APPLICATION OF FUZZY C-MEANS CLUSTERING AND PARTICLE SWARM OPTIMIZATIONTO IMPROVE VOICE TRAFFIC FORECASTINGIN FUZZY TIME SERIES

BY

SHEHU MOHAMMED YUSUF

MSc/ENG/5641/2009-2010

sheikullahi@gmail.com

A Dissertation Submitted to the Department of Electrical and Computer Engineering, Ahmadu Bello University, Zaria, in Partial Fulfillment of the Requirements for the Award of Master of Science (M.Sc.) Degree in Electrical Engineering.

May, 2015

DECLARATION

I declare that the work in this Dissertation entitled "Application of Fuzzy C-Means Clustering and Particle Swarm Optimization to improve Voice Traffic Forecasting in Fuzzy Time Series" has been carried out by me in the Department of Electrical and Computer Engineering. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other Institution.

Yusuf, Shehu Mohammed _____ _____

Signature **Date**

CERTIFICATION

This Dissertation entitled APPLICATION OF FUZZY C-MEANS CLUSTERING AND PARTICLE SWARM OPTIMIZATION TO IMPROVE VOICE TRAFFIC FORECASTING IN FUZZY TIME SERIES by SHEHU MOHAMMED YUSUF meets the regulations governing the award of the degree of Master of Science (M.Sc.)in Electrical Engineering of Ahmadu Bello University, and is approved for its contribution to knowledge and literature presentation.

Prof. M. B. Mu'azu _____ _____ (Chairman,
Supervisory Committee) (Signature) Date

Dr. O. Akinsanmi _____ _____
(Member, Supervisory Committee)(Signature) Date

Prof. M. B. Mu'azu _____ _____
(Head of Department)(Signature) Date

Prof. A. Z. Hassan _____ _____
(Dean of Postgraduate Studies) (Signature) Date

DEDICATION

This work is dedicated to my mother, wifeand late father, whom I thank for their blessings.

ACKNOWLEDGEMENTS

My eternal gratitude is to Almighty Allah for the love,knowledge and wisdom bestowed on me throughout my academic works. The successful completion of this thesis would not have been possible without the grace of the Almighty.

I owe a great deal to my supervisor and mentor, Prof. M. B. Mu'azu, who patiently read through each page of my dissertation and made critical contributions to improve its contents. My appreciation goes to Dr. S. M. Sani for his constructive criticisms and valuable contributions.

Next, I express thanks to Dr. O. Akinsanmiand Dr. T. H. Sikiru for recognizing the effort of this work.

To my minor supervisor, Dr. O. Akinsanmi, I say thank you for supporting me academically and morally. Your support motivated me to conquer my academic challenges.

I owe so much to Engr. A. Mubarak, my brother, Abdulmalik and uncle, Dr. A. Abdulkareem for their prayers andacademic support.

I am grateful to all my lecturers; Prof. B. G. Bajoga, Dr. S. Garba, Dr. B. Jimoh, Dr. D.D.Dajab, Dr A. D. Usman, Dr. K. A. Abu-Bilal, Engr. S. A. Adamu, Engr. M. Abdullahi, Engr. Taiwo,Engr.Y. Sha'aban, Engr. E. Okafor, Engr. O. Josaiah, and Engr. A. S. Eleruja.

Also, noteworthy of thanks are my colleagues; Engr. Hussaina, Engr. S. Mosunmola, Engr. A. Sarkin Bauchi, Engr. A. Galadimaand all those whose names are not captured.

Finally, I am thankful for the support of my friend, Engr. Abdullahi ("Minister"); siblings; and especially my mum; wife, Salamatu; and in-laws. They cheered me up whenever I was overwhelmed during the writing of this dissertation.

TABLE OF CONTENTS

TITLE PAGE ..... **i**

DECLARATION ..... **ii**

CERTIFICATION ..... **iii**

DEDICATION ..... **iv**

ACKNOWLEDGEMENTS ..... **v**

TABLE OF CONTENTS ..... **vi**

LIST OF APPENDICES……..**ix**

LIST OF FIGURES ..... **x**

LIST OF TABLES ..... **xi**

LIST OF ABBREVIATIONS ..... **xiii**

ABSTRACT ..... **xvi**

CHAPTER ONE: INTRODUCTION

1.1 Background Information **1**

1.2 Aim and Objectives **2**

1.3 Statement of the Problem **2**

1.4 Methodology **3**

1.5 Significant Contributions…….....**5**

1.6 Thesis Organization **5**

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction **6**

2.2 Review of Fundamental Concepts **6**

2.2.1 Time Series……..……**6**

2.2.2 Fuzzy Set Theory **6**

2.2.3 Fuzzy Time Series and Fuzzy Logic Relationship **7**

2.2.4 Universe of Discourse 9

2.2.5 Fuzzy Set Groups 10

2.2.6 Data Mining and Clustering 11

2.2.6.1Distance Measure 13

2.2.7Fuzzy C-Meeans Clustering 13

2.2.8Cluster Validity Index 16

2.2.9 Particle Swarm Optimization 17

2.2.10 Defuzzification Operator 19

2.2.11Erlang Based Voice Traffic 20

2.2.12 Performance Measure 20

2.2.13 Programming Language 23

2.2.13.1C programming Language 23

2.2.13.2 C++ Programming Language 24

2.2.13.3 Java Programming Language 24

2.2.13.4 C# Programming Language 24

2.3 Review of Similar Works 25

CHAPTER THREE: MATERIAL AND METHODS

3.1 Introduction 30

3.2Data Collection and Processing 30

3.3 Fuzzification Module 31

3.3.1 Coding fuzzy C-Means (FCM) Clustering Algorithm in C# 31

3.3.2 Applying Time Series Data on Fuzzy C-Means Code 36

3.3.3 Ranking Clusters in Ascending Order 40

3.3.4 Fuzzifying Time Series Data 41

3.4 Defuzzification Module 43

3.4.1 Establishing Fuzzy Set Groups (FSGs).....43

3.4.2 Converting Fuzzy Set Groups into "if – then" Rules.....47

3.4.3 Tuning "if – then" Rules Using Particle Swarm Optimization (PSO).....50

3.4.4 Deriving Forecasts.....63

3.5 Investigating the Effect of Reversed Weights.....64

3.6 Forecasting Test Data Set.....67

3.7 Forecasting Using Chen's (1996) Fuzzy Time Series Model.....71

3.8 Forecasting Using Cheng *et al* (2008) Hybrid Model.....72

CHAPTER FOUR: RESULTS AND DISCUSSIONS

4.1 Introduction……..75

4.2 Forecasting Results for Training Data Set.....75

4.3 Forecasting Result for Test Data SetForecasts.....80

4.4 Validation.....82

4.5 Significance of Forecasting Results.....95

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

5.1 Summary..... 96

5.2 Conclusion 96

5.3 Limitations..... 97

5.4 Recommendations for Further Works..... 97

REFERENCES..... 99

LIST OF APPENDICES

APPENDIX A: Postpaid and Prepaid Calls, Abuja Call Centre…….……102

APPENDIX B:Cluster Validity Calculation Using MSE_c ……..…..……**105**

APPENDIX C:Training Data Set Partition Matrix (Membership Degrees) for $c = 14$ **…106**

APPENDIX D: Extension Process for obtaining Disambiguated Fuzzy Set Groups (FSGs)...

……..……**107**

APPENDIX E: Calculation of Distances for Test Data Set……..…....**109**

APPENDIX F: Fuzzified Daily Erlang Training Set Traffic (Chen's Model)…..……...**111**

APPENDIX G: Fuzzy Logical Relationships (FLRs) of Erlang (Training) Traffic using Chen's (1996) Model……..…….....**112**

APPENDIX H: Fuzzy Logical Relationships (FLRs) of Erlang (Training) Traffic using Cheng *et al* (2008) Model…….……..…….....**113**

APPENDIX I: FCM Code……..…….....**114**

APPENDIX J: PSO Code……..…….....**128**

LIST OF FIGURES

Figure 1.1: Flowchart of Proposed Hybrid Model……..…….…….……..4

Figure 3.1: Flowchart of Fuzzy C-Means Clustering Algorithm…….…….……31

Figure 3.2: Fuzzy C-Means Clustering Graphical User Interface….……...….37

Figure 3.3:Screenshot Showing Partition Matrix and Cluster Centres….…….……..38

Figure 3.4:Flowchart of Particle Swarm Optimization Algorithm…….……...…….51

Figure 3.5: Screenshot of the Particle Swarm Optimization…….……..…….…….52

Figure 3.6: Graphical User Interface Showing Results of Tuning Weights of Fuzzy Rule……..59

Figure 4.1: Plot of Actual and Forecasted Erlang (Training Data Set) against Time……..78

Figure 4.2: Plot of Actual Training Data Set and Forecasted (Reversed Weights) Erlang against Time……...……79

Figure 4.3: Plot of Actual and Forecasted Erlang against Time for Test Data Set……..…81

Figure 4.4: Plot of Actual (Training Set) and Forecasted Erlang (Chen's 1996 Model) against Time…….……..…….89

Figure 4.5: Plot of Actual (Training Set) and Forecasted Erlang (Cheng's 2008 Hybrid Model) against Time……...…..92

Figure 4.6: Plot of Actual (Test Set) and Forecasted Erlang (Cheng's 2008 Hybrid Model) against Time……....94

LIST OF TABLES

Table 3.1:Erlang Traffic Data Set for Training……..…….……...**40**

Table 3.2: Cluster Centres for Training Erlang Data Set…….……..…..…..……**39**

Table 3.3: Defined Fuzzy Sets for Training Data Set Partition into Fourteen Clusters..….**42**

Table 3.4: Fuzzified Daily Erlang C Training Set Traffic…….…….……**43**

Table 3.5: Establishment of Fuzzy Set Groups……...……..……...….**45**

Table 3.6: Disambiguated Fuzzy Set Groups……...……...…….….**47**

Table 3.7: Partially Generated "if" Rules in Chronological Order…….……...**48**

Table 3.8: Completely Generated "if-then" Rules for Training Data Set…….…….**61**

Table 3.9: Generated "if-then" Rules for Training Data Set (Reversed Weights)……**65**

Table 3.10: Erlang Traffic Observation (Test Set)……...……**67**

Table 3.11: Fuzzified Daily Erlang (Testing Set) Traffic…..……**68**

Table 3.12: Established Fuzzy Set Groups for Test Data Set...……...……**69**

Table 3.13 Generated "if-then" Rules for Test Data Set...…….……..**70**

Table 3.14: Forecasted Training Data Set Using Chen (1996) FTS Model……..……..**72**

Table 3.15:Forecasted Training Data Set Using Cheng (2008) Hybrid Model...…….…….**73**

Table 3.16: Forecasted Test Data Set Using Cheng (2008) Hybrid Model…..…....……**74**

Table 4.1: Forecasted Voice (Erlang) Traffic of Training Set…..……...……...……**76**

Table 4.2: Forecasted Voice (Erlang) Traffic of Training Set (Reversed Weights)…....….**77**

Table 4.3: Forecasted Voice Traffic of Test Set for the Proposed Hybrid Model…..……....**80**

Table 4.4:Calculation of MSE and MAPE of Forecast forTraining the Proposed Model..**83**

Table 4.5:MSE and MAPE Calculation for Training the Proposed Model (Reversed Weights)…..…….……85

Table 4.6: Calculation of MSE and MAPE for Testing the Proposed Hybrid Model…….**86**

Table 4.7 Calculation of MSE and MAPE of Forecasts obtained for Chen's (1996) Model…

……...**88**

Table 4.8 Calculation of MSE and MAPE for Forecast obtained for Cheng's (2008) Model Training…..……..……...……..…….**91**

Table 4.9 Calculation of MSE and MAPE for Forecast obtained for Cheng's (2008) Model Testing…..……..…….**.....93**

LIST OF ABBREVIATIONS

Acronym	Definition
FCM	Fuzzy C-means Clustering
PSO	Particle Swarm Optimization
FTS	Fuzzy Time Series
QoS	Quality of Service
MSE	Mean Square Error
MAPE	Mean Absolute Percentage Error
FLR	Fuzzy Logic Relationship
FLRG	Fuzzy Logic Relationship Group
PE	Partition Entropy
PC	Partition Coefficient
MSE_c	Mean Square Error for Cluster Validation
EA	Evolutional Algorithm
GA	Genetic Algorithm
MA	Memetic Algorithm

ACO	Ant Colony Optimization
CPU	Central Processing Unit
ACD	Automatic Call Distributor
KPI	Key Performance Index
FORTRAN	Formula Translator
USA	United States of America
JVM	Java Virtual Machine
C#	C-Sharp
OOP	Object Oriented Programming
OS	Operating System
TFA	Trapezoidal Fuzzification Approach
HPSO	Hybrid Particle Swarm Optimization
PC	Portable Computer
GUI	Graphical User Interface
RAM	Random Access Memory
GHz	Giga Hertz

AHT	Average Handle Time
ASA	Average Speed of Answer
SE	Squared Error

ABSTRACT

Forecasting of voice traffic using an accurate model is important to the telecommunication service provider in planning a sustainable Quality of Service (QoS) for their mobile networks. This work is aimed at forecasting Erlang C – based voice traffic using a hybrid forecasting model that integrates fuzzy C-means clustering (FCM) and particle swarm optimization (PSO) algorithms with fuzzy time series (FTS) forecasting model. Fuzzy C-means (FCM) clustering, which is an algorithm for data classification, is adopted at the fuzzification phase to obtain unequal partitions. Particle swarm optimization (PSO), which is an evolutional search algorithm, is adopted to optimize the defuzzification phase; by tuning weights assigned to fuzzy sets in a rule.This rule is a fuzzy logical relationship induced from a fuzzy set group (FSG). The clustering and optimization algorithms were implemented in programs written in C#. Daily Erlang C traffic observations collected over a three (3) month period from 1 December, 2012 – 28 February, 2013 from Airtel, Abuja region, was used to evaluate the proposed hybrid model.To evaluate the forecasting efficiency of the proposed hybrid model, its statistical performance measures of mean square error (MSE) and mean absolute percentage error (MAPE), were calculated and compared with those of a conventional fuzzy time series (FTS) model and, a fuzzy C-means (FCM) clustering and fuzzy time series (FTS) hybrid model.Statistical results of $MSE = 0.9867$ and $MAPE = 0.47\ \%$ were obtained during training of the proposed hybrid forecasting model. Compared with the training results of $MSE = 845.122$ and $MAPE = 13.47\ \%$, for Chen's (1996) FTS model and; $MSE = 856.145$ and $MAPE = 13.37\ \%$, for Cheng's (2008); the proposed hybrid forecasting model resulted in a relatively higherforecasting accuracy and precision. Also, performancemeasures of $MSE = 59.22$ and $MAPE = 3.85\ \%$ were obtained during thetesting phase of the proposed model. Compared with the test results of $MSE = 1567.4$ and $MAPE = 23.98\ \%$ obtained for Cheng's (2008) FCM/ FTS hybrid model, the proposed hybrid forecasting model also showed a relatively higher forecasting accuracy and precision. Finally, it was determined that reversing the weights of the forecasting rules, during training, resulted to a lesser performance; $MSE = 42.73$ and $MAPE = 0.88\ \%$. Thus, reversing the weights of forecasting rule affected the forecasting accuracy.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND INFORMATION

Since its inception over three decades ago, mobile telecommunication call centres have witnessed exponential growth. Call centres are on the increase owing to the large number of mobile subscribers and the need for telecommunication operators to lower cost of providing services while increasing time access of their services. Understanding voice traffic pattern of a call centre becomes critical to service providers in predicting traffic, planning and budgeting for future changes of their mobile networks. This is important for sustaining a good Quality of Service (QoS).

