
  2.     for each edge to destination *j*
  3.          $\text{diff} = (j - i) \times \text{difference of angle between edges between } i \text{ and } j \text{ in graph Current and baseDb}$  //  $(j - i)$  multiplied to assign weight.
  4.     Store this difference in `angle[i]`;
- //Difference of all edges going from vertex *i* between Current and baseDb is in `angle[i]`.
6.     Do division step of computing weighted average of `angle[i]`
  7.     In `angle[i]` add the angle between vertex *i* and predictor set using baseDb.
  8. `angles[d-1]=(baseDb.getElements(i,d));` //for last vertex in Current graph, the baseDb gives the angle
  9. for each vertex *i* in Current graph
  10.     predicts the future observation using `angle[i]` and store it in `angle[i]`.
  11. Call function which returns the predicted value as average value predicted by every node (taking average of `angle[i]`).

**Figure 4.4 Algorithm to Predict Future observation in Time Series using Graph Based Framework.**



### **4.1.6 Learning Approach in Graph Based Framework**

As mentioned in Section 3.1.6, the Learning approach in graph based framework is learning over experience. As a new observation is recorded 'knowledge graph' corresponding to that observation is generated by main class. Generated 'knowledge graph' on predictor observation  $i$  is added to  $mod(i, d)$  ArrayList of graph database, first searching if it can be added to any of the cluster and if it cannot be added to any of the clusters, a new cluster is created and this graph is set as its prototype using graph clustering module as described in Section 4.1.6.

## **4.2 The Experiment dataset**

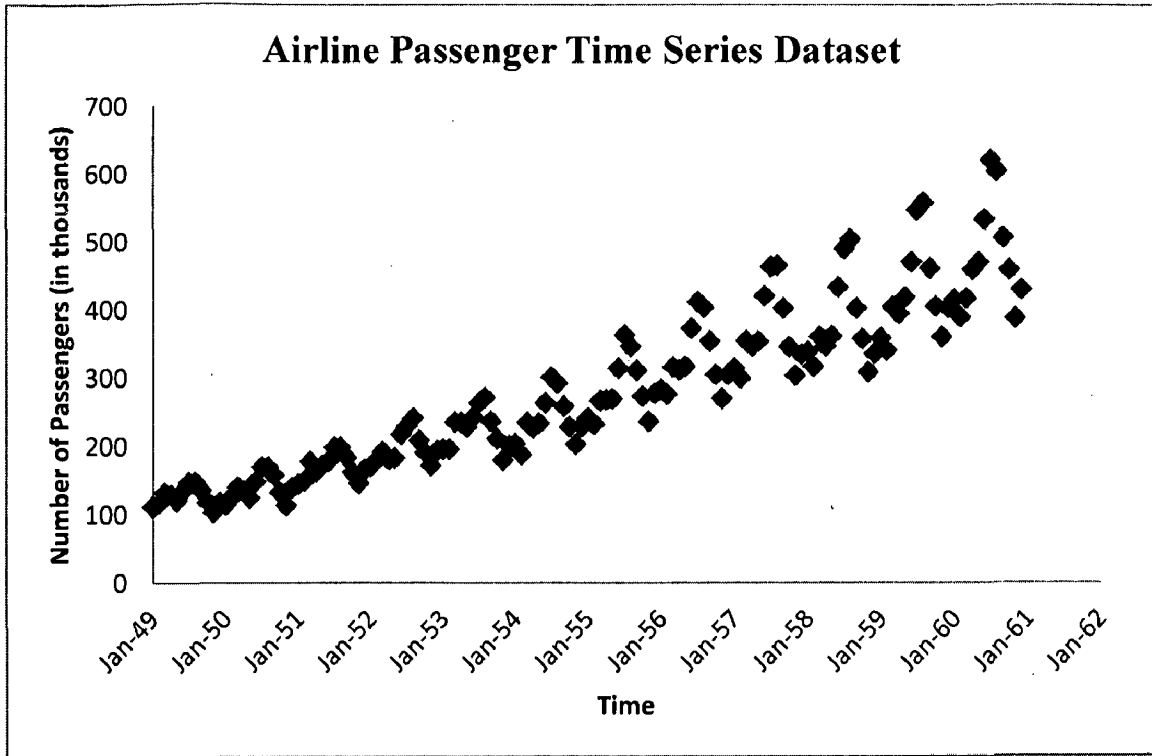
In this section we have presented the brief introduction about the dataset that have been used for validating the proposed framework.

### **4.2.1 Airline Passenger Time series (APTS) Dataset**

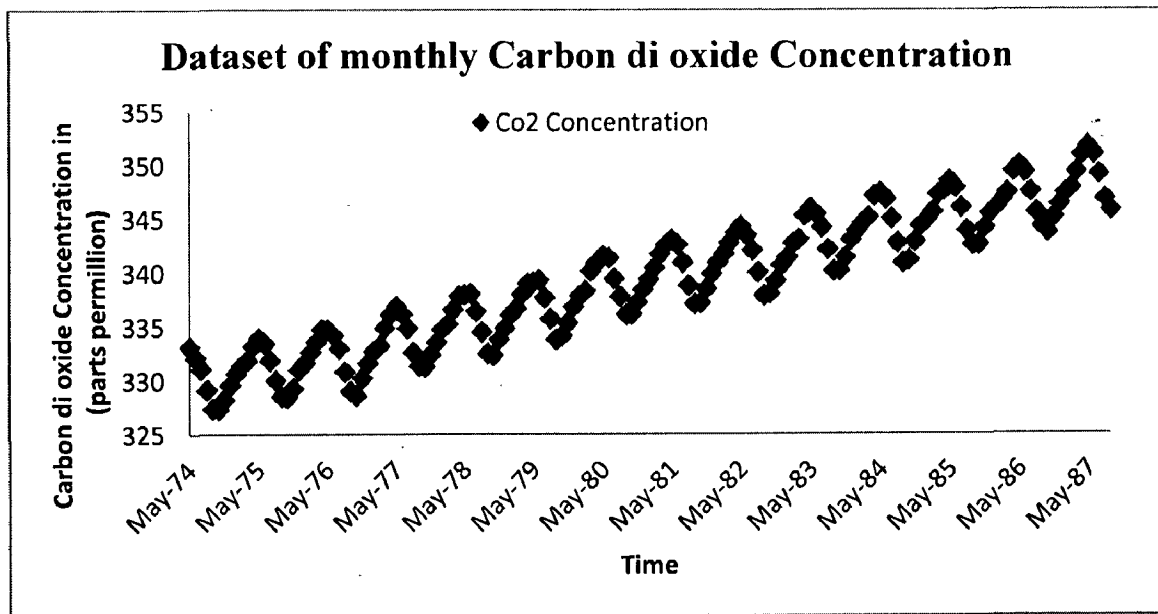
The Graph Based Time series prediction approach was applied on the airline passenger data set, which was used in [15]. It represents the number of airline passengers in thousands who travelled between January 1949 and December 1960 by air on a monthly basis. In Figure 4.5 the Data is represented in chart to show the variation and pattern in data with over time. From Figure 4.5 it is observed that data has seasonal variation.

### **4.2.2 Dataset of Monthly Carbon di oxide Concentration**

This data set contains selected monthly mean Carbon di oxide concentrations at the Mauna Loa Observatory from May, 1974 to September, 1987. The Carbon di oxide concentrations were measured in parts per million by the continuous infrared analyzer of the Geophysical Monitoring for Climatic Change division of NOAA's Air Resources Laboratory. The Dataset is given by [38]. In Figure 4.6 the Data is represented in chart to show the variation and pattern in data with over time. From Figure 4.6 it is observed that data has very less seasonal variation.