Forecasting is used to predict, model and simulate the future from past events in virtually all fields of endeavours. In the telecommunication industry, forecasting is a useful tool in planning, budgeting, evaluating and verifying network resources (Eleruja *et al*, 2012).

Voice traffic is one of the critical measures in mobile telecommunication systems. Since this measure is non – linear and dynamic with time, forecasting Erlang based voice traffic observations using fuzzy time series (FTS) models seems to be more suitable than conventional statistical models. Fuzzy time series (FTS) models take care of uncertainties in observations over time and does not require any restrictive assumptions and too much background knowledge of the data; like in the case of conventional statistical forecasting methods.The use of fuzzy time series (FTS) in forecasting was first introduced by Song and Chissom (1993). This approach comprises two phases; fuzzification and defuzzification. Fuzzification is a technique for conversion of real observations into discrete or linguistic fuzzy sets. Defuzzification is a technique for converting linguistic observations to real values.

Recently, due to the need for improving forecasting accuracy, hybrids of fuzzy time series approaches have become the research trends in literature. In this study a novel hybrid fuzzy time series that integrates fuzzy C-means (FCM) clustering and particle swarm optimization, in the fuzzification and defuzification phases respectively, with fuzzy time series (FTS) model was proposed.

1.2 AIM AND OBJECTIVES

The aim of this research is the application of fuzzy C-means (FCM) clustering and particle swarm optimization (PSO) algorithms in fuzzy time series (FTS) forecasting model in order to improve the forecasting accuracy. The objectives of the research are as follows:

I. Development of a GUI based fuzzy C-means (FCM) clustering model to objectively partition the universe of discourse into unequal lengths (cluster centres) and to learn the memberships in hidden data structure.
II. Development of a GUI based particle swarm optimization (PSO) model to optimize the defuzzification process.
III. Reduction of computational complexities of the forecasting process in using FCM clustering and particle Swarm optimization (PSO) by implementation of the algorithms in C#.
IV. Validation using Erlang based voice traffic data obtained from Airtel, Abuja Call Centre.
V. Comparison of the results obtained using the developed hybrid model with results obtained using other models (Chen's (1996) model and Cheng et al (2008) hybrid model).

1.3 STATEMENT OF PROBLEM

Accurate and robust fuzzy time series models capable of determining objective interval length, memberships that explains unknown structures in data sets, minimizing loss of forecasting rules,

and reducing computational complexities are challenging issues facing fuzzy time series forecasting. It has become necessary to solve these challenges using hybrid fuzzy time series models. As a consequence, employing fuzzy C-means (FCM) clustering algorithm in fuzzification, fuzzy set groups (FSGs) to generate logical relationships, and particle swarm optimization (PSO) algorithm in defuzzification will improve fuzzy time series forecasting accuracy. Coding thevarious algorithms used in the forecasting process in high – level or object – oriented programming languages like matlab, C++ or C# will reduce computational complexity. C# was chosen in this work because of its pure object – oriented programming features which can be integrated with Windows operating systems.

1.4 METHODOLOGY

The following methodology was adopted in carrying out this research:

1. Collection and processing voice (Erlang) traffic observations for Airtel, Abuja call centre.
2. The fuzzification module comprises the following steps:
    a. Code fuzzy C-means (FCM) clustering algorithm in C#.
    b. Apply voice (Erlang) traffic observations on the fuzzy C-means (FCM) clustering code to compute cluster centres, v_i , and membership degrees (partition matrix).
    c. Rank cluster values, v_i , in ascending order to determine ordered linguistic variables, A_r $r = 1,2,3,....,c$.
    d. Fuzzify the voice traffic data sets using the partition matrix and the rank.
3. The defuzzification module comprises the following steps:
    a. Establish disambiguated fuzzy set groups (FSGs).
    b. Convert fuzzy set groups (FSGs) to “if - then” rules.

c. Tune the "if - then" rules using particle swarm optimization (PSO) algorithm coded in C#.

d. Based on the results of the training, derive the possible outcomes of the voice traffic forecast, using a defuzzification operator.

4. Investigate the effect of reversed weights.

5. Validate and compare forecasts using measured data from Airtel, Abuja Call Centre.

The flowchart for the model design is shown in Figure 1.1.

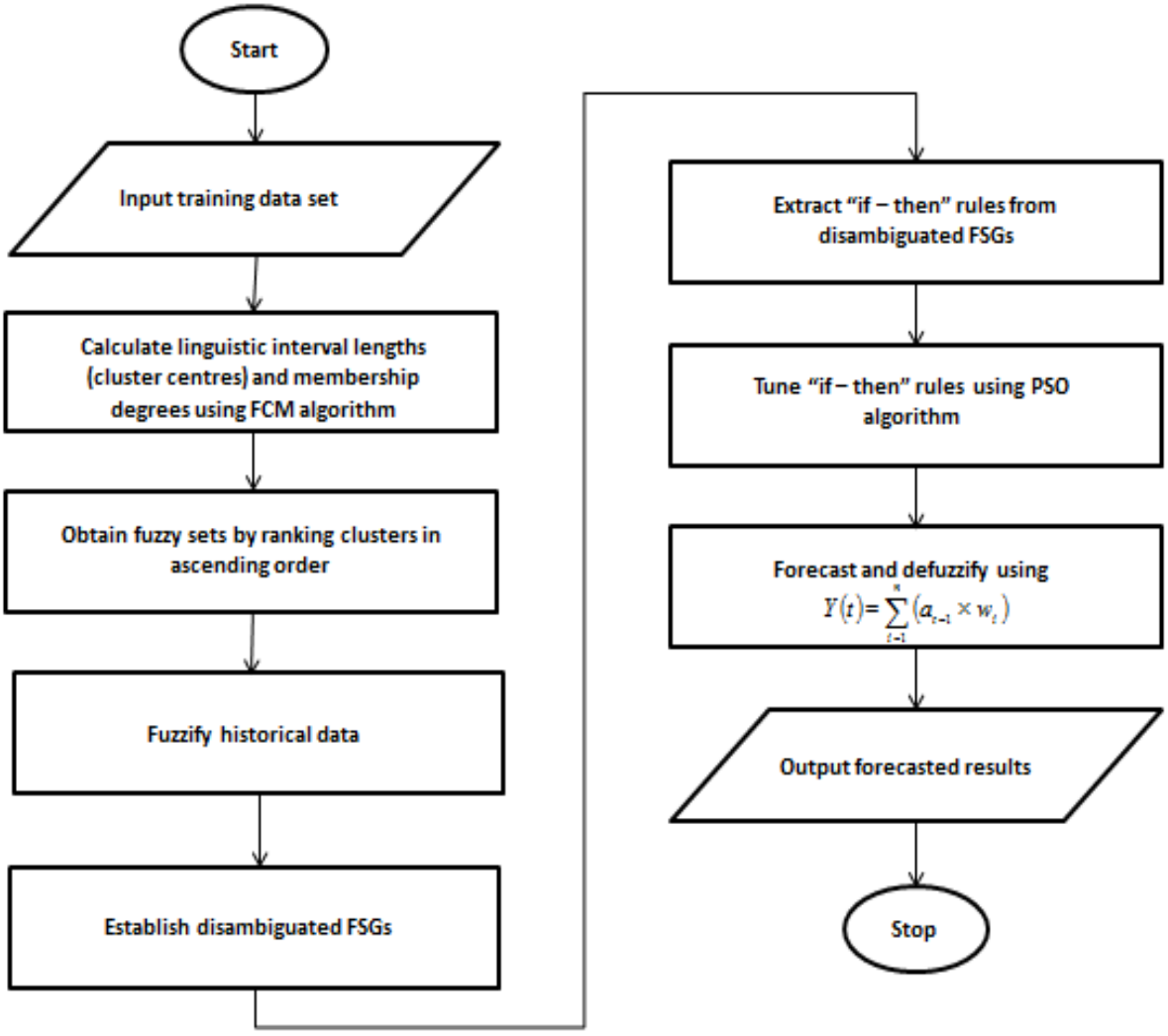


Figure 1.1: Flowchart of the Proposed Hybrid Model.

1.5 SIGNIFICANT CONTRIBUTIONS

The significant contributions derivable from this work are as follows:

1) Development of a GUI based FTS model that incorporates fuzzy C-means clustering and particle swarm optimization which reduces the computational cost whilst easing the forecasting process.

2) Development of a hybrid fuzzy time series forecasting model that has improved forecasting performance in terms of mean square error (MSE). In comparison with the Chen's (1996) fuzzy time series model and Cheng's (2008) hybrid model, during the training phase, the developed model has improved mean square errorperformance by over 99 % in forecasting the Airtel voice traffic, Abuja region.When compared with the Cheng's (2008), the developed model showedimprovement in mean square error performance by over 96 % during the testing phase.

1.6 DISSERTATION ORGANIZATION

The general introduction has been presented in Chapter One. The remaining chapters were structured as follows: A detailed review of the relevant literature and pertinent fundamental conceptswas carried out in Chapter Two.Chapter Three discussed the methodology adopted in achieving the set objectives.The results obtained were analyzed and discussed in Chapter Four. Chapter Five discussed the conclusion and recommendations for further work. Quoted references and Appendices are also provided at the end of the thesis.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter is divided into two parts. The first part discusses the fundamental concepts relevant to the thesis and the second part provides a review ofsimilar works.

2.2 REVIEW OF FUNDAMENTAL CONCEPTS

This section presents the review of theoretical background and fundamental concepts relevant in the context of this work.

2.2.1 Time Series

A time series is a sequence of well – defined observations measured at regular intervals of time. It is a sequence of n data points, $X_1, X_2 ....., X_n$, consisting of continuous and non – linear values changing with time (Cheng et al, 2008). Time series forecasting is a mathematical method for predicting future time series observation, X_{n+1}, from historical time series observations, $X_1, X_2 ....., X_n$.

2.2.2 Fuzzy Set Theory

Fuzzy set theory was first introduced by Zadeh (1965) designed to mathematically represent and manipulate imprecise (or fuzzy) data. This was developed based on the notion of purely crisp set. In fuzzy set, elements belong to the set with a certain degree of membership. It provides formal tools for dealing with uncertainty or vagueness in many problems.

Definition 1: Fuzzy Set

Let Y be a non-empty set and a subset of real numbers. A fuzzy set A, in Y, the universe of discourse, is characterized by its membership function, $\mu_A : Y \to [0, 1]$. This set A in Y is defined as a set of ordered pairs (Zadeh, 1965);

$$A = \{y, \quad \mu_A(y) / y \in Y \} \qquad (2.1)$$

If $Y = \{y_1, y_2, ...y_n\}$, is a finite set and A is a fuzzy set in Y then,

$$A = \mu_1 / y_1 + \mu_2 / y_2 + ...\mu_n / y_n \qquad (2.2)$$

Where μ_n is the grade membership of y_n and, $n = 1,2,3.....$

2.2.3 Fuzzy Time Series (FTS) and Fuzzy Logical Relationship (FLR)

Fuzzy time series was first defined by Song and Chissom (1993). This was developed from the notion of traditional crisp time series to deal with incomplete and vague series of data ordered in time sequence by applying fuzzy logic. Fuzzy time series differs from traditional time series in that observations are linguistic (vague and imprecise) values instead of real numeric values (Li *et al*, 2008). Fuzzy time series forecasting can be defined as the application of fuzzy mathematics to model and predict the future from a time series of linguistic (imprecise) historical observations. Thus, it is a mathematical forecasting model that assumes information needed to generate forecasts is contained in a time series of linguistic data. This model consists of three steps; fuzzification of observations, establishing fuzzy relationship, and defuzzification (Yolcu *et al*, 2009). Fuzzification is the process of converting crisp data into fuzzy (linguistic) values by identifying variations in the crisp data. In the context of forecasting, it is the process of identifying associations between historical values in the data set and the fuzzy sets defined on the universe of discourse (Poulsen,

2009). Defuzzification is the process of converting linguistic values to crisp data. In the context of forecasting, it is the process of determining the forecasted (crisp) values from fuzzified historical values (fuzzy relationships).

Definition 2: Fuzzy Time Series

Let $Y(t) = \ldots 0,1,2,\ldots$ a subset of real numbers, be the universe of discourse on which $f_i(t) = 1,2,3,\ldots$ are defined.

If $F(t)$ is a collection of $f_i(t) = 1,2,3,\ldots$, then $F(t)$ is called a fuzzy time series on $Y(t) = \ldots 0,1,2,\ldots$.

Definition 3: Fuzzy Logic Relation (FLR)

If there exist a fuzzy logic relationship $R(t-1,\ t)$ such that $F(t) = F(t-1) \circ R(t-1,\ t)$ where $\circ$ represents an operator, then $F(t)$ is said to be caused by $F(t-1)$.

The relationship between $F(t)$ and $F(t-1)$ is denoted by;

$$F(t-1) \rightarrow F(t) \qquad (3)$$

If $F(t-1) = A_i$ and $F(t) = A_j$, then $A_i \rightarrow A_j$.

Definition 4: N-Order Fuzzy Relation

Let $F(t)$ be a fuzzy time series. If $F(t)$ is caused by $F(t-1), F(t-2), \ldots, F(t-n)$ then this fuzzy relationship is represented by

$$F(t-n), \ldots, F(t-2), F(t-1) \rightarrow F(t). \qquad (4)$$

This is called an n-order fuzzy time series. This was first introduced by Chen (2002). Increasing the order of a model from $n-1$ to n does not necessarily result in high accuracy rates for all data cases (Poulsen, 2009).

Definition 5: Time – Invariant Fuzzy Time Series

Suppose $R(-1,\ t)$ is a first order model of $F(t)$. If for any t, $R(-1,\ t) = R(-2,\ t-1)$, then $F(t)$ is called a time – invariant fuzzy time series otherwise it is called a time – variant fuzzy time series.

Definition 6: Fuzzy Logic Relationship Group (FLRG)

Relationships with the same fuzzy set on the left hand side can further be grouped into a relationship group. Relationship groups are also referred to as fuzzy logical relationship groups (FLRG). Suppose that: $A_i \to A_{j1}, A_i \to A_{j2}, \ldots\ldots A_i \to A_{jn}$, then, they can be grouped into a relationship group as follows: $A_i \to A_{j1}, A_{j2}, \ldots\ldots, A_{jn}$. The same fuzzy set cannot appear more than once on the right hand side of the relationship group.

2.2.4 Universe of Discourse

The range of possible values fuzzy sets (linguistic variables) can take is called the universe of discourse. In fuzzy time series (FTS), the universe of discourse, $Y(t)$, can be different at different times (Song and Chissom, 1993).