**Figure 4.5 Representation of Airline Passenger Time series (APTS) Dataset with respect to Time.**



**Figure 4.6 Representation of Data Set of Monthly Carbon Di Oxide Concentrations in Parts Per Million with Respect to Time.**



## Chapter 5

### Results and Discussion

---

The implementation of the graph based time series prediction application is done in java and tested with Airline Passenger Time series (APTS) dataset, Dataset of Monthly Carbon di oxide as described in the Section 4.3. The results are reported using proposed framework from four variants of the proposed framework:

- **Scenario 1:** Graph based framework on seasonal Time series Data without clustering.
- **Scenario 2:** Graph based framework on Deseasonalized Time series Data without clustering.
- **Scenario 3:** Graph based framework on seasonal Time series Data with clustering.
- **Scenario 4:** Graph based framework on Deseasonalized Time series with clustering.

Results are reported in subsequent section using variants of the proposed framework above mentioned on both the datasets.

#### 5.1 Results for Airline Passenger Time Series dataset

The dataset has 144 observations representing the number of passengers in thousands that took the airline service on a monthly basis from 1949 to 1960. The data was partitioned into two parts the training data and the testing data. The training data consists of 24 observations and the testing data consists of 120 observations. As model predicts the observation for a particular time interval it is assumed that actual observation would have been recorded and learning in framework is done before predicting the next observation of the successive time interval.

The results are summarized as:

- In Figure 5.1, graph represents predicted values which were calculated using scenario 1 and actual observations recorded. Figure 5.2 represents the percentage error observed at each prediction using this (scenario 1) technique.
- In Figure 5.3, graph represents predicted values which were calculated using scenario 2 and actual observations recorded. Figure 5.4 represents the percentage error at each prediction using this (scenario 2) technique.
- In Figure 5.5, graph represents predicted values which were calculated using scenario 3 and actual observations recorded. Figure 5.6 represents the percentage error at each prediction using this (scenario 3) technique.
- In Figure 5.7, graph represents predicted values which were calculated using scenario 4 and actual observations recorded. Figure 5.8 represents the percentage error at each prediction using this (scenario 4) technique.

The average percentage error, reason of error and error pattern in prediction corresponding to all the scenarios depicted in Figure 5.2, 5.4, 5.6 and 5.8 is summarized in Table 5.1.

**Table 5.1 Average Percentage Error on Airline Passenger Dataset and Error Pattern in Different Scenarios and Reason for Error.**

SCENARIO	Average percentage Error	Reason for error	Error pattern
1	7.05	The error is since the pattern in time series is evolving with time and since the seasonal fluctuations are also their adding to the error.	Error is decreasing.
2	5.91	The error is reduced due to the deseasonalizing the time series.	Error remains almost constant
3	6.63	The error has reduced as compared to scenario 1 due to clustering. Error greater than scenario 2, because of seasonality in data,	Error is showing slight growth.
4	5.81	The error in this case is least because of clustering and deseasonalizing.	Error is decreasing.

We applied least Square method to **seasonal time series**, where polynomial of degree 3 was fitted using MATLAB using the function *polyfit*. The training data was of 24 observations and 120 observations were predicted and learning on previous

observation was done before predicting next observation. The predictions were made with average percentage error rate was 10.4 and when Least square method was applied to deseasonalized time series percentage with similar scenario, percentage error rate was 8.2. Clearly Graph based framework for time series prediction performs better.

### SCENARIO 1

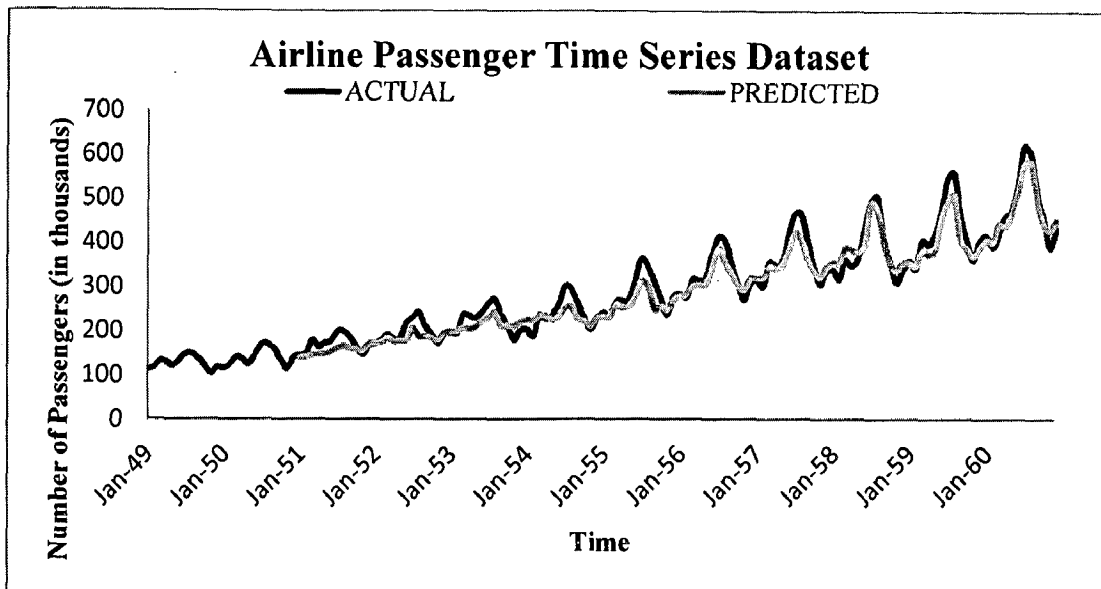


Figure 5.1 Represents Actual and Predicted Observations with respect to Time. Predictions are Made Using Graph Based Framework on Time Series Data with Seasonal Fluctuations Without Clustering.

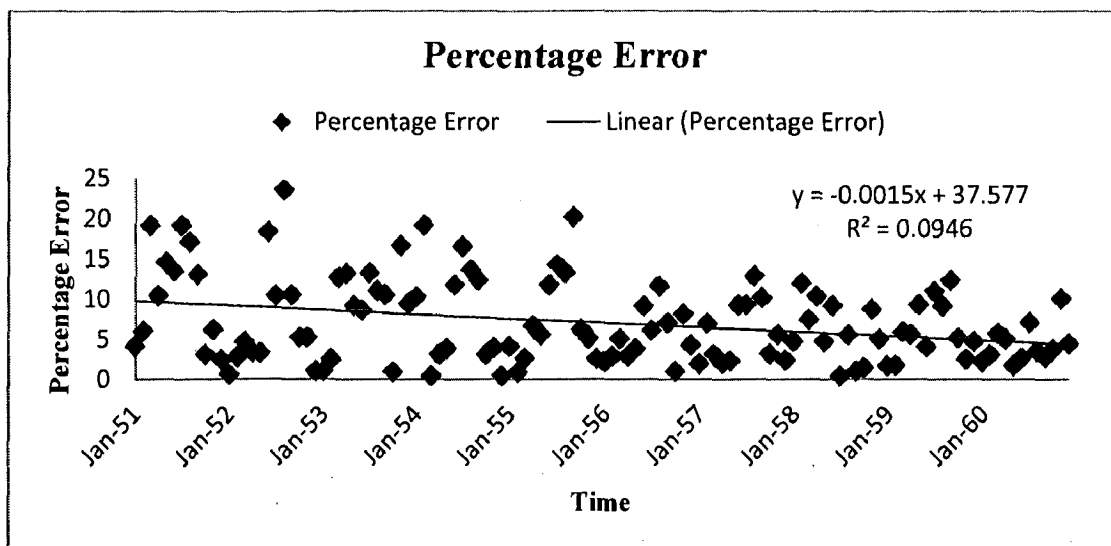


Figure 5.2 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.1. Average Percentage Error is= 7.05.