When forecasting with fuzzy time series (FTS) using real (crisp) historical data, this data must first be converted to fuzzy sets. To define fuzzy sets on the historical data, the universe of discourse, $Y(t)$, must be defined. And, to define the universe of discourse, $Y(t)$, the steps are as follows (Song and Chissom, 1993):

i. Find the minimum, $D_{\min}$, and maximum, $D_{\max}$, values of the historical data.

ii. Then, define the universe of discourse, $Y(t)$, as:

$$Y(t) = [D_{\min} - b_o, \quad D_{\max} + b_1] \qquad (2.5)$$

Where;

b_0 and b_1 are two positive numbers (buffers).

These buffers, according to Song and Chissom (1993), are arbitrarily assigned to adjust the lower and upper bounds of the range.

2.2.5 Fuzzy Set Groups (FSGs)

In conventional fuzzy time series (FTS), fuzzy logical relationship groups (FLRGs) identified, after historical data have been fuzzified, are not unique for some values. This implies that unique observations in a partition will have the same forecasted outputs which cause some mismatches between forecasts and actual historical data. These mismatches affect forecasting accuracy. Fuzzy set groups (FSGs) are established against fuzzy relationship groups (FLRGs) to give the historical data a unique set of fuzzy relations (sub patterns) which subsequently are converted to "if – then" statements. The fuzzy set group (FSG) algorithm is implemented as follows:

Step 1: combine consecutive fuzzy sets in the pairwise manner $\{F(t-2), F(t-1)\} = \{A_{i,t-2}, A_{i,t-1}\}$ to create second – order fuzzy set groups (FSGs).

Step 2: if fuzzy set groups are disambiguated, then stop; otherwise extend only ambiguous fuzzy set groups to third – order fuzzy set groups in the form $\{F(t-3), F(t-2), F(t-1)\} = \{A_{i,t-3}, A_{i,t-2}, A_{i,t-1}\}$ to produce disambiguated fuzzy set groups.

Step 3: continue the extension process until disambiguated fuzzy set groups (FSGs) are obtained.

Ultimately, the goal of the fuzzy set group (FSG) algorithm is to obtain fuzzy relationships (FSGs) free of ambiguities (Poulsen, 2009). Ambiguities occur if two or more fuzzy set groups (FSGs) contain the same combination of elements (Eleruja *et al*, 2012).

2.2.6 Data Mining and Clustering

Data mining is the process of extracting previously unknown and most valuable information from a data set. It is a collection of techniques and tools for handling large amounts of information (Jafar and Sivakumar, 2013).

Data clustering is a popular data mining technique used in data analysis. This is an unsupervised classification method of finding classes or subgroups in a data set. Data with most similarities belong to same cluster (class) while data with most dissimilarity belong to different classes. Clustering techniques have the ability to discover different cluster shapes, handle different data types and deal with noise and outliers (Jafar and Sivakumar, 2013). Noise, in this context, is an observation whose membership degrees to all clusters are very low. Partitioning clustering is one of the major categories of clustering proposed by researchers. Partitioning clustering of a data set $X_t (=1,2,3,...,n)$ partitions X into $2 \le c < n$ subgroups (classes) by optimizing a certain objective function. Each subgroup represents the "natural" substructure in X (Nikhil *et al*, 2005). In partition clustering, previous knowledge about the number of clusters is required. Common types of partitioning clustering used by researchers, in many real world applications, are K-means and fuzzy clustering (or fuzzy c-partitioning). Partitioning algorithms determine a number of partitions even when such partitions do not properly classify a dataset based on similarity and dissimilarity.

K-means is a hard clustering technique in which each data point belongs to only one cluster. This method of partitioning is not suitable for real world data sets in which there are no definite boundaries between the clusters (Jafar and Sivakumar, 2013).

Fuzzy clustering is a set of membership degrees (or label vectors) u_k, that partitions data set, X, into fuzzy subsets (clusters) by minimizing an objective function. It permits one piece of data to belong to two or more clusters (Jafar and Sivakumar, 2013). Fuzzy clustering algorithm partitions data sets into fuzzy clusters with varying degree of membership values that lies within 0 and 1.

Set of membership degree is an array of $c \times n$ matrix. In the context of forecasting, fuzzy clustering is defined as the partitioning and fuzzifying of data sets based on a specific objective function and distance measure. The objective function is given by;

$$J_\beta(X,U,V) = \sum_{i=1}^{c}\sum_{t=1}^{n} \mu_{it}^{\beta} d_{it} \qquad (2.6)$$

Where;

d_{it} is a distance measure.

$\beta > 1$ is the fuzziness index (or weighting exponent) used to tune out noise in the data set.

n = the number of observations, x_t .

$c \geq 2$ is the number of clusters (partitions) in the set.

X is the observations matrix.

U is the partition matrix.

V is the cluster matrix.

μ is the membership grade.

2.2.6.1*Distance Measure*

This measure is required in all clustering techniques. A distance measure is a proximity measure that reflects the similarity (or dissimilarity) between data vectors in a given cluster. The following are some distance measures used in data clustering:

1. Euclidean (L_2 norm) distance

$$d^2(x_t, v_i) = \|x_t - v_i\| = \sum_{t=1}^{n} (x_t - v_i)^T (x_t - v_i) \quad (2.7)$$

2. Chebyshev (L_∞ norm) distance

$$d(x_t, v_i) = \underset{i=1,2,\ldots,n}{Max} |x_t - v_i| \quad (2.8)$$

There is no common distance measure which can be best suited for all clustering applications (Jafar and Sivakumar, 2013). Traditionally, Euclidean distance is the distance measure for fuzzy C-means clustering.

2.2.7 Fuzzy C-Means(FCM) Clustering

When Euclidean (L_2 norm) distance is used as distance measure in fuzzy clustering, the clustering algorithm is called fuzzy c-means (FCM) clustering. Fuzzy C-means (FCM) algorithm is one of the most widely used fuzzy clustering models first proposed by Dunn (1974) and then generalized by Bezdek (1981). Given an unlabelled data set, $X \in R^n$, FCM partitions data sets into clusters

v_i $(=1,2,3....,c)$ and fuzzify data sets by minimizing the least-squared error (cluster error) objective function:

$$J_\beta(X,U,V) = \sum_{i=1}^{c}\sum_{t=1}^{n} \mu_{it}^{\beta} d^2(x_t; v_i) = \sum_{i=1}^{c}\sum_{t=1}^{n} \mu_{it}^{\beta} \times \|x_t - v_i\| \qquad (2.9)$$

Where;

$d^2(x_t; v_i)$ is the Euclidean distance measure.

$\beta > 1$ is the fuzziness index (or weighting exponent) used to tune out noise in the data set.

n = the number of observations, x_t .

$c \geq 2$ is the number of clusters (partitions) in the set.

An iterative minimization algorithm is used to minimize J_β under the following constraints (Aladag *et al*, 2012):

$$0 \leq \mu_{it} \leq 1, \qquad \forall i,t \qquad (2.10)$$

$$0 < \sum_{t=1}^{n} u_{it} \leq n, \qquad \forall i \qquad (2.11)$$

$$\sum_{i=1}^{c} u_{it} = 1, \qquad \forall t \qquad (2.12)$$

In each iteration, the calculations of v_i and x_t are achieved using equations (2.13) and (2.14) (Aladag *et al*, 2012):

$$v_i = \frac{\sum_{t=1}^{n} \mu_{it}^{\beta} x_t}{\sum_{t=1}^{n} \mu_{it}^{\beta}} \quad (13)$$

$$\mu_{it} = \frac{1}{\sum_{i=1}^{c} \left(\frac{d(x_t; v_i)}{d(x_t; v_k)} \right)^{\frac{2}{\beta-1}}} \quad (14)$$

Forecasting using fuzzy C-means clustering (FCM) algorithm is made up of the following steps (Cheng *et al*, 2008):

Step 1: Apply appropriate fuzzy C-means (FCM) clustering procedure to time series data.

Step 2: Rank cluster centres and fuzzify raw data.

Step 3: Induce the fuzzy relationship and fuzzy relationship groups.

Step 4: Forecast and defuzzify possible outcomes based on fuzzy relation.

The first step utilizes fuzzy C-means clustering (FCM) algorithm to identify partitions and calculates memberships with respect to the measured cluster centres. The next step ranks the values of the cluster centres, v_i, in ascending order. The ranks are used to define the clusters as ordered linguistic variables $L_r (r = 1,2,3,....,c)$ (Cheng *et al*, 2008).

When compared to Chen's (1996) model, each cluster centre represents the mid-point of a crisp interval. Thus, cluster centres represent unequal partitions while membership values represent the degree of closeness (belonging) to these mid-points (partitions).

2.2.8 Cluster Validity Index

Partitioning algorithms, which are unsupervised data mining techniques, need to know the number of clusters (partitions) to look for. Thus, finding the right number of clusters is important in discovering distribution of patterns and interesting correlations in data sets. Finding an optimal number of clusters that can best describe the data structure (especially for high dimensional data) is called cluster validity (Wang and Zhang, 2007). Cluster validity indices are used to evaluate the performance of fuzzy clustering (Jafar and Sivakumar, 2013). Among the cluster validity functions (indices), applied for fuzzy C-means algorithm are (Jafar and Sivakumar, 2013; Xu *et al*, 2013):

1. Partition Coefficient (PC):

$$PC = \frac{1}{n}\sum_{i=1}^{c}\sum_{t=1}^{n} u_{it}^{2} \qquad (2.15)$$

2. Partition Entropy (PE):

$$PE = -\frac{1}{n}\left[\sum_{i=1}^{c}\sum_{t=1}^{n} u_{it}\log u_{it}\right] \qquad (2.16)$$

3. Mean Squared Error (MSE):

$$MSE_c = \frac{1}{n}\sum_{i}^{c}\sum \left\| x_t - v_i \right\| \qquad (2.17)$$

x_t is the set of data belonging to the i^{th} cluster.

MSE is one of the very important indices. Based on this concept, when a cluster number is fixed, a good clustering algorithm should place cluster centroids in a way that it reduces the distances to data point as much as possible (Xu *et al*, 2013).

2.2.9 Particle Swarm Optimization (PSO)

This is a stochastic optimization technique that is robust and fast in optimizing continuous non-linear functions. It was first introduced by Kennedy and Eberhart (Kennedy and Eberhart, 1995). It is a population – based evolutional algorithm which can efficiently search the near optimal solution of any kind of optimization problem. Just like most population – based algorithms are inspired by nature's evolution; it mimics the behaviour of bird flocking or fish grouping in searching towards the place where food is located. As search continues, the birds move towards the place where food is by following the bird which is closest to the food (Poulsen, 2009).

In the PSO algorithm, "particle swarms" represents bird flocks and "solution" represents bird's position. These particles are randomly initialized and allowed to search for the best solution with respect to the corresponding best solution of an optimization problem in the virtual search space. Any particle remembers its personal best position it has passed so far when it moves to another position (Kuo *et al*, 2009).

During PSO implementation, a set of randomly generated particles, n, which are candidate solutions, are used to initialize the process. Then, an iterative search process is set in motion to improve the set of current solutions. The moving particle adjusts (updates) its candidate solution according to equation (2.18) and (2.19) below:

$$V_{i+1} = w \times V_i + C_1 \times Rand_1 \times (X_p - X_i) + C_2 \times Rand_2 \times (X_g - X_i) \quad (2.18)$$

$$X_{i+1} = X_i + V_{i+1} \quad (2.19)$$

Where;

- V_i is the velocity of the particle *i* which is limited to $[-V_{\max}, V_{\max}]$. *V* max is a predefined constant.
- *w* is the inertial weight factor.
- X_i is particle current position.
- X_p is the particle personal best position.
- X_g is the particle global (group) best position.
- C_1 and C_2 denotes self-confidence ("cognition part") and social-confidence ("social part") coefficients, respectively.
- $Rand_1$ and $Rand_2$ are randomly generated numbers between 0 and 1.

$V_i$defines the rate of change in position for particle *i*. In equation (2.18),$V_{i+1}$and $V_i$represents the particles new and previous velocity, respectively. During any iteration, particle evaluates the new velocity it should fly with towards a new position. The particle flies toward a new position according to (2.19). In equation (2.19), X_{i+1}represent the particles new current position.

C_1 and C_2 are user defined constants that say how much the particle is directed towards good positions. They affect how much the particle's local and global best influence its movement. Generally, C_1 and C_2 are set to 2.

The performance of each particle is measured according to an optimization problem (fitness function) to be solved. A fitness value is associated to each particle to be optimized, and the movement of each particle is directed by a randomly generated velocity parameter.

During any iteration, new solutions are proposed by each particle which is individually evaluated against:

i. The particle's own personal best solution found in any proceeding iteration and,

ii. The global best solution currently found by any particle in the swarm. (Poulsen, 2009).

The particle's velocity, personal and the global best positions are only updated when the particle finds a position better than its previous position.

In terms of ease of implementation, processing time and quality of solution in optimizing non – linear functions, particle swarm optimization (PSO) generally performs better than other well – known evolutional algorithms (EA) like; genetic algorithm (GA), memetic algorithm (MA) and ant colony optimization (ACO) (Elbeltagi *et al*, 2005). Thus, computing particle swarm optimization (PSO) is inexpensive since its memory and CPU speed requirement is low.

2.2.10 Defuzzification Operator

This operator is mainly used for defuzzifying linguistic variable observations associated with the satisfaction of multiple criteria. The defuzzified output is the weighted sum of the historical fuzzy sets values, a_{i-1}, from time $(-n)$ to (-1), where n depends on the time series span defined by (Poulsen, 2009);

$$Y(\) = \sum_{i=1}^{n} (a_{t-1} \times w_i) \qquad (2.20)$$

$w_i \in [0, 1]$, is the strength of fuzzy logical relationship between past fuzzy values (inputs) and future forecasts (outputs). The closer w_i is to 1 the stronger the relationship (Poulsen, 2009).

2.2.11 Erlang – based Voice Traffic

Voice traffic measurement of telecommunication networks are carried out to analyze traffic patterns and determine the necessary resources to handle it optimally. Erlang is the unit of voice traffic volume that represents the continuous use of one voice path per hour (Margaret, 2007). n Erlang is equivalent to the continuous use of n telephone lines per hour. It is non – linear and dynamic with time. Hence, Erlang observations are real numbers of time series in a certain period. Erlang model apply to wherever users arrive more or less random to receive exclusive service from any one of a group of service providing elements (Eleruja, 2012). This makes it suitable for modeling traffic of a mobile phone call centre.

A mobile phone call centre is a telephone network system designed to effectively connect mobile subscribers with telephone operators to provide services such as: assistance to activate (or change) services, information about packages and resolving billing problems. These services are handled by agents, who reside in remote places. The core of a call centre is the Automatic Call Distributor (ACD) (Chromy *et al*, 2011). This system processes large number of incoming subscriber voice calls by distributing them to available agents based on first-come-first-served. It queues calls when agents are busy with other customers who seek the same services and keeps reports of call centre's network parameters.

2.2.12 Performance Measures

These are statistical inference tools used to measure or summarize how well a time series data is forecasted by a forecasting model. Two most popular key performance indices (KPIs) used to evaluate the performance of fuzzy time series (FTS) forecasting model, in the literature, are: mean squared error (MSE) and mean absolute percentage error (MAPE).

2.2.12.1 Mean Square Error (MSE)

This is a performance measure that is a combination of both average accuracy and precision of a forecasting model. The average accuracy of the forecasting model is known as the bias of the model. The bias is a measure of how far the mean of the forecasts is from the actual time series (Woschnagg and Cipan, 2004). Mathematically;

$$Bias(forecast_t) = E(forecast) - actual_t \quad (2.21)$$

$E(forecast_t)$ is the mean (expected) value of the forecast.

The precision (reliability) of the forecasting model is called the variance. This measures the uncertainty around the most likely forecast. Mathematically;

$$Var(forecast) = E[forcast_t - E(forecast_t)]^2 \quad (2.22)$$

A good forecasting model should have low bias and as well as low variance (Tamhane and Dunlop, 2010). The mean square error of a forecasting model is the expected value of the squared error (SE; quadratic loss) in forecasting a time series. Mathematically;

$$SE = (forecast_t - actual_t)^2 \quad (2.23)$$

$$MSE = E[forecast_t - actual_t]^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(forecast_t - actual_t)^2 \quad (2.24)$$

Let $forecast_t = Y_t$ and $actual_t = A_t$.

$$\Rightarrow MSE = E[Y_t - A_t]^2$$

$$= E\left[Y_t - E(Y_t) + E(Y_t) - A_t\right]^2$$
$$= E\left[Y_t - E(Y_t)\right]^2 + E\left[E(Y_t) - A_t\right]^2 + 2E\left[Y_t - E(Y_t)\right]\left[E(Y_t) - A_t\right] \quad (2.25)$$

But;

$$E\left[Y_t - E(Y_t)\right] = 0 \quad (2.26)$$

$$\Rightarrow MSE = E\left[Y_t - E(Y_t)\right]^2 + E\left[E(Y_t) - A_t\right]^2 \quad (2.27)$$

Substituting equations (2.26) and (2.27) into (2.25);

$$\therefore \ MSE = Var(Y_t) + Bias(Y_t)^2 \quad (2.28)$$

Thus, the means square error is the sum of the variance and bias – squared term. MSE is useful in detecting large errors whose negative consequences are proportionately much bigger than equivalent smaller ones. MSE are highly influenced by outliers (extreme values) and are scale dependent (Shcherbakov, 2013).

2.2.12.2 Mean Absolute Percentage Error (MAPE)

This is an error measure that is expressed in percentage of the actual observation. It is an accuracy measure that provides an intuitive way of judging errors. It expresses the performance of a forecasting model regardless of the scale at which the measure is computed (Birek *et al*, 2014). Mathematically;

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{\left|forecast_t - actual_t\right|}{actual_t} \times 100\% \quad (2.29)$$

n is the number of forecasts.

The lower the MAPE of a competing forecasting model, the better the forecast (Birek *et al*, 2014). Appearance of division by zero when the actual value equal to zero is a shortcoming of MAPE (shcherbakov *et al*, 2013). This makes MAPE only suitable when actual data is preprocessed, to remove zero records. MAPE is also known as mean absolute percentage deviation (MAPD). It can also be defined as the expected value of the absolute error (AE) relative to the actual observations. Mathematically;

$$MAPE = E\left[\frac{|forecast_t - actual_t|}{actual_t}\right] \qquad (2.30)$$

E[ ] is the expected value.

2.2.13 Programming Language

A programming language is a set of rules, symbols and special words used for commanding a computer to perform some set of instructions (computations or algorithms). These could be low – level, high – level, object – oriented or machine language. Object – oriented languages are becoming the most dominant ones now.Among the most popular object – oriented programming languages (Hao, 2010) are: C++, Java and C# (these were enhancement of the original C language). A set of rules, symbols and special words are used to construct a program.

2.2.13.1 C Programming Language

C is a structured programming language with easy-to-use syntax compared with earlier programing languages like Pascal and FORTRAN. It was developed by Dennis Ritchie in 1970 at Bell Laboratories of USA. This high level programming language was created to design the UNIX operating system and allow busy programmers get things done. C programming language cannot

support the development and maintenance of large and complex programs (Schildt, 2009). Its program requires recompilation when run on different environments. Thus, programs were not portable.

2.2.13.2 C++ Programming Language

C++ (C with classes) is an object oriented programming language developed by Bjarne Stroustup in the early 1980's for UNIX environment. C programming language was enhanced by adding object oriented features. Object oriented features tie data closely to the function that operates on it, and protects it from accidental modification. This enhancement improved the quality of codes produced, thus making reusable code easier to write (Hao, 2010). Programs written in this language are not portable because they require recompilation when run on different environment.

2.2.13.3 Java Programming Language

Java is a programming language designed for programming home appliances and Internet applications. Java was first developed in 1991 by James Gosling and his team at Sun Microsystems (Hao, 2010). Its syntax and philosophy were derived from C++. This language solved the issue recompilation of programs by first converting programs to an intermediate language (*bytecode*) which can then be executed by any environment with Java Virtual Machine (JVM) (Schildt, 2009). Java Virtual Machine also introduced Internet security since it had full control over program.

2.2.13.4 C# Programming Language

C# (c-sharp) is a pure object – oriented and component oriented programming language whose chief architect was Ander Hejlsberg. It was developed by Microsoft within its .NET initiative and made its appearance in the year 2000. C# has the ability to work in secured and mixed language environment. Since no one paradigm solves all problems in the most efficient way, C# mixed

language programming ability allows the intermixing of different programming paradigms to solve complex problems. Its portability was important for programmers who were familiar with C and C++. C# also introduced more features to deal with objects and directly supported Windows operating systems (Schildt, 2009).

2.3 REVIEW OF SIMILAR WORKS

This section presents a review of literature with respect to fuzzy time series forecasting.

The first fuzzy time series model was developed by Song and Chissom (1993). This fuzzy time series was used to forecast enrolment in university of Alabama. The time – invariant model proposed comprises seven steps; define the universe of discourse, partition the universe of discourse into equal length of intervals, define fuzzy sets on the universe of discourse, fuzzification of historical data, establish fuzzy relational matrix, forecast using $F(t) = F(t-1) \circ R$, defuzzify the forecasted results. The model, significantly, showed that fuzzy time series is more suitable in modeling uncertainties in observations than traditional statistical approaches. Its limitations are; unnecessarily high computational overheads of performing max – min composition operations, subjective choice of the length of interval, and determination of the universe of discourse.

Chen (1996) proposed an efficient first – order fuzzy logical relationship (FLRs) to simplify the complexity in establishing forecast. This model reduced the computational overheads of the forecasting process in the pioneer fuzzy time series model. It also showed a higher forecasting accuracy than the Song and Chissom (1993) models. The model was made up of the following steps; the first four steps of the Song and Chissom model identify fuzzy relationships (FLRs), establish fuzzy logical relationship groups (FLRGs), defuzzify the forecasted output. This work showed that fuzzy logical relationship groups (FLRGs) and simple arithmetic operations can reduce

computational overheads and improve forecasting accuracy in fuzzy time series (FTS) forecasting than max – min operation. The subjective definition of the universe of discourse and ineffective partitioning remained unresolved issues. Also, the parametric form of the membership function was assumed to be known and was not satisfactorily utilized in defining fuzzy relations.

Huarng (2001) investigated the impact of effective interval lengths on forecasting results. This work was the first to show that the length of interval indeed affected the performance of the forecast results. Hence, its determination was critical for forecasting in fuzzy time series. Two approaches were subsequently proposed to determine the effective interval length of observations; distribution and average – based lengths. Each approach was incorporated into the Chen's (1996) model to improve forecasting. The new models provided better overall forecasting results than previous models in which interval lengths are randomly selected. Parametric form of memberships and their underutilization still remained unresolved issues.

Huarng and Yu (2006) introduced the ratio – based interval. In this approach interval length was increased by a statistically distributed ratio. The forecasting results showed that unequal interval lengths improved forecasting accuracy than equal interval lengths. Although, interval length was objectively determined, this was not an optimal approach.

Mu'azu and Adeola (2008) investigated the effect of varying interval lengths on the fuzzy time series forecast model and indeed ascertained the result of Huarng (2001). The work, though not an optimal approach, also showed that odd number interval lengths produced better results than even number interval lengths; though the interval lengths were randomly selected. This work failed when no previous information about the membership functions is known.

Cheng *et al* (2008) proposed the first hybrid fuzzy time series model used to learn memberships and the internal structure of a data set by combining fuzzy C-means (FCM) clustering with fuzzy time series. Fuzzy C-means clustering was used to objectively partition the universe of discourse, define memberships that best explain the unknown structure of the data set, and eliminate the need to define the universe of discourse. The forecasting results showed that the proposed hybrid model, when applied on a one – attribute data set, outperformed previous models. It also showed that the hybrid model could also forecast multi – attribute data sets effectively. However, there was no mechanism for optimizing the defuzzification phase, which could further improve the forecasting accuracy. Also, the effect of high order of the model was not investigated.

Kuo *et al* (2009) proposed a hybrid fuzzy time series forecasting model based on particle swarm optimization (PSO). PSO was used in the fuzzification phase to determine the length of interval and the proper content of forecast rules. This was the first time FTS was combined with PSO for forecast. When applied to forecast enrolments into the University of Alabama, the results showed that the hybrid model had higher accuracy than some previous models. It also showed that PSO algorithm was more effective than genetic algorithm (GA) in searching for appropriate intervals in fuzzy time series. Although, it was an optimal approach to defining the length of intervals, the proposed model's subjective definition of the upper and lower bounds of the universe of discourse was a limitation.

Poulsen (2009) proposed a new hybrid fuzzy time series based on trapezoid fuzzification approach and particle swarm optimization (PSO). This is the first work in which fuzzy set groups (FSGs) were introduced in place of the conventional fuzzy logical relationship groups (FLRGs) to produce disambiguous forecasting rules. PSO was then applied to tune forecasting rules (FLRGs) in the defuzzification process. The significance of this work was that since rules were unique andforecasts

were unique for observations that belonged to the same interval partition. The limitations of this work included computational complexities that arose from implementing Trapezoidal fuzzification approach analytically and the fact that the effect of the reversed weight was not investigated. Another drawback was that parametric form of memberships function (trapezoid memberships) for the data set was assumed to be known.

Aladag *et al* (2012) proposed a hybrid fuzzy time series based on fuzzy C-means (FCM) clustering and particle swarm optimization (PSO). FCM was used in the fuzzification phase to objectively partition the universe of discourse and define memberships in the fuzzy relational matrix. PSO was used to optimize these memberships. This model had the advantage of using membership values of observations to define fuzzy relations. Therefore, information loss was minimal. It was also a objective method because the number of interval, interval lengths, and degree of memberships were not subjectively chosen. Also, forecasting accuracy was higher than previous models. The limitation of this hybrid model included computational complexities that may arise from optimizing fuzzy relational matrix. Also, forecasts were not unique for observations in the same interval partition.

Eleruja *et al* (2012) applied trapezoidal fuzzification approach (TFA) and particle swarm optimization (PSO) in fuzzy time series (FTS) forecasting. TFA was employed in the fuzzification phase of FTS model to objectively partition the universe of discourse. PSO and aggregation were utilized to reduce mismatch between the forecasted and actual data. The hybrid model addressed the issue of analytical complexity in the defuzzification phase by developing PSO using C#. When applied to forecasting maximum temperature data of Zaria, the result obtained indicated that forecast were unique and the technique improved forecasting accuracy. High computational requirement of the TFA which consequently implied high cost of computation was a limitation of

this hybrid model. Another drawback was that parametric form of membership function for the data set was assumed to be known.

In essence, this work proposes a hybrid fuzzy time series forecasting model to minimize the limitations of the mentioned models in order to further improve the forecasting accuracy. It will use fuzzy C-means clustering – based fuzzy time series to objectively determine the interval lengths, calculate the degree of membership, partition the universe of discourse and, eliminate the need to define the universe of discourse in the fuzzification phase. Then, it will use particle swarm optimization to tune FSGs in the defuzzification phase. To reduce computational cost, both FCM and PSO algorithms will be implemented using a GUI developed using C#.

C# programing language is adopted above other popular object - oriented programming languages (such as; Java and C++) because of its pure object oriented programming (OOP) features which can fully be integrated with Windows platforms (Operating Systems, OS); one of the most widely used platforms in the world. The model is then validated using voice traffic observations from Airtel, Abuja Call Centre. In addition, the effect of the reversed weights on the defuzzification process was also investigated.

CHAPTER THREE

MATERIAL AND METHODS

3.1 INTRODUCTION

This chapterfocuses on discussing the steps of the methodology and their implementation. The implementation of the methodology is a detailed explanation of the steps itemized in the methodology.

3.2 DATA COLLECTION AND PROCESSING

A three (3) month Erlang-based voice traffic data set is collected from the daily inbound calls of Airtel, Abuja, from 1 December, 2012 – 28 February, 2013 (shown in Appendix A). To minimize redundant information, records with missing values are deleted. Approximately, 67% of the data set is used for training the hybrid model while the remaining 33% is used for testing the model. Table 3.1 shows the training data set.

Table 3.1: Erlang Traffic Data Set for Training (1 December, 2012 – 29 January, 2013)

Date	Traffic	Date	Traffic	Date	Traffic	Date	Traffic	Date	Traffic
01/12/12	170.232	**13/12/12**	169.367	**25/12/12**	219.480	**06/01/13**	90.319	**18/01/13**	184.075
02/12/12	160.343	**14/12/12**	156.205	**26/12/12**	259.722	**07/01/13**	153.194	**19/01/13**	153.452
03/12/12	150.965	**15/12/12**	152.556	**27/12/12**	291.306	**08/01/13**	183.423	**20/01/13**	157.500
04/12/12	163.151	**16/12/12**	176.442	**28/12/12**	300.841	**09/01/13**	188.883	**21/01/13**	150.486
05/12/12	179.860	**17/12/12**	209.617	**29/12/12**	308.587	**10/01/13**	134.050	**22/01/13**	187.741
06/12/12	176.526	**18/12/12**	164.002	**30/12/12**	292.317	**11/01/13**	139.442	**23/01/13**	141.908
07/12/12	155.550	**19/12/12**	168.123	**31/12/12**	301.779	**12/01/13**	187.514	**24/01/13**	143.357
08/12/12	154.571	**20/12/12**	189.797	**01/01/13**	315.685	**13/01/13**	221.600	**25/01/13**	166.878
09/12/12	160.874	**21/12/12**	202.416	**02/01/13**	291.862	**14/01/13**	232.545	**26/01/13**	166.439
10/12/12	148.72	**22/12/12**	131.758	**03/01/13**	211.912	**15/01/13**	260.859	**27/01/13**	159.809
11/12/12	138.522	**23/12/12**	166.487	**04/01/13**	175.319	**16/01/13**	319.248	**28/01/13**	151.401
12/12/12	159.171	**24/12/12**	202.214	**05/01/13**	93.802	**17/01/13**	158.539	**29/01/13**	191.997

3.3FUZZIFICATION MODULE

The steps in the fuzzification module are implemented as follows:

3.3.1 Coding fuzzy C-Means (FCM) Clustering Algorithm in C#.

The aim of integrating fuzzy C-means clustering into the fuzzification phase of fuzzy time series (FTS) forecasting model is to empirically determine the unknown, valid and useful unequal partitions and membership degrees for any type of data set. This will also eliminate the need to define the universe of discourse. This algorithm is implemented in C# to reduce the computational cost of fuzzification. The flowchart for the fuzzy C-means algorithm is shown in Figure 3.1.