## SCENARIO 2

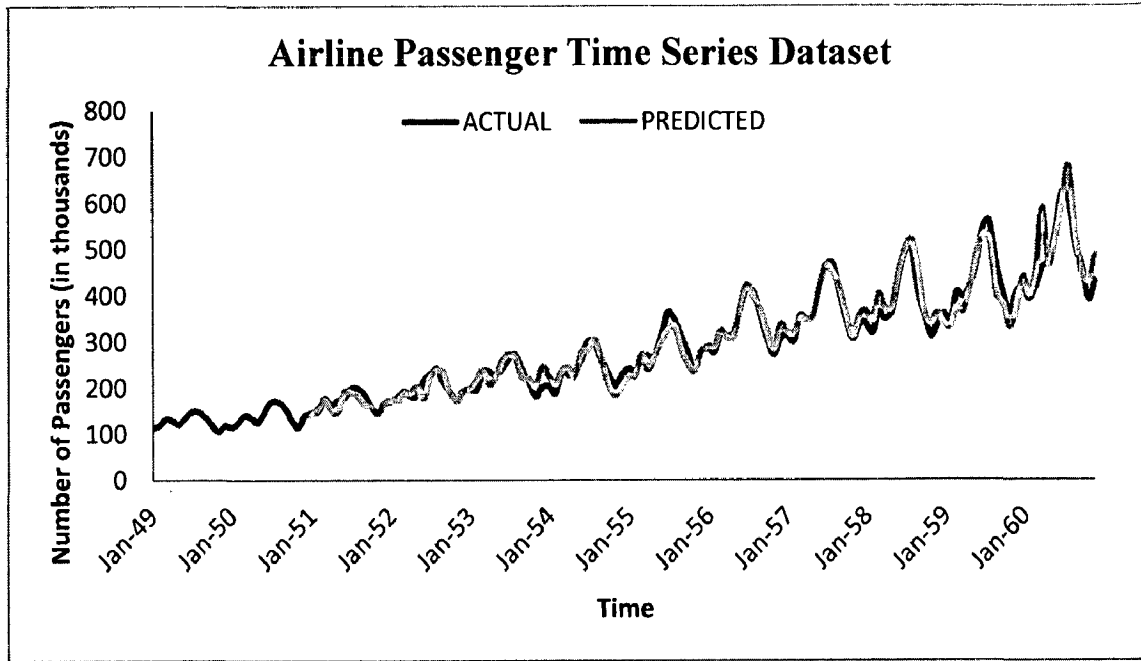


Figure 5.3 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data Without Clustering.

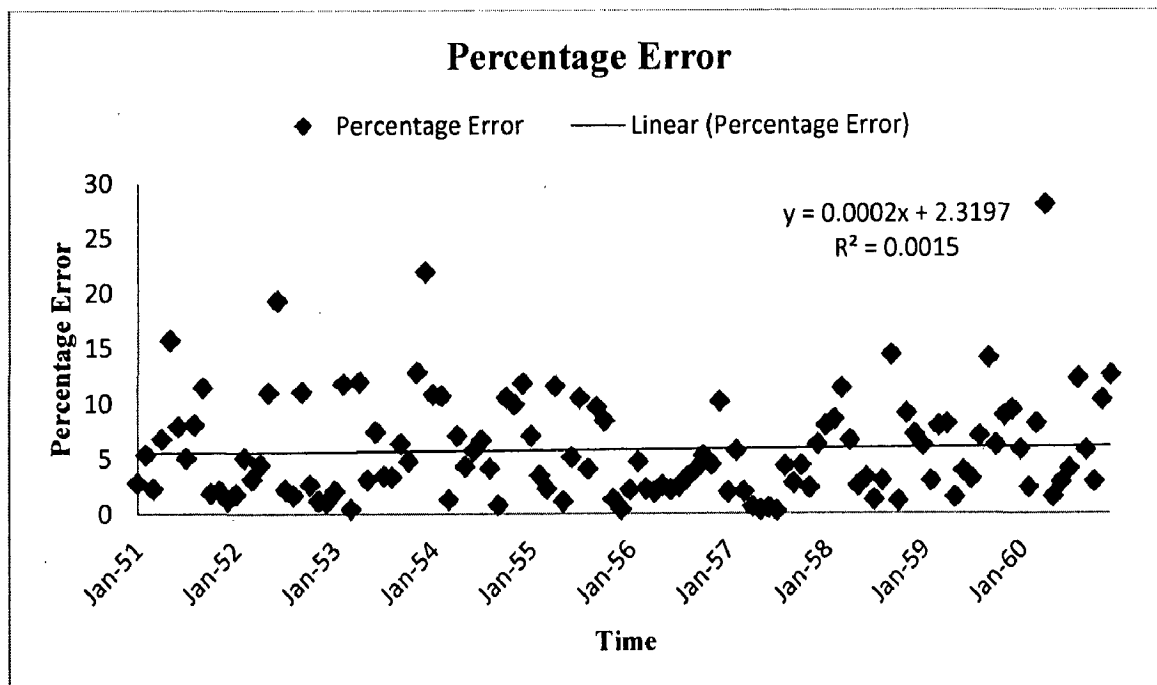


Figure 5.4 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.3. Average Percentage Error is= 5.91.

### SCENARIO 3

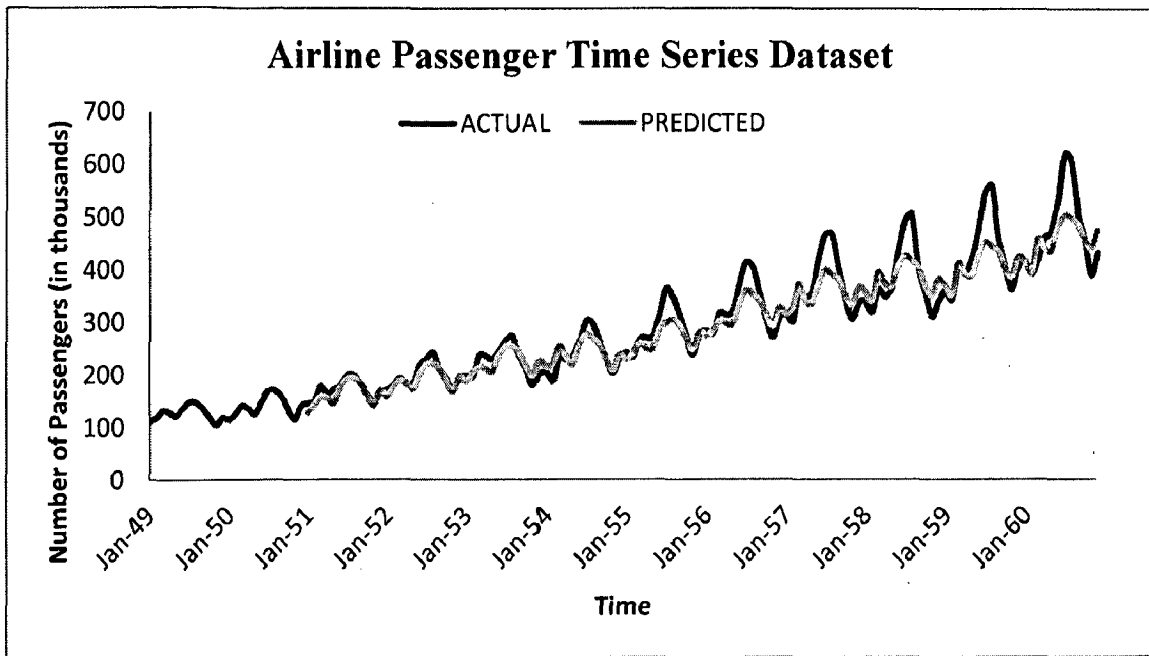


Figure 5.5 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Time Series Data with Seasonal Fluctuations and Clustering.