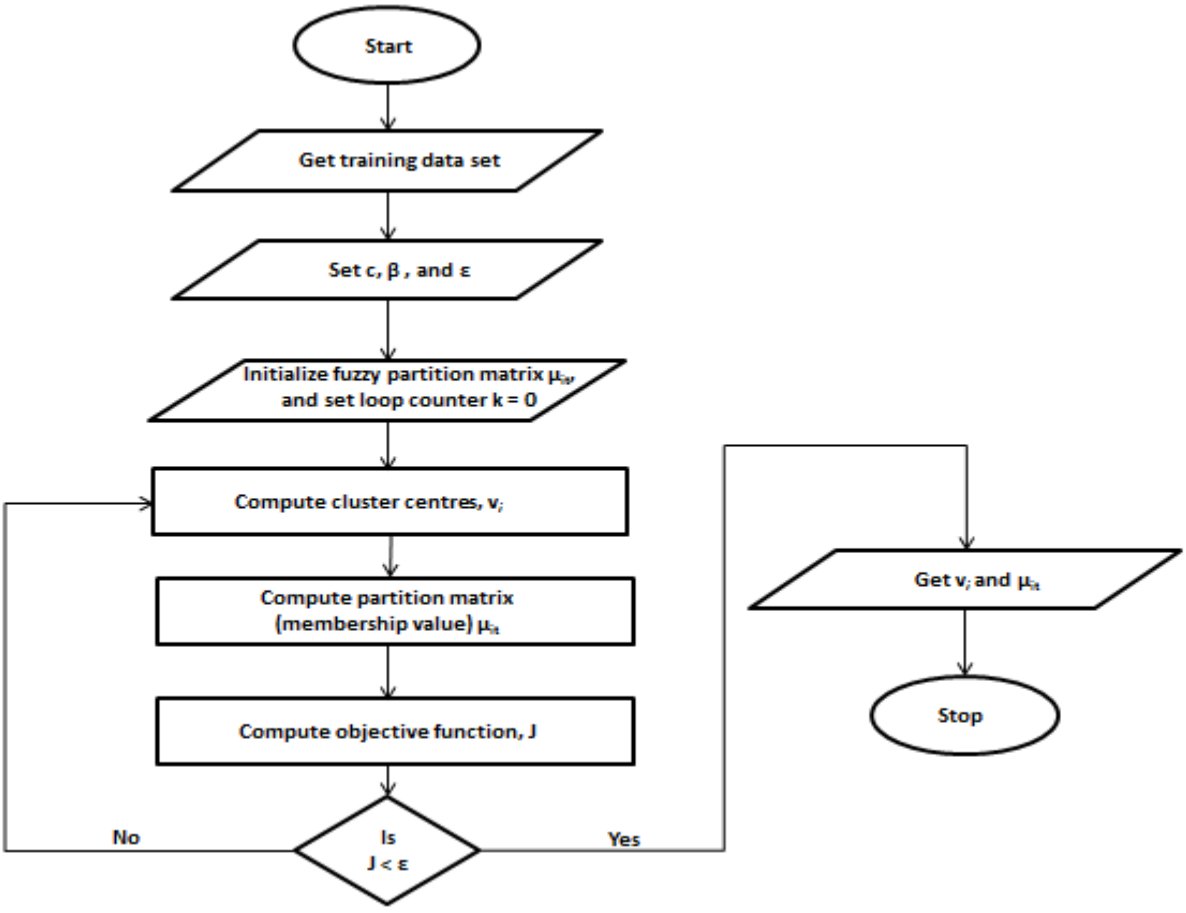


Figure 3.1: Flowchart of the Fuzzy C-Means Algorithm.

To build the fuzzy C-means code, the C# code editor of Microsoft Visual Studio Express 2012 for Windows desktop is the integrated development environment (IDE) utilized. A Graphical User Interface (GUI) is created to allow options such as number of clusters, maximum iterations and precision to be set and changed. File menu, start and cancel buttons are also available on the GUI to input voice traffic data, run the clustering code and stop the code from running, respectively. A status bar is also available on the GUI to display progress, number of iterations required to optimize the objective function, duration and precision. The "duration" is a function of processing capacity of the computer used.

The pseudo-code for the clustering algorithm is as follows:

i. **Set the number of clusters, c, the fuzzy index, β, and the stopping conditions (number of iterations and precision) for the objective function, J.**

The snippet of the program that allows the number of clusters and stopping conditions for the objective function to be set is as follows:

```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
    {
      backgroundWorker.ReportProgress(0, "Working...");

      int numClusters = (int)numericUpDown2.Value;
      int maxIterations = (int)numericUpDown3.Value;
      double accuracy = (double)numericUpDown4.Value;
     ..............
     }
```

This is a worker thread found in the "Form1" class of the complete code. The worker thread allows changes to be made to number of clusters and stopping condition, when it is necessary, through the graphical user interface (GUI).

Fuzzy index β, is a fixed value of 2. The following is the program the snippet of the "Form1" object code that sets the fuzzy index, β.

```
FCM alg = new FCM(points, centroids, 2, this.Data, (int)numericUpDown2.Value);
```

ii. Initialize the fuzzy partition matrix, μ_{it}

This step is achieved by first randomly initializing the cluster centres, v_i, during the first pass of the algorithm, as shown in the following program snippet of the code located in the object (class) "Form1".

```
//Create random points to use as the cluster centroids
        Random random = new Random();
        for (int i = 0; i < numClusters; i++)
        {
           int randomNumber1 = random.Next(this.Data.Length);
           int randomNumber2 = random.Next(this.Data.Length);
           centroids.Add(new ClusterCentroid(randomNumber1,this.Data[randomNumber2]));
        }
```

Then, use the randomly initialized cluster centres to calculate the Euclidean distance (*diff*). Finally, the Euclidean distance (*diff*) is used to initialize the partition matrix (*U[i, j]*), as shown in the snippet of the program code for the object (class) "FCM" as follows:

```
// Iterate through all points to create initial U matrix
        for (int i = 0; i < this.Points.Count; i++)
        {
           ClusterPoint p = this.Points[i];
           double sum = 0.0;

           for (int j = 0; j < this.Clusters.Count; j++)
           {
              ClusterCentroid c = this.Clusters[j];
              diff = Math.Sqrt(Math.Pow(CalculateEuclideanDistance(p, c), 2.0));
              U[i, j] = (diff == 0) ? Eps : diff;
              sum += U[i, j];
           }
        }
```

iii. Set a loop counter, k, to zero

The loop counter is set to zero in the code for the "Form1" class. Below is the snippet of the object program code that implements this step.

```
int k = 0;
```

iv. Compute the cluster centres, v_i

During any loop count execution, a method in the "FCM" class computes cluster centres. Below is the snippet of the program code that computes cluster centres.

```
public void CalculateClusterCentroids()
        {
            //Console.WriteLine("Cluster Centroid calculation:");
            for (int j = 0; j < this.Clusters.Count; j++)
            {
                ClusterCentroid c = this.Clusters[j];
                double l = 0.0;
                c.DataCount = 1;
                c.DSum = 0;
                c.MembershipSum = 0;

                for (int i = 0; i < this.Points.Count; i++)
                {

                    ClusterPoint p = this.Points[i];
                    l = Math.Pow(U[i, j], this.Fuzzyness);
                    c.DSum += l * p.Data;
                    c.MembershipSum += l;

                    if (U[i, j] == p.ClusterIndex)
                    {
                        c.DataCount += 1;
                    }
                }

                c.Data = c.DSum / c.MembershipSum;
             }
```

In the method above, $c.data = v_i$ (cluster centres).

v. For each observation, for each cluster, compute the membership values in the partition matrix

In any count, the following method calculates the membership values.

```
public void Step()
    {
      for (int c = 0; c < Clusters.Count; c++)
     {
       for (int h = 0; h < Points.Count; h++)
       {
         double top;
         top = CalculateEuclideanDistance(Points[h], Clusters[c]);
         if (top < 1.0) top = Eps;
         double sumTerms = 0.0;

         for (int ck = 0; ck < Clusters.Count; ck++)
          {
           sumTerms += top / CalculateEuclideanDistance(Points[h], Clusters[ck]);

          }
        // Then the membership value can be calculated as...
        U[h, c] = (double)(1.0 / Math.Pow(sumTerms, (2 / (this.Fuzzyness - 1))));
         }
      };
      this.RecalculateClusterMembershipValues();
    }
```

In the method above, U[h, c] represents the calculated partition matrix.

vi. Compute J

The method to compute the least square objective function, J, is found in the "FCM" class. The method is shown as follows:

```
public double CalculateObjectiveFunction()
    {
      double Jk = 0.0;

      for (int i = 0; i < this.Points.Count;i++)
      {
        for (int j = 0; j < this.Clusters.Count; j++)
        {
                Jk += Math.Pow(U[i, j], this.Fuzzyness) *
      Math.Pow(this.CalculateEuclideanDistance(Points[i], Clusters[j]), 2);
        }
      }
      return Jk;
    }
```

If the difference in value of J between consecutive iteration is less than the stopping condition, then stop; otherwise set loop counter to one and go to (iii).

vii. Get v_i and μ_{it}

The auto – implemented parameter shown is used to get the value of cluster centres, v_i, and partition matrix, μ_{it}, calculated in step four and five, respectively, if the stopping condition is met.

```
public double Data { get; set; }
```

The complete fuzzy C-means (FCM) clustering code is shown in Appendix I.

3.3.2Applying Time Series Data on Fuzzy C-Means Code

Voice (Erlang C) traffic training dataset is applied on the fuzzy C-means (FCM) clustering algorithm, coded in C#, to obtain its hidden partitions (clusters centres) and membership degrees (partition matrix). For this work, a PC Pentium (R) (CPU 2.10 GHz and 2.00GB RAM) is used. Maximum iteration, stopping criterion (precision) and fuzzy index are set at 100, 0.000001 and 2

respectively. The values of maximum iteration and precision are set to allow the program terminate when any of them are reached. The optimal number of clusters between the ranges of 7 – 14 is determined. This range is chosen based on the fact that small number of partitions affects forecasting rules and accuracy while large number of partitions diminishes use fuzzy time series (FTS) forecasting models above traditional time series forecasting models. A cluster number is discarded immediately if historical observation(s) maximum degree does not lie in a single cluster (partition). This is because of the dissimilarity that should exist between observations in different clusters. Also, Mean squared error (MSE_c) is the validity index used to find the optimal number of clusters within the range, since a good partition should reduce the distances from the data set to cluster centres. Figure 3.2 shows the screenshot for the fuzzy C-Means GUI display when the number of cluster set is 14 whilst Figure 3.3 shows the results windows for the partition matrix and cluster centres.

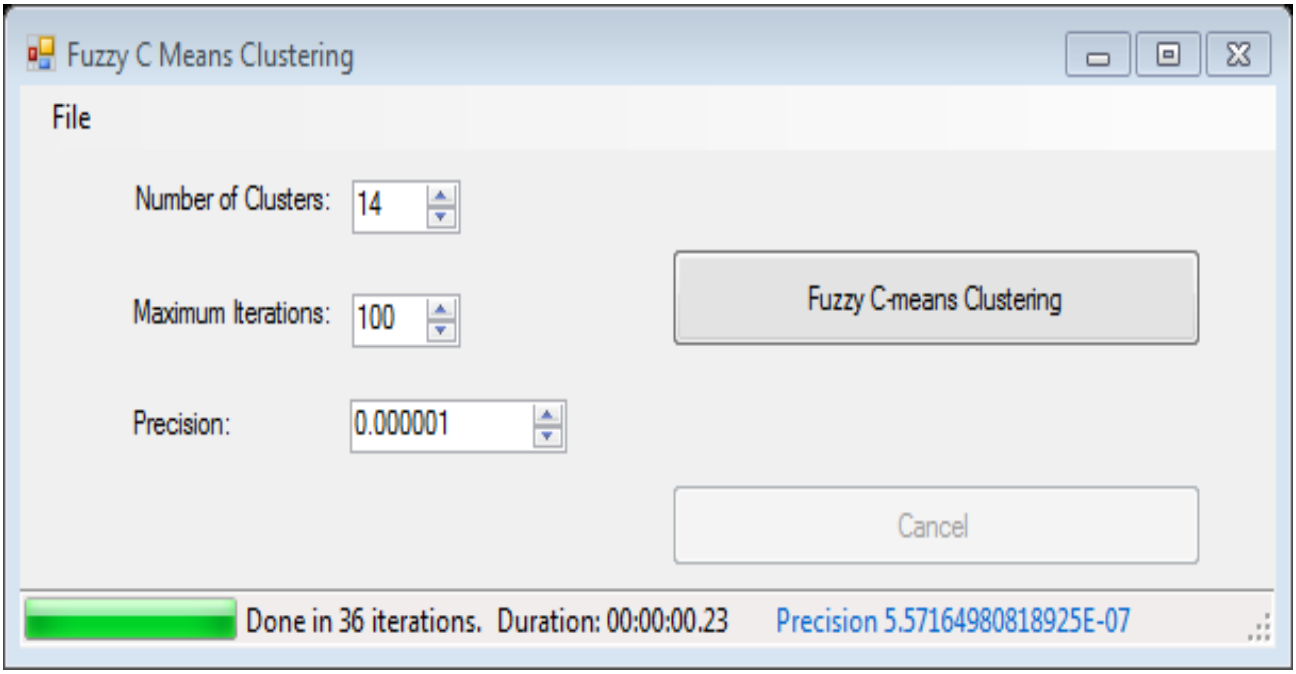


Figure 3.2: Fuzzy C-Means Clustering Graphical User interface

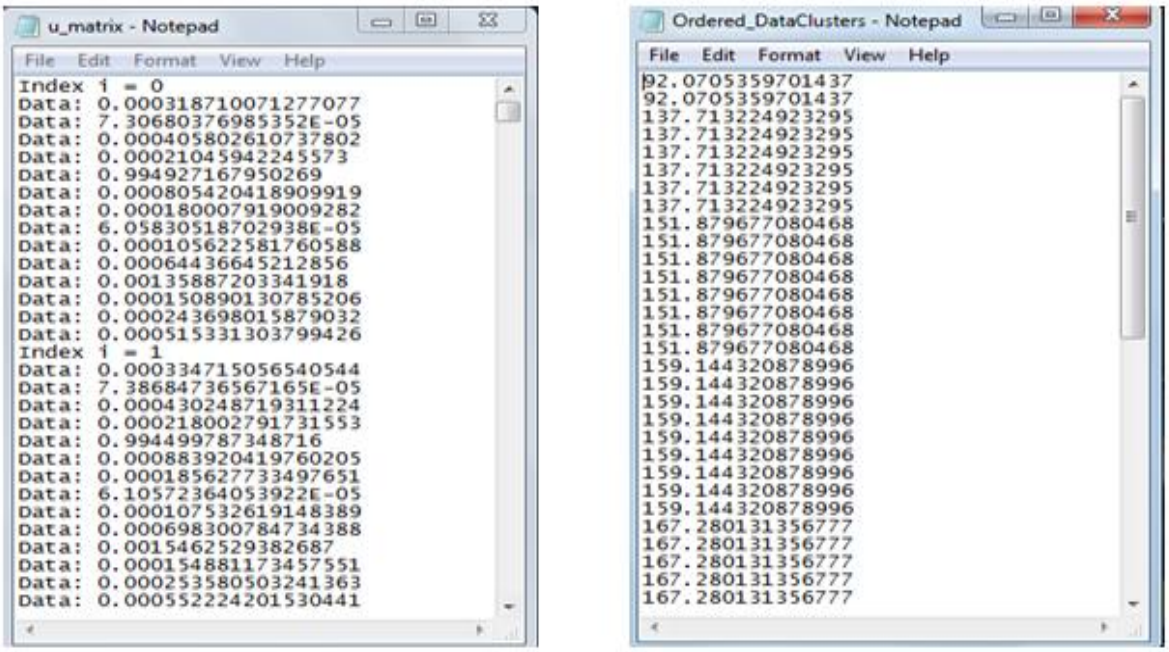


Figure 3.3: Screenshot Showing Partition Matrix and Cluster Centres

Table 3.2 shows the cluster centres obtained for number of clusters (partitions), c, ranging from 7 to 14.

Table 3.2: Cluster Centres for Training Erlang Data Set

		c = 7	c = 8	c = 9	c = 10	c = 11	c = 12	c = 13	c = 14
CLUSTERS	v_1	152.72	294.701	215.613	314.745	92.142	203.882	291.822	188.182
	v_2	258.827	140.789	168.18	205.817	308.594	302.431	92.084	294.705
	v_3	294.641	314.663	314.692	167.523	204.158	138.266	138.257	177.047
	v_4	211.843	92.173	154.859	223.058	291.813	291.864	260.209	210.748
	v_5	314.591	184.568	294.717	92.086	222.345	154.563	154.505	92.071
	v_6	93.556	259.383	187.508	154.612	159.102	219.273	223.22	151.88
	v_7	178.893	159.97	138.342	260.201	182.55	186.141	210.967	220.536
	v_8		214.275	92.097	138.271	260.131	92.084	317.554	314.779
	v_9			259.622	294.749	301.295	317.01	301.305	260.314
	v_{10}				186.281	140.099	232.367	201.907	159.144
	v_{11}					317.544	260.281	185.899	137.713
	v_{12}						167.397	167.261	232.547
	v_{13}							308.611	202.234
	v_{14}								167.28
PRECISION		8.85E-07	7.22E-07	8.07E-07	5.85E-07	7.75E-07	6.49E-07	6.85E-07	5.57E-07
ITERATIONS		97	74	57	99	89	60	36	36
DURATION		2	0.85	0.57	1.24	0.47	0.36	0.25	0.23

From Table 3.2, the cluster centres obtained are randomly arranged. This is because fuzzy c-means (FCM) clustering algorithm is an iterative search algorithm. It can also be seen from Table 3.2 that the stopping conditions ($J < 0.000001$ and $k = 100$) were met. For instance, to obtain fourteen (14) clusters, it took 36 iterations. This means that, after 36 iterations the precision $J = 5.57 \times 10^{-7}$ was reached. Thus, the program terminated because $J = 5.57 \times 10^{-7} < 0.000001$.

The mean square error (MSE_c) validity index for the range of clusters chosen is shown in Appendix B. The result in Appendix B shows that, for a one-dimensional data set, the mean square error (MSE_c) decreases as the number of clusters increases. The least mean squared error for the range of clusters investigated is $MSE_{14} = 7.526$. Thus, within the range chosen, the appropriate number of clusters required to partition the training data set is $c = 14$. It also shows that cluster validity is only an issue when describing the data structure for high dimensional data.

3.3.3 Ranking Clusters in Ascending Order

Fuzzy C-means algorithm is a heuristic search algorithm that randomly searches for optimal clusters and memberships that nearly describes degree of a data point belonging to all clusters. Clusters (partitions) produced are not orderly arranged like in the case of traditional fuzzy time series (FTS) partitions. In the literature, fuzzy sets are only defined on ordered partitions. Thus, there is the need to rank (order) clusters. In this work, clusters are arranged in ascending order to define fuzzy sets (linguistic variables), $A_r (= 1,2,3,....,c)$. Table 3.3 shows the ranked cluster centres and defined fuzzy sets for the training data set.

Table 3.3: Defined Fuzzy Sets for Training Data Set Partitioned into Fourteen (14) Clusters

Cluster Rank (r)	Cluster Centre (v_i)	Centre Value	Fuzzy Set A_r
1	v_5	92.071	A_1
2	v_{11}	137.713	A_2
3	v_6	151.88	A_3
4	v_{10}	159.144	A_4
5	v_{14}	167.28	A_5
6	v_3	177.047	A_6
7	v_1	188.182	A_7
8	v_{13}	202.234	A_8
9	v_4	210.748	A_9
10	v_7	220.536	A_{10}
11	v_{12}	232.547	A_{11}
12	v_9	260.314	A_{12}
13	v_2	294.705	A_{13}
14	v_8	314.779	A_{14}

These defined fuzzy sets and partition matrix are used to fuzzify the training data set. Appendix C shows the $\{0,1\}^{n\times 14}$ partition matrix for the Erlang (Training) set.

3.3.4Fuzzifying Time Series Data

The historical observations of Erlang traffic are converted to linguistic values using defined fuzzy sets and membership degree of each data point computed. A linguistic variable among $A_r \{r=1,2,3,....,c\}$ is the linguistic value of data point if in that fuzzy set, the data point has its maximum membership degree. For instance, the membership degrees for the voice traffic collected on 6th January, 2013 are shown in Appendix C (1st row of the table). Its membership degree is

maximum in the fifth cluster $\mu_5(\cdot)$; defined as A_1 (Table 3.3). Thus, the fuzzified result 90.319 is A_1. The fuzzy C–means (FCM) algorithm is coded in such a way that one of its outputs is a collection of data points that belong to a cluster centre, based on their maximum degree. Table 3.4 shows the fuzzified training data set.

Table 3.4: Fuzzified Daily Erlang C Training Set Traffic (1 December, 2012 – 29 January, 2013)

Date	Training Data Set	Fuzzy Set	Date	Training Data Set	Fuzzy Set
12/1/2012	170.232	A_5	12/31/2012	301.779	A_{13}
12/2/2012	160.343	A_4	1/1/2013	315.685	A_{14}
12/3/2012	150.965	A_3	1/2/2013	291.862	A_{13}
12/4/2012	163.151	A_4	1/3/2013	211.912	A_9
12/5/2012	179.86	A_6	1/4/2013	175.319	A_6
12/6/2012	176.526	A_6	1/5/2013	93.802	A_1
12/7/2012	155.55	A_4	1/6/2013	90.319	A_1
12/8/2012	154.571	A_3	1/7/2013	153.194	A_3
12/9/2012	160.874	A_4	1/8/2013	183.423	A_7
12/10/2012	148.72	A_3	1/9/2013	188.883	A_7
12/11/2012	138.522	A_2	1/10/2013	134.05	A_2
12/12/2012	159.171	A_4	1/11/2013	139.442	A_2
12/13/2012	169.367	A_5	1/12/2013	187.514	A_7
12/14/2012	156.205	A_4	1/13/2013	221.6	A_{10}
12/15/2012	152.556	A_3	1/14/2013	232.545	A_{11}
12/16/2012	176.442	A_6	1/15/2013	260.859	A_{12}
12/17/2012	209.617	A_9	1/16/2013	319.248	A_{14}
12/18/2012	164.002	A_5	1/17/2013	158.539	A_4
12/19/2012	168.123	A_5	1/18/2013	184.075	A_7
12/20/2012	189.797	A_7	1/19/2013	153.452	A_3
12/21/2012	202.416	A_8	1/20/2013	157.5	A_4
12/22/2012	131.758	A_2	1/21/2013	150.486	A_3
12/23/2012	166.487	A_5	1/22/2013	187.741	A_7
12/24/2012	202.214	A_8	1/23/2013	141.908	A_2
12/25/2012	219.48	A_{10}	1/24/2013	143.357	A_2
12/26/2012	259.722	A_{12}	1/25/2013	166.878	A_5
12/27/2012	291.306	A_{13}	1/26/2013	166.439	A_5
12/28/2012	300.841	A_{13}	1/27/2013	159.809	A_5
12/29/2012	308.587	A_{14}	1/28/2013	151.401	A_3
12/30/2012	292.317	A_{13}	1/29/2013	191.997	A_7

3.4DEFUZZIFICATION MODULE

The steps in the defuzzification module are implemented as follows:

3.4.1 Establishing Fuzzy Set Groups (FSGs)

In this work, fuzzy set groups (FSGs) are established against fuzzy logic relationship groups (FLRGs) in conventional fuzzy time series (FTS), to partition linguistic historical values into unique set of sub patterns which subsequently are converted into unique "if – then" statements. Table 4.5 shows the first pass of the fuzzy set groups (FSGs) algorithm. Every group appears in chronological order. For instance, label 1 is gotten by grouping the fuzzy sets for the traffic collected on 1st $\hat{A}_5$ and 2nd $\hat{A}_4$ December, 2012 in ascending order of time.

Table 3.5: Establishment of Fuzzy Set Groups

Date	LABEL	FSG	Date	LABEL	FSG
12/3/2012	1	$\{A_5, A_4\}$	1/2/2013	31	$\{A_{13}, A_{14}\}$
12/4/2012	2	$\{A_4, A_3\}$	1/3/2013	32	$\{A_{14}, A_{13}\}$
12/5/2012	3	$\{A_3, A_4\}$	1/4/2013	33	$\{A_{13}, A_9\}$
12/6/2012	4	$\{A_4, A_6\}$	1/5/2013	34	$\{A_9, A_6\}$
12/7/2012	5	$\{A_6, A_6\}$	1/6/2013	35	$\{A_6, A_1\}$
12/8/2012	6	$\{A_6, A_4\}$	1/7/2013	36	$\{A_1, A_1\}$
12/9/2012	7	$\{A_4, A_3\}$	1/8/2013	37	$\{A_1, A_3\}$
12/10/2012	8	$\{A_3, A_4\}$	1/9/2013	38	$\{A_3, A_7\}$
12/11/2012	9	$\{A_4, A_3\}$	1/10/2013	39	$\{A_7, A_7\}$
12/12/2012	10	$\{A_3, A_2\}$	1/11/2013	40	$\{A_7, A_2\}$
12/13/2012	11	$\{A_2, A_4\}$	1/12/2013	41	$\{A_2, A_2\}$
12/14/2012	12	$\{A_4, A_5\}$	1/13/2013	42	$\{A_2, A_7\}$
12/15/2012	13	$\{A_5, A_4\}$	1/14/2013	43	$\{A_7, A_{10}\}$
12/16/2012	14	$\{A_4, A_3\}$	1/15/2013	44	$\{A_{10}, A_{11}\}$
12/17/2012	15	$\{A_3, A_6\}$	1/16/2013	45	$\{A_{11}, A_{12}\}$
12/18/2012	16	$\{A_6, A_9\}$	1/17/2013	46	$\{A_{12}, A_{14}\}$
12/19/2012	17	$\{A_9, A_5\}$	1/18/2013	47	$\{A_{14}, A_4\}$
12/20/2012	18	$\{A_5, A_5\}$	1/19/2013	48	$\{A_4, A_7\}$
12/21/2012	19	$\{A_5, A_7\}$	1/20/2013	49	$\{A_7, A_3\}$
12/22/2012	20	$\{A_7, A_8\}$	1/21/2013	50	$\{A_3, A_4\}$
12/23/2012	21	$\{A_8, A_2\}$	1/22/2013	51	$\{A_4, A_3\}$
12/24/2012	22	$\{A_2, A_5\}$	1/23/2013	52	$\{A_3, A_7\}$
12/25/2012	23	$\{A_5, A_8\}$	1/24/2013	53	$\{A_7, A_2\}$
12/26/2012	24	$\{A_8, A_{10}\}$	1/25/2013	54	$\{A_2, A_2\}$
12/27/2012	25	$\{A_{10}, A_{12}\}$	1/26/2013	55	$\{A_2, A_5\}$
12/28/2012	26	$\{A_{12}, A_{13}\}$	1/27/2013	56	$\{A_5, A_5\}$
12/29/2012	27	$\{A_{13}, A_{13}\}$	1/28/2013	57	$\{A_5, A_5\}$
12/30/2012	28	$\{A_{13}, A_{14}\}$	1/29/2013	58	$\{A_5, A_3\}$
12/31/2012	29	$\{A_{14}, A_{13}\}$	1/30/2013	59	$\{A_3, A_7\}$
1/1/2013	30	$\{A_{13}, A_{13}\}$			

It can be seen from Table 3.5 that not all fuzzy set groups are unique. Fuzzy set groups labelled as 1 and 13; 2, 7, 9, 14 and 51; 3, 8 and 50; 18, 56 and 57; 22 and 55; 27 and 30; 28 and 31; 29, 32; 38,

52 and 59; 40 and 53; and 41 and 54 are identical. In order to obtain disambiguated fuzzy set groups, the ambiguous fuzzy set groups are extended to the next order and so on until unique groups are established. For instance, label 38, 52 and 59 is disambiguated by extending each label to the third order. This extension is achieved by adding the next past historical fuzzified set from A_3 into each group. These extensions result to A_1, A_3, A_7, A_4, A_3, A_7, and A_5, A_3, A_7 as new disambiguated fuzzy set groups (FSGs) for labels 38, 52 and 59 respectively. Appendix D shows the complete extension process for obtaining unique fuzzy set groups (FSGs). Table 3.6 shows the unique fuzzy set groups established for the training data set.

Table 3.6: Disambiguated Fuzzy Set Groups (FSGs)

LABEL	FSG	LABEL	FSG
1	{#, A_5, A_4}	31	{A_{14}, A_{13}, A_{13}, A_{14}}
2	{#, A_5, A_4, A_3}	32	{A_{14}, A_{13}, A_{13}, A_{14}, A_{13}}
3	{A_5, A_4, A_3, A_4}	33	{A_{13}, A_9}
4	{A_4, A_6}	34	{A_9, A_6}
5	{A_6, A_6}	35	{A_6, A_1}
6	{A_6, A_4}	36	{A_1, A_1}
7	{A_6, A_4, A_3}	37	{A_1, A_3}
8	{A_6, A_4, A_3, A_4}	38	{A_1, A_3, A_7}
9	{A_4, A_4, A_3}	39	{A_7, A_7}
10	{A_3, A_2}	40	{A_7, A_7, A_2}
11	{A_2, A_4}	41	{A_7, A_7, A_2, A_2}
12	{A_4, A_5}	42	{A_2, A_7}
13	{A_4, A_5, A_4}	43	{A_7, A_{10}}
14	{A_4, A_5, A_4, A_3}	44	{A_{10}, A_{11}}
15	{A_3, A_6}	45	{A_{11}, A_{12}}
16	{A_6, A_9}	46	{A_{12}, A_{14}}
17	{A_9, A_5}	47	{A_{14}, A_4}
18	{A_9, A_5, A_5}	48	{A_4, A_7}
19	{A_5, A_7}	49	{A_7, A_3}
20	{A_7, A_8}	50	{A_7, A_3, A_4}
21	{A_8, A_2}	51	{A_3, A_4, A_3}
22	{A_8, A_2, A_5}	52	{A_4, A_3, A_7}
23	{A_5, A_8}	53	{A_3, A_7, A_2}
24	{A_8, A_{10}}	54	{A_3, A_7, A_2, A_2}
25	{A_{10}, A_{12}}	55	{A_2, A_2, A_5}
26	{A_{12}, A_{13}}	56	{A_2, A_5, A_5}
27	{A_{12}, A_{13}, A_{13}}	57	{A_5, A_5, A_5}
28	{A_{12}, A_{13}, A_{13}, A_{14}}	58	{A_5, A_3}
29	{A_{12}, A_{13}, A_{13}, A_{14}, A_{13}}	59	{A_5, A_3, A_7}
30	{A_{14}, A_{13}, A_{13}}		

The # symbol, in a fuzzy set group (FSG), represents an unknown historical linguistic set.