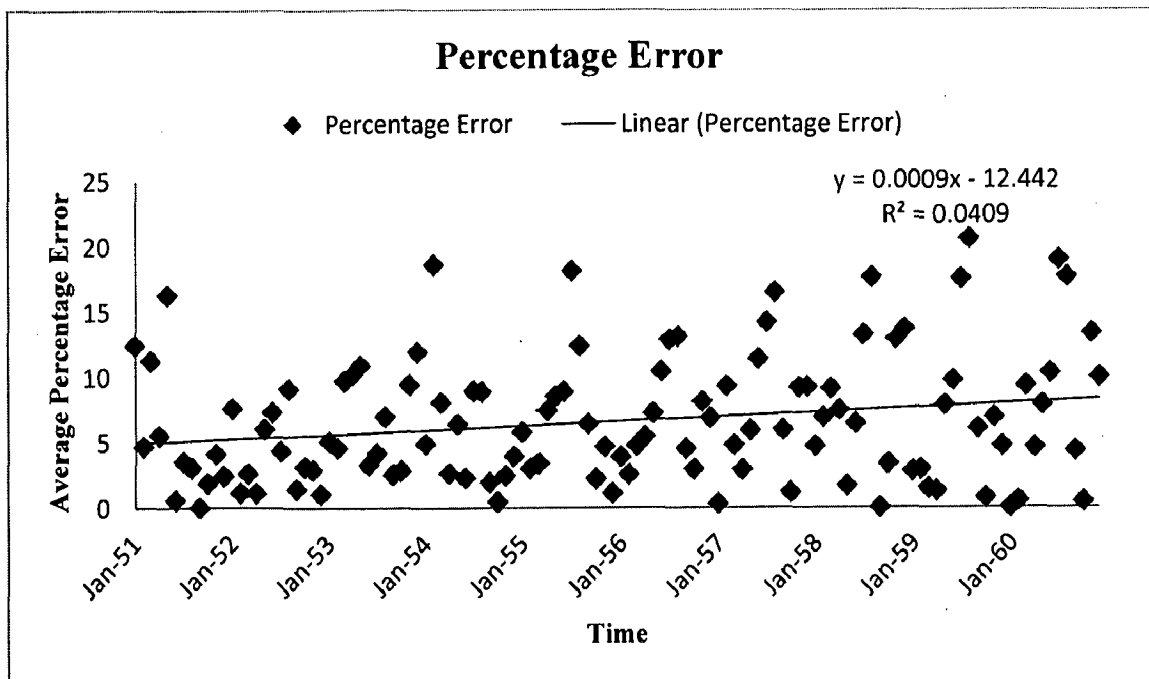


Figure 5.6 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.5. Average Percentage Error is= 6.63.



### SCENARIO 4

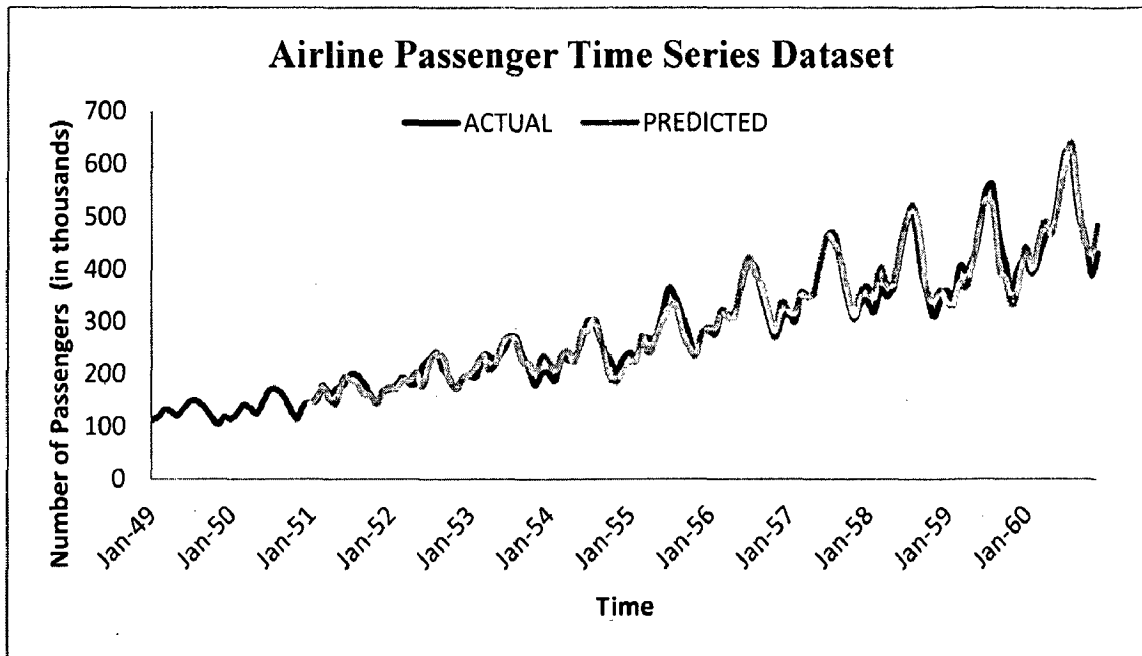


Figure 5.7 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data with Clustering.

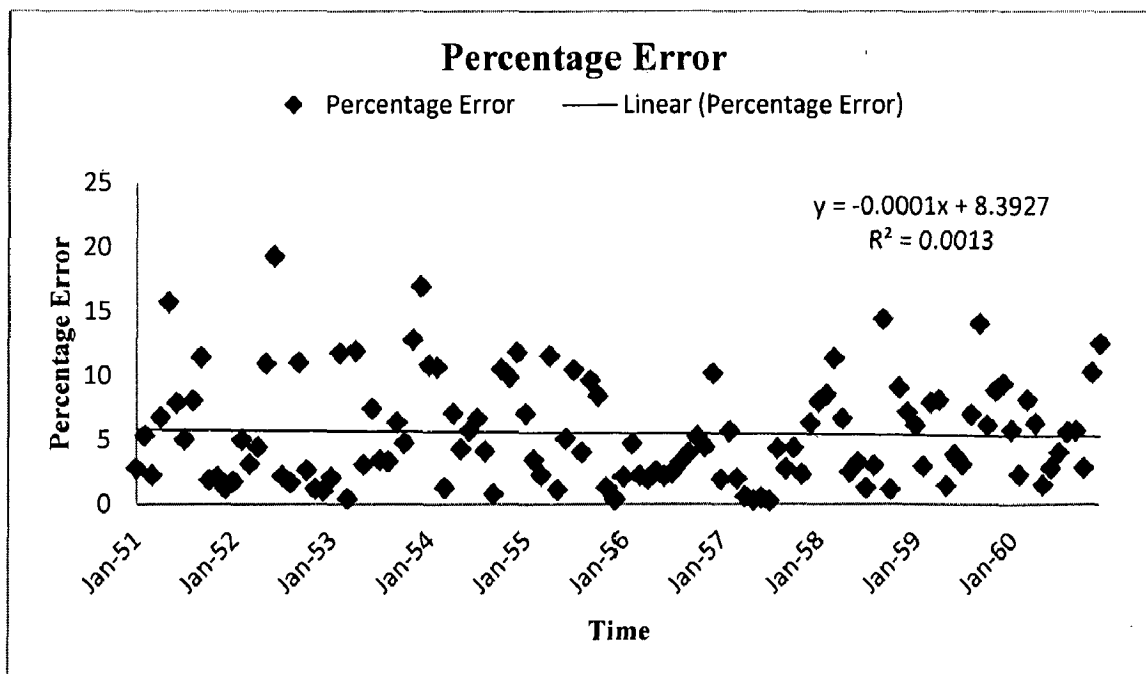


Figure 5.8 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.7. Average Percentage Error is= 5.81.