Such fuzzy set group is incomplete.

3.4.2 Converting Fuzzy Set Groups into "if – then" Rules

These rules are generated on the basis of the content of the fuzzy set groups (FSGs). The "if – then" rule is of the form (Eleruja *et al*, 2012):

$$\textit{if } F(-1) = A_{r,t-1} \wedge F(-2) = A_{r,t-2} \wedge \ldots \wedge F(-n+1) = A_{r,t-n+1} \wedge F(-n) = A_{r,t-n}$$

$$\textit{then} \quad w_{t-1} = ? \wedge w_{t-2} = ? \wedge \ldots . \wedge w_{t-n+1} = ? \wedge w_{t-n} = ? \qquad (3.1)$$

Where;

w_{t-n} = the weight of the previous historical data point at time $(-n)$.

$t =$ time (period) the future data point whose forecast is required.

$n =$ time (period) of the previous historical data point matched in a forecasting rule.

This weight w_{t-n}, represents the strength of fuzzy logical relationship between the previous historical data at n and future forecast at t.

By processing all the content of the disambiguated fuzzy set groups (FSGs), a series of "if" – statements are generated as shown in Table 3.7.

Table 3.7: Partially Generated "if"- rules in Chronological Order

Rule	Matching Part
1	*if* $F(-1)=A_4 \wedge F(-2)=A_5 \wedge F(-3)=\#$
2	*if* $F(-1)=A_3 \wedge F(-2)=A_4 \wedge F(-3)=A_5 \wedge F(-4)=\#$
3	*if* $F(-1)=A_4 \wedge F(-2)=A_3 \wedge F(-3)=A_4 \wedge F(-4)=A_5$
4	*if* $F(-1)=A_6 \wedge F(-2)=A_4$
5	*if* $F(-1)=A_6 \wedge F(-2)=A_6$
6	*if* $F(-1)=A_4 \wedge F(-2)=A_6$
7	*if* $F(-1)=A_3 \wedge F(-2)=A_4 \wedge F(-3)=A_6$
8	*if* $F(-1)=A_4 \wedge F(-2)=A_3 \wedge F(-3)=A_4 \wedge F(-4)=A_6$
9	*if* $F(-1)=A_3 \wedge F(-2)=A_4 \wedge F(-3)=A_4$
10	*if* $F(-1)=A_2 \wedge F(-2)=A_3$
11	*if* $F(-1)=A_4 \wedge F(-2)=A_2$
12	*if* $F(-1)=A_5 \wedge F(-2)=A_4$
13	*if* $F(-1)=A_4 \wedge F(-2)=A_5 \wedge F(-3)=A_4$
14	*if* $F(-1)=A_3 \wedge F(-2)=A_4 \wedge F(-3)=A_5 \wedge F(-4)=A_4$
15	*if* $F(-1)=A_6 \wedge F(-2)=A_3$
16	*if* $F(-1)=A_9 \wedge F(-2)=A_6$
17	*if* $F(-1)=A_5 \wedge F(-2)=A_9$
18	*if* $F(-1)=A_5 \wedge F(-2)=A_5 \wedge F(-3)=A_9$
19	*if* $F(-1)=A_7 \wedge F(-2)=A_5$
20	*if* $F(-1)=A_8 \wedge F(-2)=A_7$
21	*if* $F(-1)=A_2 \wedge F(-2)=A_8$
22	*if* $F(-1)=A_5 \wedge F(-2)=A_2 \wedge F(-3)=A_8$
23	*if* $F(-1)=A_8 \wedge F(-2)=A_5$
24	*if* $F(-1)=A_{10} \wedge F(-2)=A_8$
25	*if* $F(-1)=A_{12} \wedge F(-2)=A_{10}$
26	*if* $F(-1)=A_{13} \wedge F(-2)=A_{12}$
27	*if* $F(-1)=A_{13} \wedge F(-2)=A_{13} \wedge F(-3)=A_{12}$

Rule	Matching Part
28	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{12}$
29	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{12}$
30	*if* $F(-1)=A_{13} \wedge F(-2)=A_{13} \wedge F(-3)=A_{14}$
31	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{14}$
32	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{14}$
33	*if* $F(-1)=A_{9} \wedge F(-2)=A_{13}$
34	*if* $F(-1)=A_{6} \wedge F(-2)=A_{9}$
35	*if* $F(-1)=A_{1} \wedge F(-2)=A_{6}$
36	*if* $F(-1)=A_{1} \wedge F(-2)=A_{1}$
37	*if* $F(-1)=A_{3} \wedge F(-2)=A_{1}$
38	*if* $F(-1)=A_{7} \wedge F(-2)=A_{3} \wedge F(-3)=A_{1}$
39	*if* $F(-1)=A_{7} \wedge F(-2)=A_{7}$
40	*if* $F(-1)=A_{2} \wedge F(-2)=A_{7} \wedge F(-3)=A_{7}$
41	*if* $F(-1)=A_{2} \wedge F(-2)=A_{2} \wedge F(-3)=A_{7} \wedge F(-4)=A_{7}$
42	*if* $F(-1)=A_{7} \wedge F(-2)=A_{2}$
43	*if* $F(-1)=A_{10} \wedge F(-2)=A_{7}$
44	*if* $F(-1)=A_{11} \wedge F(-2)=A_{10}$
45	*if* $F(-1)=A_{12} \wedge F(-2)=A_{11}$
46	*if* $F(-1)=A_{14} \wedge F(-2)=A_{12}$
47	*if* $F(-1)=A_{4} \wedge F(-2)=A_{14}$
48	*if* $F(-1)=A_{7} \wedge F(-2)=A_{4}$
49	*if* $F(-1)=A_{3} \wedge F(-2)=A_{7}$
50	*if* $F(-1)=A_{4} \wedge F(-2)=A_{3} \wedge F(-3)=A_{7}$
51	*if* $F(-1)=A_{3} \wedge F(-2)=A_{4} \wedge F(-3)=A_{3}$
52	*if* $F(-1)=A_{7} \wedge F(-2)=A_{3} \wedge F(-3)=A_{4}$
53	*if* $F(-1)=A_{2} \wedge F(-2)=A_{7} \wedge F(-3)=A_{3}$
54	*if* $F(-1)=A_{2} \wedge F(-2)=A_{2} \wedge F(-3)=A_{7} \wedge F(-4)=A_{3}$
55	*if* $F(-1)=A_{5} \wedge F(-2)=A_{2} \wedge F(-3)=A_{2}$
56	*if* $F(-1)=A_{5} \wedge F(-2)=A_{5} \wedge F(-3)=A_{2}$

Rule	Matching Part
57	$if \left(F(-1) = A_5 \wedge F(-2) = A_5 \wedge F(-3) = A_5\right)$
58	$if \left(F(-1) = A_3 \wedge F(-2) = A_5\right)$
59	$if \left(F(-1) = A_7 \wedge F(-2) = A_3 \wedge F(-3) = A_5\right)$

3.4.3 Tuning "if – then" Rules Using Particle Swarm Optimization (PSO)

The aim of integrating particle swarm optimization (PSO) algorithm into the defuzzification phase of fuzzy time series forecasting (FTS) model is to tune the "if – then" rules so that optimal weights, w_i, can be assigned to fuzzy sets in a relationship group (FSG). Each weight represents the strength of relationship between a past value and a future forecast. Tuning the rules will reduce mismatch between the forecasts and actual historical observations. The particle swarm optimization (PSO) algorithm is implemented in C# to reduce the computational cost of defuzzification. The flow chart for the particle swarm optimization algorithm is shown in Figure 3.4.

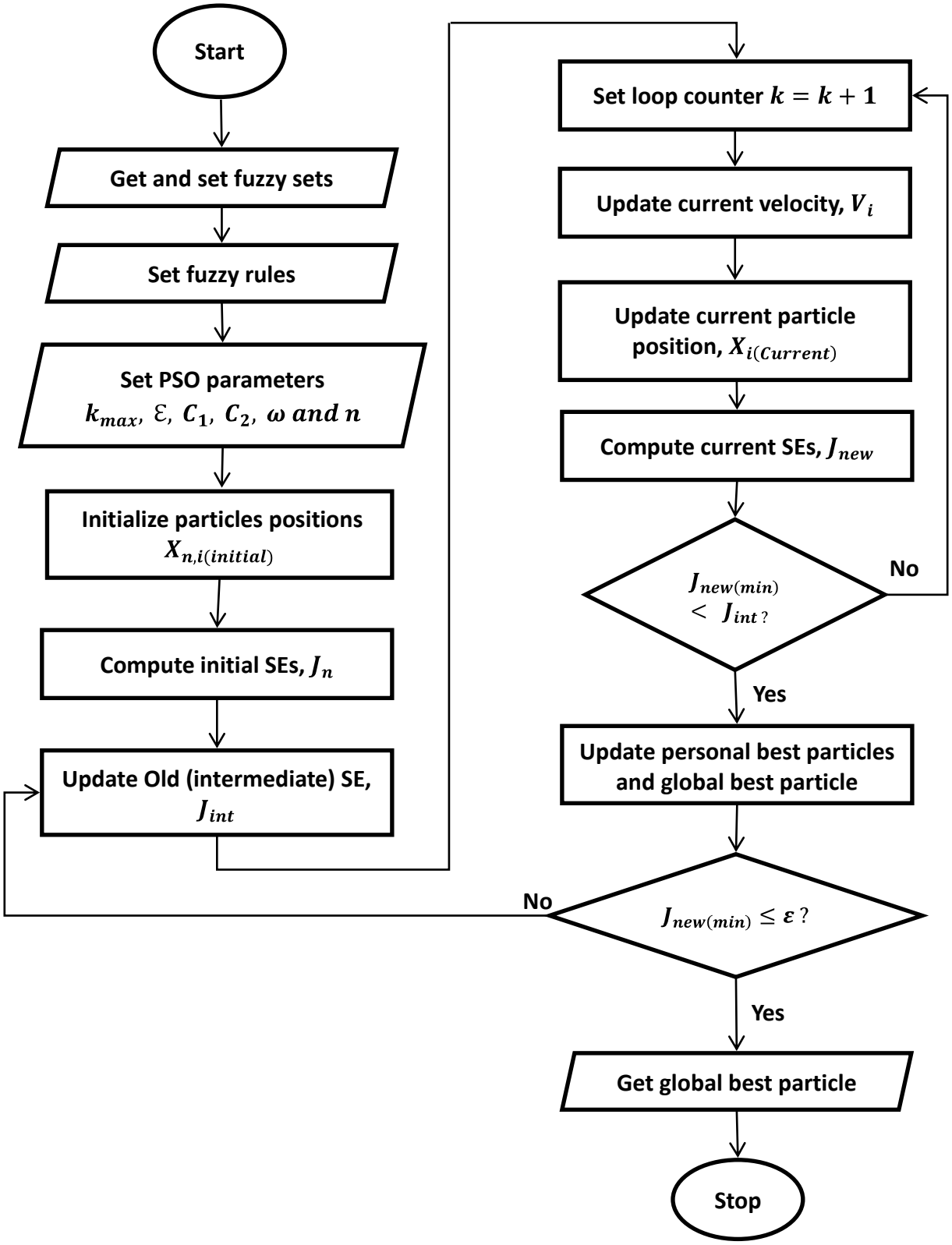


Figure 3.4: Flowchart of the Particle Swarm Optimization Algorithm.

To build the particle swarm optimization (PSO) code, the C# code editor of Microsoft Visual Studio Express 2012 for Windows desktop is the integrated development environment (IDE) utilized. Data form and fuzzy rule form are created to allow options such as; data sets and fuzzy rule sets, respectively, to be set and changed. A Graphical User Interface ("PSO Computer") is created to allow values of PSO parameters such as; inertia weight, ω,"self-confidence", C_1,"social-confidence", C_2,maximum number of iterations, $k_{\max}$,Target squared error, ε,number of particles, n,and particles initial positions, w_i, to be set and changed. Rule menu, calculate and cancel buttons are also available on the GUI to input a voice traffic rule, run the optimization code and stop the code from running, respectively. A status bar is also available on the GUI to display the global best particle results, number of iterations required to optimize the objective function, particles personal best positions and precision (squared error) of optimization. Figure 3.5 shows a screen shot of the GUI.

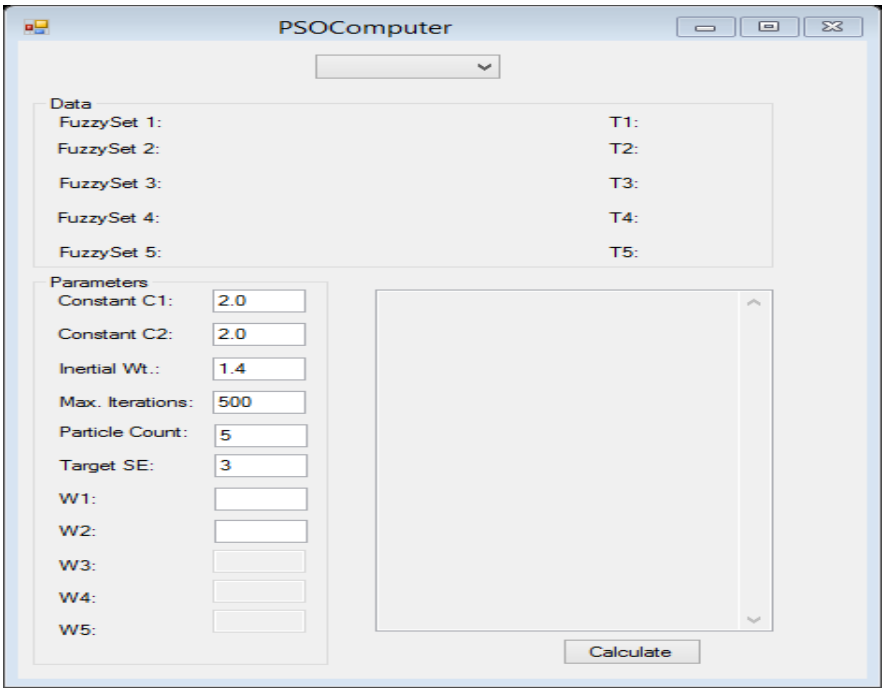


Figure 3.5: Screenshot of the PSO GUI.