## 5.2 Results for Monthly Carbon di oxide Concentration Dataset

The dataset has 161 observations which represent monthly mean Carbon di oxide concentrations at the Mauna Loa Observatory from May, 1974 to September, 1987. The data was partitioned into two parts the training data consisting of 24 observations and the testing data consisting of 137 observations. As model predicts the observation for a particular time interval, it is assumed that actual observation would have been recorded and learning in framework is done before predicting the next observation.

The results are summarized as:

- In Figure 5.9, graph represents predicted values which were calculated using scenario 1 and actual observations recorded. Figure 5.10 represents the percentage error observed at each prediction using this (scenario 1) technique.
- In Figure 5.11, graph represents predicted values which were calculated using scenario 2 and actual observations recorded. Figure 5.12 represents the percentage error at each prediction using this (scenario 2) technique.
- In Figure 5.13, graph represents predicted values which were calculated using scenario 3 and actual observations recorded. Figure 5.14 represents the percentage error at each prediction using this (scenario 3) technique.
- In Figure 5.15, graph represents predicted values which were calculated using scenario 4 and actual observations recorded. Figure 5.16 represents the percentage error at each prediction using this (scenario 4) technique.

The average percentage error, reason of error and error pattern in prediction corresponding to all the scenarios depicted in Figure 5.10, 5.12, 5.14 and 5.16 is summarized in Table 5.2.

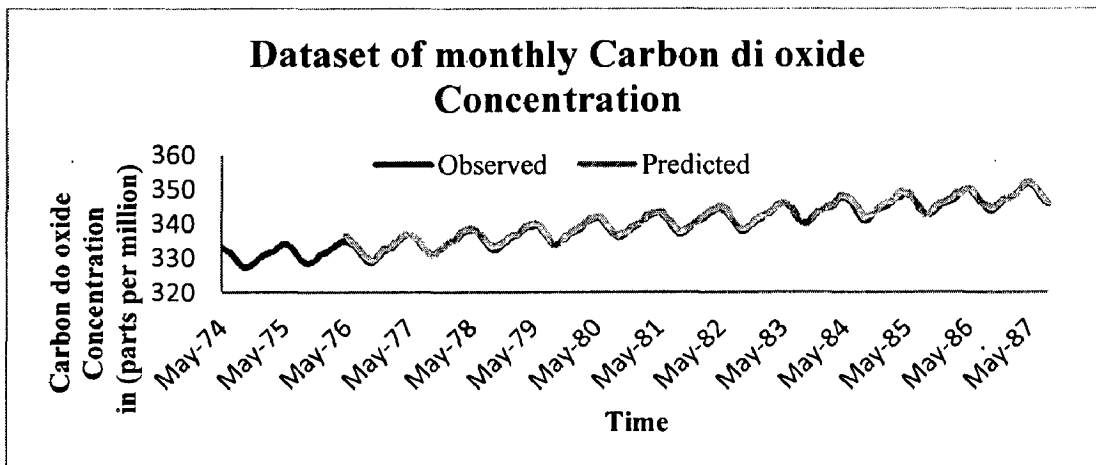
We applied least Square method to **seasonal time series**, where polynomial of degree 3 was fitted using MATLAB. The training data was of 24 observations and 137 observations were predicted and learning on previous observation was done before predicting next observation. The predictions were made with average percentage error rate was 0.5. and when Least square method was applied to **deseasonalized time series** percentage with similar scenario, percentage error rate was 0.44.

**Table 5.2 Average percentage Error on monthly carbon di oxide concentration dataset and Error pattern in different scenarios and reason for such percentage error.**

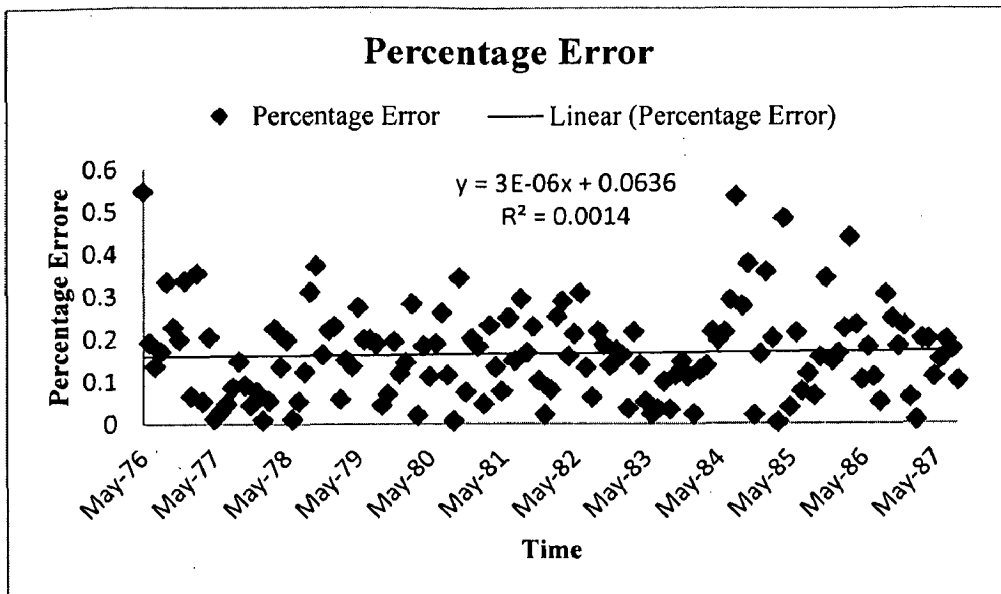
SCENARIO	Average percentage Error	Reason for error	Error pattern
1	.165	The error is since the pattern in time series is evolving with time and since the seasonal fluctuations are also their adding to the error.	Slight increase in Error.
2	.163	The error is reduced due to the deseasonalizing the time series, this approach performs best on this dataset.	Slight Decrease in Error.
3	.168	With clustering result are not good since all the patterns are almost same and they all get clustered hence error has increased.	Error is showing slight decrease.
4	.165	Deseasonalized time series performs better and with clustering results are good.	Error is slightly increasing.

Almost the percentage error is same and is low. The average percentage error is low because the data almost repeats its pattern and there is very low fluctuation. Even in this case graph based framework performs better.

**SCENARIO 1**

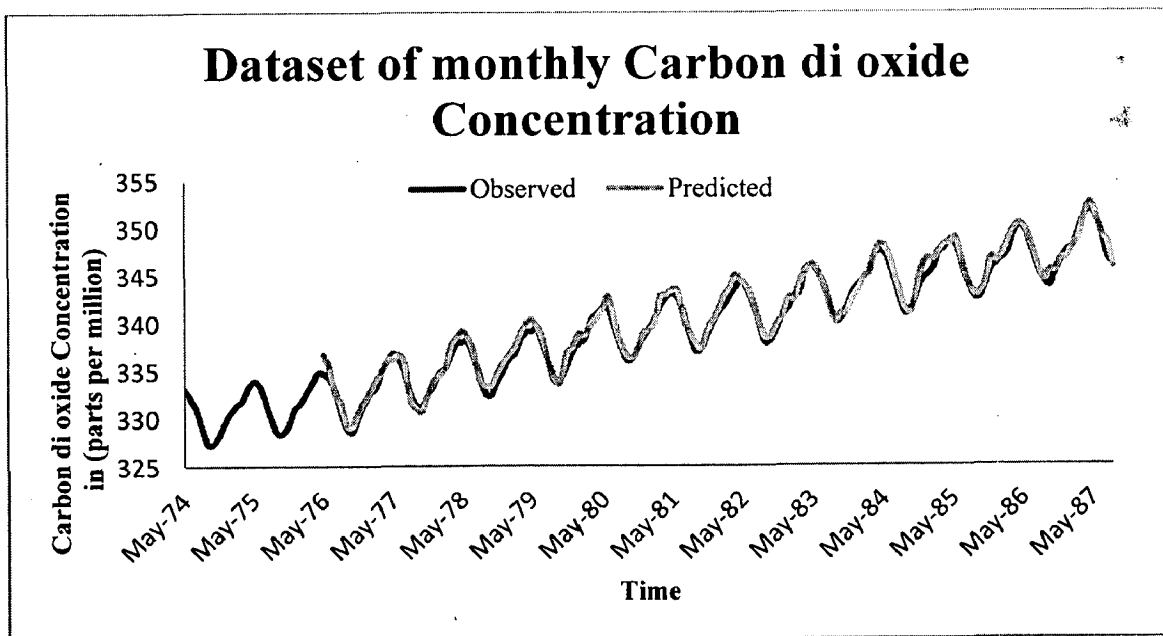


**Figure 5.9 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on time series with Seasonal Fluctuations Without Clustering.**

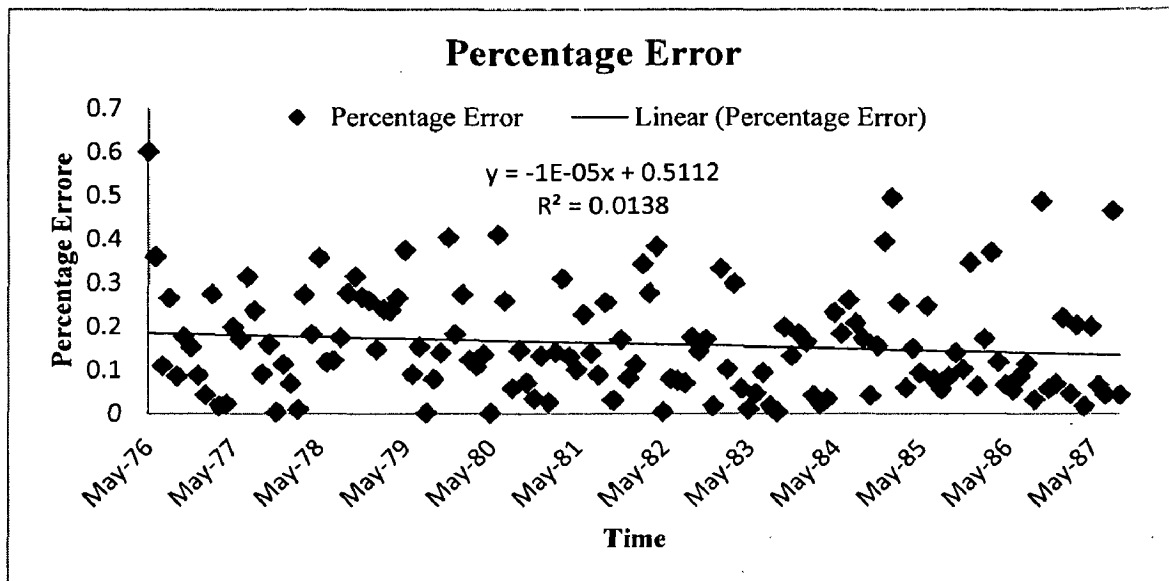


**Figure 5.10 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.9. Average Percentage Error is= 0.165.**

**SCENARIO 2**

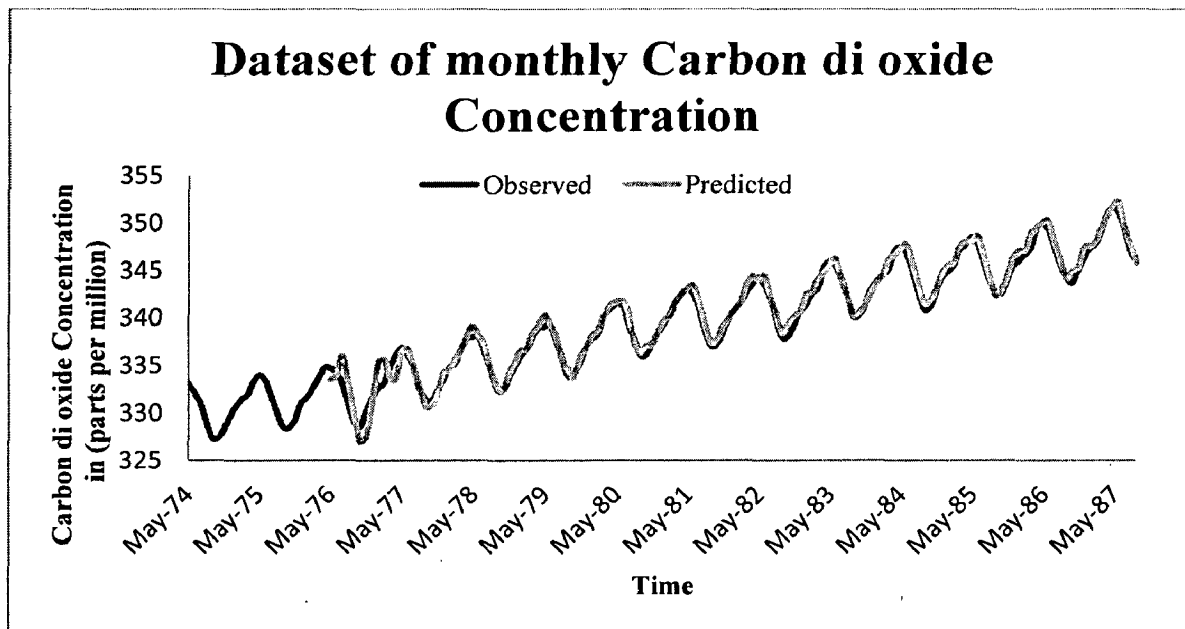


**Figure 5.11 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data Without Clustering.**



**Figure 5.12** Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.11. Average Percentage Error is= 0.163.

*SCENARIO 3*



**Figure 5.13** Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Time Series Data with Seasonal Factors with Clustering.

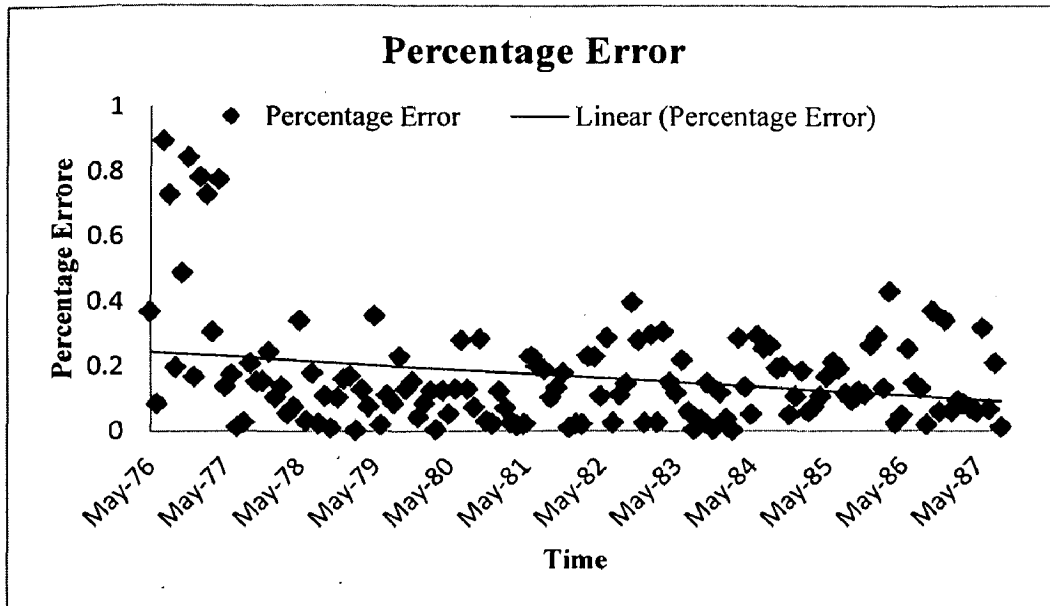


Figure 5.14 Represents the Percentage Error Observed while Predicting Time Series for Predictions shown in Figure 5.13. Average Percentage Error is= 0.168.

#### SCENARIO 4

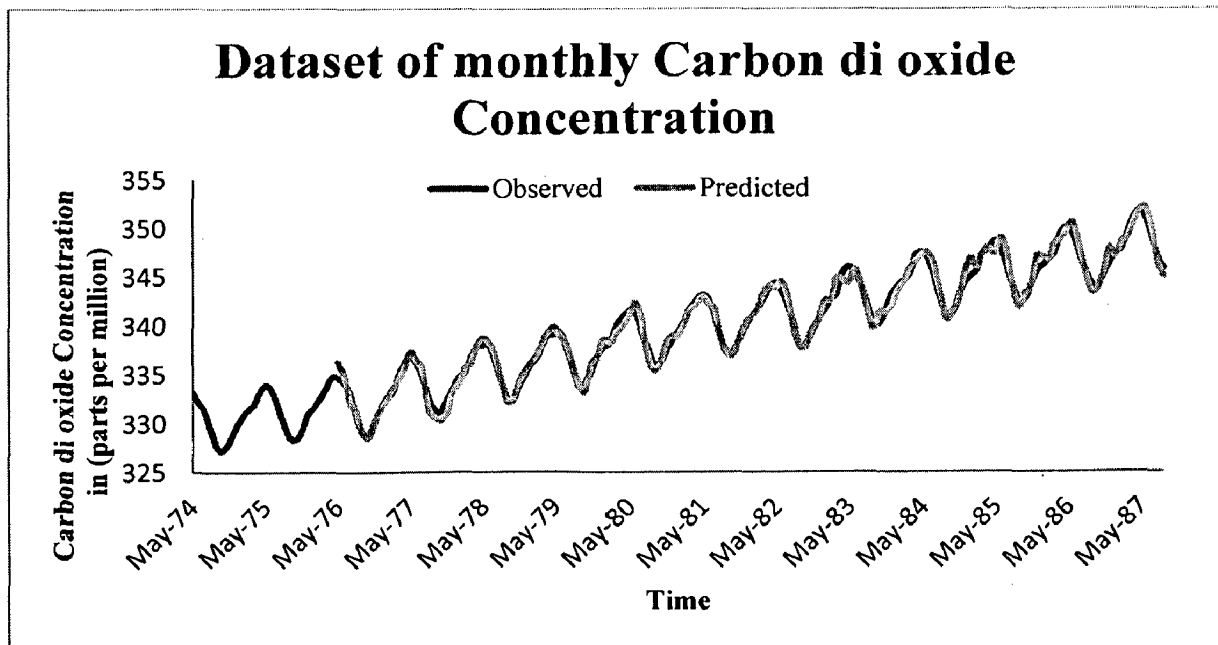
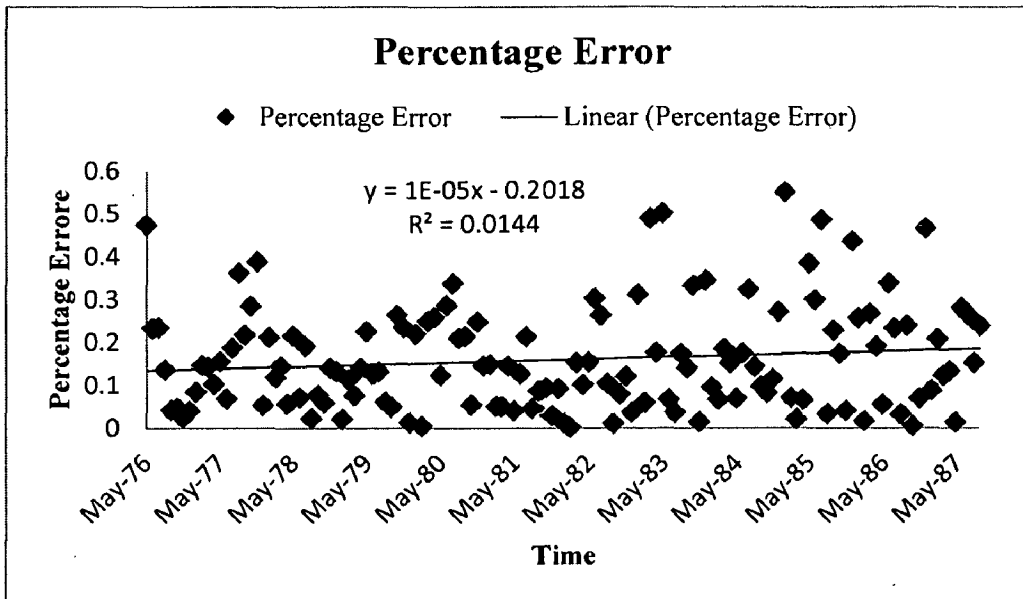


Figure 5.15 Represents Actual and Predicted Observations with respect to Time. Predictions are made using Graph Based Framework on Deseasonalized Time Series Data with Clustering.



**Figure 5.16** Represents the Percentage Error Observed while Predicting Time Series for predictions shown in Figure 5.15. Average Percentage Error is 0.165.

**Table 5.3** Accuracy Rate of Different variants of Graph Based Framework on Different Datasets.

	<b>ACCURACY RATE of Graph Based Framework for Time Series Prediction</b>	
	<b>Airline Passenger Time Series Dataset</b>	<b>Dataset of Monthly Carbon di oxide concentration</b>
<i>Scenario 1</i>	92.95	99.835
<i>Scenario 2</i>	94.09	99.837
<i>Scenario 3</i>	93.37	99.832
<i>Scenario 4</i>	94.19	99.835

**Table 5.4** Accuracy of Least Square Method on Different Dataset, when Third Degree Polynomial was fitted.

	<b>ACCURACY RATE of Least Square Method</b>	
	<b>Airline Passenger Time Series Dataset</b>	<b>Dataset of Monthly Carbon di oxide concentration</b>
<i>Seasonal Time Series</i>	89.6	99.5
<i>Deseasonalized Time Series</i>	91.8	99.56

### 5.3 Discussion

The result obtained conclude that

- The results obtained using deseasonalized time series are better since, deseasonalized time series is free of seasonal fluctuation. Seasonal fluctuations conceal true underlying movement of the time series.
- The results obtained by incorporating graph clustering in the framework have improved the performance since in three ways. First, the searching time has increased. Second, the cluster is more general. Third, it gives framework ability to deal with outliers due to averaging, since mean of cluster, becomes cluster prototype.
- A comparison with Least square technique is done and graph based framework outperforms it considerably.
- The graph based framework with clustering that operates on deseasonalized time series data performs better.
- The proposed framework is robust and can be used on diverse time series data.





## Chapter 6

### Conclusion and Future Work

---

In the present work, time series data was represented as graph to predict time series despite change of pattern in time series over time. This graph representation of time series is then used with existing graph mining techniques in order to extract the concealed patterns from it. The performance of the proposed framework was evaluated using real datasets. The conclusions drawn from the present work can be summarized as follows:

#### 6.1 Conclusion

The following conclusion can be drawn from present study:

- A generic framework has been proposed for the purpose time series prediction on univariate time series that can deal with changes in pattern over time series. The proposed framework has the advantage of being generic and modular and hence it is extensible. It is easy to use and deploy.
- Time series data have been mapped to graphs.
- Clustering of time series graph is done in framework which optimizes the performance of the framework.
- Classification of time series pattern is done using graph edit distance, which is graph matching technique and prediction algorithm for predicting future values using graph representation is proposed.
- The graph based framework with clustering that operates on deseasonalized time series data performs better.
- Finally, the proposed framework was applied on two real world case datasets. First, Airline Passenger time series dataset and second, Dataset of monthly Carbon di oxide concentration. The performance of the proposed framework in predicting these time series was observed and accuracy of 94.2% was achieved for Airline passenger dataset and 99.84% was achieved for Dataset

of monthly Carbon di oxide concentration. The results obtained were good and justify the validity, robustness of the proposed framework and its applicability in diverse environment.

## **6.2 Suggestions for the Future Work**

Some directions for further research work are as follows:

- Existing framework was developed for univariate time series prediction; this framework could be further extended to deal with multivariate time series.
- We have used Graph-Edit distance as metric for similarity, spectral method for graph matching can also be used. Spectral method can model structural variation among graphs accurately.
- We have used a variant of k-mean for graph clustering. This leaves the scope for using other clustering methods such as density based methods in place of k-means.

## REFERENCES

- [1] J. Han and M. Kamber, *Data mining: concepts and techniques*: Morgan Kaufmann, 2006.
- [2] J. T. Thai, "Is Data Mining Ever a Search under Justice Stevens's Fourth Amendment," *Fordham L. Rev.*, vol. 74, p. 1731, 2005.
- [3] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data* vol. 40: Springer-Verlag New York Inc, 2010.
- [4] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on Graph Clustering Algorithms," in *Algorithms - ESA 2003*. vol. 2832, G. Di Battista and U. Zwick, Eds.: Springer Berlin / Heidelberg, 2003, pp. 568-579.
- [5] K. Riesen, X. Jiang, and H. Bunke, "Exact and Inexact Graph Matching: Methodology and Applications," in *Managing and Mining Graph Data*. vol. 40, C. C. Aggarwal and H. Wang, Eds.: Springer US, 2010, pp. 217-247.
- [6] K. Riesen, M. Neuhaus, and H. Bunke, "Bipartite Graph Matching for Computing the Edit Distance of Graphs," in *Graph-Based Representations in Pattern Recognition*. vol. 4538, F. Escolano and M. Vento, Eds.: Springer Berlin / Heidelberg, 2007, pp. 1-12.
- [7] H. Bunke, "Recent developments in graph matching," in *Proc. 15th International Conference on Pattern Recognition*, Barcelona , Spain, 2000, pp. 117-124 vol.2.
- [8] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 3, pp. 265-298, 2004.
- [9] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications*: Springer Verlag, 2000.

- [10] B. Gep, G. Jenkins, and G. Reinsel, "Time series analysis: Forecasting and control," *Oakland CA: Holden-Day*, 1976.
- [11] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis*: Holden-day San Francisco, 1970.
- [12] T. Bollerslev, "A conditionally heteroskedastic time series model for speculative prices and rates of return," *The review of economics and statistics*, vol. 69, no. 3, pp. 542-547, 1987.
- [13] A. kanda, "The Analysis of Time Series," 2008, Available: <http://www.youtube.com/watch?v=zlZaOnBbpUg>, [Last Accesed June, 2011].
- [14] P. J. Brockwell and R. A. Davis, *Time series: theory and methods*: Springer Verlag, 2009.
- [15] G. Box, "Time series analysis: forecasting and control," *S. Francisco Holden-Day*, 1976.
- [16] P. F. Velleman and R. E. Welsch, "Efficient computing of regression diagnostics," *The American Statistician*, vol. 35, no. 4, pp. 234-242, 1981.
- [17] StatSoft, "Electronic Statistics Textbook," 2011, Available: <http://www.statsoft.com/textbook>, [Last Accesed June, 2011].
- [18] D. C. Montgomery, E. A. Peck, G. G. Vining, and J. Vining, *Introduction to linear regression analysis*: Wiley New York, 2001.
- [19] J. C. Hoff, *A practical guide to Box-Jenkins forecasting*: Lifetime Learning Publications, 1983.
- [20] W. Vandaele, *Applied time series and Box-Jenkins models*, 1983.
- [21] D. McDowall, R. McCleary, E. Meidinger, and R. Hay, "Interrupted time series analysis: Sage University Paper series on Quantitative Applications in the Social Sciences," *Interrupted time series analysis: Sage University Paper series on Quantitative Applications in the Social Sciences*, 1980.

- [22] D. G. Bails and L. C. Peppers, *Business fluctuations: forecasting techniques and applications*: Prentice-Hall, 1982.
- [23] E. S. Gardner Jr, "Exponential smoothing: The state of the art," *Journal of Forecasting*, vol. 4, no. 1, pp. 1-28, 1985.
- [24] D. C. Montgomery, L. A. Johnson, and J. S. Gardiner, *Forecasting and time series analysis*: McGraw-Hill New York etc., 1990.
- [25] E. S. Gardner, "Exponential smoothing: The state of the art--Part II," *International Journal of Forecasting*, vol. 22, no. 4, pp. 637-666, 2006.
- [26] S. Makridakis, "Empirical evidence versus personal experience," *Journal of Forecasting*, vol. 2, no. 3, pp. 295-309, 1983.
- [27] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159-175, 2003.
- [28] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1-2, pp. 35-49, 2003.
- [29] M. Gori, M. Maggini, and L. Sarti, "The RW2 Algorithm for Exact Graph Matching," in *Pattern Recognition and Data Mining*, vol. 3686, S. Singh, M. Singh, C. Apte, and P. Perner, Eds.: Springer Berlin / Heidelberg, 2005, pp. 81-88.
- [30] B. Luo and E. R. Hancock, "Structural graph matching using the EM algorithm and singular value decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1120-1136, 2001.
- [31] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis & Applications*, vol. 13, no. 1, pp. 113-129, 2010.

- [32] M. Gori, M. Maggini, and L. Sarti, "Graph matching using random walks," in Proc. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004, pp. 394-397 Vol.3.
- [33] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*: Prentice-Hall, Inc., 1988.
- [34] M. Ferrer and H. Bunke, "An Iterative Algorithm for Approximate Median Graph Computation," in Proc. *Pattern Recognition (ICPR), 2010 20th International Conference on*, 2010, pp. 1562-1565.
- [35] S. Jouili, S. Tabbone, and V. Lacroix, "Median Graph Shift: A New Clustering Algorithm for Graph Domain," in *Proceedings of the 2010 20th International Conference on Pattern Recognition*: IEEE Computer Society, 2010, pp. 950-953.
- [36] H. Bunke and S. Günter, "Weighted mean of a pair of graphs," *Computing*, vol. 67, no. 3, pp. 209-224, 2001.
- [37] T. H. Cormen, *Introduction to algorithms*: The MIT press, 2001.
- [38] K. W. Thoning, P. P. Tans, and W. D. Komhyr, "Atmospheric carbon dioxide at Mauna Loa Observatory, 2, Analysis of the NOAA/GMCC data, 1974-1985," *J. Geophys. Res.*, vol. 94, pp. 8549–8565, 1989.

## LIST OF PUBLICATIONS

---

- [1] **Vivek Yadav** and Durga Toshniwal, "Graph Based Framework for Time Series Prediction," in *National Seminar on Open Source Software Systems (OSSS2011)* Srinagar, Kashmir: Trends in Information Management (TRIM) 2011.