The pseudo-code for the particle swarm optimization (PSO) algorithm is as follows:

i. Get and set voice traffic data and their fuzzified values.

Voice traffic data and their corresponding linguistic values are gotten from a "Data" form. The snippet below allows the Erlang C based voice traffic data and their corresponding fuzzified values to be set.

```
public float Data { get; set; }
public string FuzzySet { get; set; }
```

ii. Set fuzzy rules.

A "rule" form allows rules to be inputed. A fuzzy rule is a collection of historical fuzzy sets showing their combination in determining future forecast. The snippet of the "rule" class which allows fuzzy rules to be uploaded on a "rule form" is as follows:

```
public Rule(short _index, string[] _operands)
      {
         this.IndexNo = _index;
         this.Operands = _operands;
      }

      public override string ToString()
      {
         return "Index Number: " + this.IndexNo + ", " + "Operands: " + Operands[0] + ", " +
Operands[1] + ", " + Operands[2] + ", " + Operands[3] + ", " + Operands[4];
      }

      public short IndexNo { get; set; }
      public string[] Operands { get; set; }
   }
```

The snippet of the "PSOComputer" class that allows rule to be gotten from the rule form and set on the "PSO computer" is as follows:

```
private List<Rule> Rules { get; set; }
```

iii. **Set PSO parameters**

The snippet of the "PSOComputer" class allows inertia weight, ω, "self-confidence", C_1, "social-confidence", C_2, maximum number of iterations, $k_{\max}$, Target squared error, ε, number of particles, n, and particles initial positions, w_i, to be set and changed is as follows:

```
private void PrepareForPSO()
    {
        CurrentData.C1 = float.Parse(_txtC1.Text.Trim());//sets C1
        CurrentData.C2 = float.Parse(_txtC2.Text.Trim());//sets C2
        CurrentData.EnhancedComputations = (CurrentData.T3 >= 0); //if true, then con-
sider w3
        CurrentData.EnhancedComputations4 = (CurrentData.T4 >= 0); //if true, then con-
sider w4
        CurrentData.EnhancedComputations5 = (CurrentData.T5 >= 0); //if true, then con-
sider w5
        CurrentData.InertialWtCfft = float.Parse(_txtIWC.Text.Trim()); //sets ω
        CurrentData.MinimumSE = ulong.Parse(_txtMinSE.Text.Trim()); //sets ε
        CurrentData.NumberOfIterations = int.Parse(_txtMaxIterations.Text.Trim()); //sets
kmax
        CurrentData.ParticleCount = int.Parse(_txtParticleCount.Text.Trim());
//sets n

        CurrentData.W1 = float.Parse(_txtW1.Text.Trim());//sets wi = Xi

        CurrentData.W2 = float.Parse(_txtW2.Text.Trim());
        if (CurrentData.EnhancedComputations) //only bother if there is actually even a T3
        {
            CurrentData.W3 = float.Parse(_txtW3.Text.Trim());
        }
        if (CurrentData.EnhancedComputations4) //only bother if there is actually even a T4
        {
            CurrentData.W4 = float.Parse(_txtW4.Text.Trim());
        }
        if (CurrentData.EnhancedComputations5) //only bother if there is actually even a T5
        {
            CurrentData.W5 = float.Parse(_txtW5.Text.Trim());
        }
    }
```

iv.Initialize Particles Positions

The GUI allows particles initial positions to be entered into the program. The snippet of the "PSOEngine" class used to initialize initial positions of particles is given by:

```
Initialise(particles, __input, out initialVelocities);
```

In the snippet, "*_input*" refers to the particle positions, W_i, gotten from the GUI. These initial positions represents weights assigned to fuzzy sets in a rule. During initialization of particles positions, personal best positions and global best positions equals particles initial positions.

v.Compute initial squared errors, J_n, for each particle

The initial squared error for each particle is calculated to allow for an initial value of the objective function to be set. The snippet of the "PSOEngine" class that computes the initial squared error, for each particle, is shown as follows:

```
if (__input.EnhancedComputations)
        {
           initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3,
__input.W1, __input.W2, __input.W3);
        }
        else if (__input.EnhancedComputations4)
        {
           initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2,
__input.T3,__input.T4, __input.W1, __input.W2, __input.W3,  __input.W4);
        }
        else if (__input.EnhancedComputations5)
        {
           initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3,
__input.T4,__input.T5, __input.W1, __input.W2, __input.W3, __input.W4, __input.W5);
        }
         else
         {
           initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2,__input.W1,
__input.W2);
        }

        dumpVal = initialSE;
```

vi.Update intermediate square error

J_{int} is called the intermediate squared error. It is initialized by getting the maximum value among the old squared errors computed in step (iii). The snippet of the "PSOEngine" class performs the update.

```
ulong initialSE = long.MaxValue;

Initialise(oldSquaredErrors, initialSE);

for (int i = 0; i < intermediateSE.Length; i++)
        {
           intermediateSE[i] = initialSE;
        }
```

During any iteration, if the stopping condition, $J_{new(\min)} \leq \varepsilon$, is not met and $J_{new(\min)} \leq J_{\text{int}}$, J_{int} is updated. The snippet of the "PSOEngine" class code that performs this update is shown as follows:

```
intermediateSE[currPat] = currentSquaredErrors[currPat];

currentSquaredErrors[currPat] = GetSquaredErrors();

private static ulong GetSquaredErrors(float __data, float __T1, float __T2, float __T3, float
__T4, float __T5, float __W1, float __W2, float __W3, float __W4, float __W5)
      {
         checked //double insurance to ensure no overflows.
         {
            return (ulong)(((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3) + (__T4 * __W4)
+(__T5 * __W5) - __data) * ((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3) + (__T4 *
__W4) + (__T5 * __W5) - __data));
         }
      }
```

vii.Update current velocity, V_i

After the loop counter is set, velocities are updated according to equation (2.18). The snippet of the "PSOEngine" class that helps update the current velocities for each particle is as follows:

```
for (int particleNumber = 0; particleNumber < __input.ParticleCount; particleNumber++)
        {
      Particle currentParticle = particles[particleNumber]; currentParticle.VelocityOne =
ComputeVelocity(__input, currentParticle.LocalBest.W1, Particle.Globalbest.W1, currentParti-
cle.CurrentPosition1, currentParticle.VelocityOne);
                currentParticle.VelocityTwo = ComputeVelocity(__input,
currentParticle.LocalBest.W2, Particle.Globalbest.W2, currentParticle.CurrentPosition2,
currentParticle.VelocityTwo);

. . . .

        }
```

viii. Update current positions, X_i

The snippet of the "PSOEngine" class that updates the current positions for each particle is as follows:

```
currentParticle.CurrentPosition1 = (float)(currentParticle.CurrentPosition1 + currentParti-
cle.VelocityOne);
currentParticle.CurrentPosition2 = (float)(currentParticle.CurrentPosition2 +
currentParticle.VelocityTwo);
```

ix. Compute current square errors, J_{new}

J_{new} is a matrix of square errors calculated using the updated particles positions. The snippet of the "PSOEngine" class that computes J_{new} is as follows:

```
currentSquaredErrors[currPat] = GetSquaredErrors(__input.Data, __input.T1, __input.T2,
__input.T3, __input.T4, __input.T5, particles[currPat].CurrentPosition1,
particles[currPat].CurrentPosition2, particles[currPat].CurrentPosition3,
particles[currPat].CurrentPosition4, particles[currPat].CurrentPosition5);
```

x.Update personal best particles and global best particle

The PSO code updates particles' local best positions only when the minimum value of the current squared errors $J_{new\langle \min \rangle}$ is less than the intermediate squared error, J_{int}. The snippet of the

"PSOEngine" class that performs this update is as follows:

```
if (currentSquaredErrors[currPat] < oldSquaredErrors[currPat])
 {
   doGlobalUpdate = true;
   oldSquaredErrors[currPat] = currentSquaredErrors[currPat]; //update the SE
   particles[currPat].LocalBest = new BestValue { W1 = parti
                cles[currPat].CurrentPosition1, W2 = particles[currPat].CurrentPosition2,
                W3 = particles[currPat].CurrentPosition3, W4 = parti
                cles[currPat].CurrentPosition4, W5 = particles[currPat].CurrentPosition5};

  }
```

A global best particle is the particle among the local best particles whose squared error value is J_{int}. The snippet of the "PSOEngine" class that performs global best particle (positions) update is as follows:

```
if (doGlobalUpdate)
  {
   oldSquaredErrors[__input.ParticleCount] = currentSquaredErrors.Min();

   for (int par = 0; par < __input.ParticleCount; par++)
      {
       if (currentSquaredErrors[par] == oldSquaredErrors[__input.ParticleCount])
         {
          Particle.Globalbest = new BestValue { W1 = particles[par].
          CurrentPosition1, W2 = particles[par].CurrentPosition2, W3 = parti
          cles[par].CurrentPosition3, W4 = particles[par].CurrentPosition4,
          W5= particles[par].CurrentPosition5
         };

        luckyParticle = par;
        break;
       }//this must be the particle who is the global best for this run.
   }
```

xi.Get global best particle positions

If the minimum of the current square errors, $J_{new(\min)}$ is less than or equal to the stopping criteria, $k_{\max} = 500, \varepsilon = 3$, the algorithm terminates. The global best positions, after termination, are gottenby

the structure, “BestValue”, shown as follows:

```
public BestValue Result { get; set; } //I shall leave this here just in case I'll need it someday.


public struct BestValue
  {
    public float W1 { get; set; }
    public float W2 { get; set; }
    public float W3 { get; set; }
    public float W4 { get; set; }
    public float W5 { get; set; }
  }
```

The complete PSO program is shown in Appendix J. Figure 3.6 shows the screen shot of the PSO GUI after a rule has been uploaded and the PSO program is run.

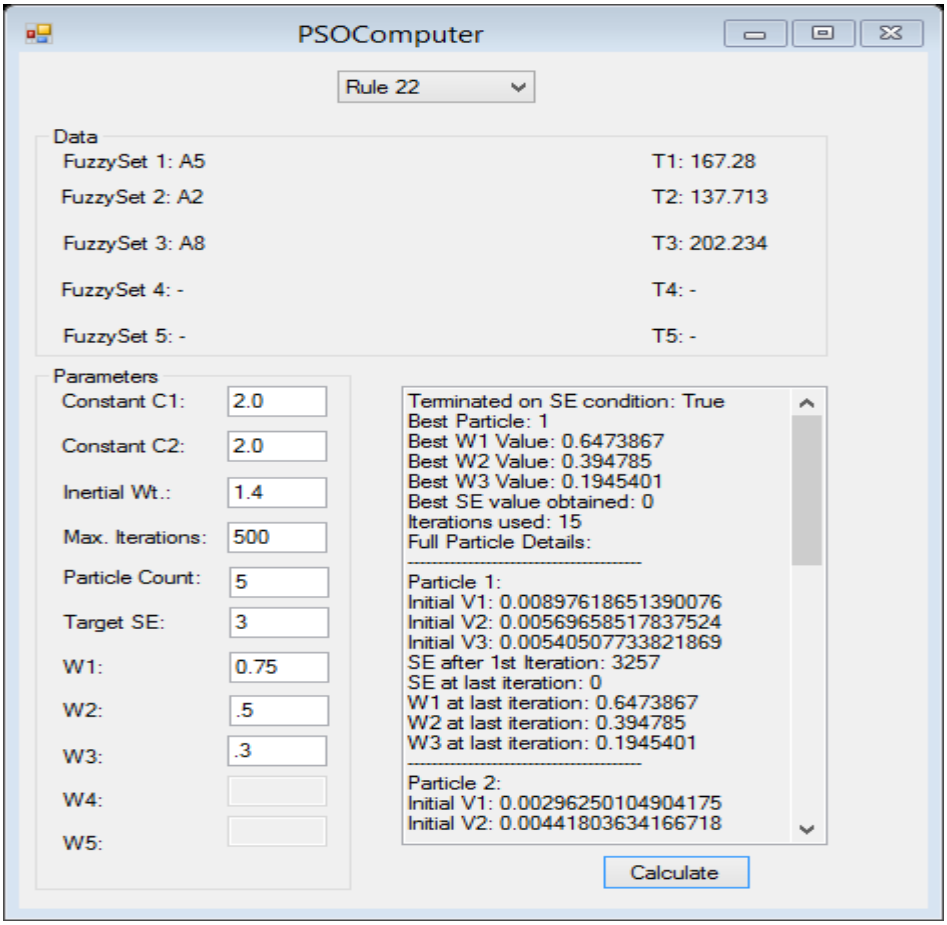


Figure 3.6: Graphical User Interface Showing Results of Tuning Weights of a Rule

The GUI shows the inputted parameters, particles initial positions,$(W_1\ to\ W_5)$, and results optimal weights (global best positions) obtained when a rule is optimized. As seen from the GUI, weights $W_1 = 0.75, W_2 = 0.5\ and\ W_3 = 0.3\,0.75$, 0.5 and 0.3 are initially assigned to fuzzy sets A_5, A_2, and A_8, respectively. When the program is run, the optimal weights for "rule 22" are: 0.6474, 0.3948 and 0.1945 for fuzzy sets A_5, A_2, and A_8, respectively. It is only when weights have been tuned that the complete "if – then" rule can be obtained. Table 3.8 shows the complete "if-then" rules for the training data set.

Table 3.8: Completely Generated "if-then" Rules for the Training Data Set

Rule	Matching Part	Weights
1	*if* $F(-1) = A_4 \wedge F(-2) = A_5 \wedge F(-3) = \#$	*then* $w_1 = ?, w_2 = ? \ \& \ w_3 = ?$
2	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_5 \wedge F(-4) = \#$	*then* $w_1 = ?, w_2 = ?, \ w_3 = ? \ \& \ w_4 = ?$
3	*if* $F(-1) = A_4 \wedge F(-2) = A_3 \wedge F(-3) = A_4 \wedge F(-4) = A_5$	*then* $w_1 = 0.6137, w_2 = 0.3637, \ w_3 = 0.167$ & $w_4 = 0.0063$
4	*if* $F(-1) = A_6 \wedge F(-2) = A_4$	*then* $w_1 = 0.6482 \ \& \ w_2 = 0.3934$
5	*if* $F(-1) = A_6 \wedge F(-2) = A_6$	*then* $w_1 = 0.5659 \ \& \ w_2 = 0.3165$
6	*if* $F(-1) = A_4 \wedge F(-2) = A_6$	*then* $w_1 = 0.5988 \ \& \ w_2 = 0.3449$
7	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_6$	*then* $w_1 = 0.5763, w_2 = 0.3274 \ \& \ w_3 = 0.123$
8	*if* $F(-1) = A_4 \wedge F(-2) = A_3 \wedge F(-3) = A_4 \wedge F(-4) = A_6$	*then* $w_1 = 0.5474, w_2 = 0.2974, \ w_3 = 0.098$ & $w_4 = 0.001$
9	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_4$	*then* $w_1 = 0.5374, w_2 = 0.2834, \ w_3 = 0.0858$
10	*if* $F(-1) = A_2 \wedge F(-2) = A_3$	*then* $w_1 = 0.6802 \ \& \ w_2 = 0.4361$
11	*if* $F(-1) = A_4 \wedge F(-2) = A_2$	*then* $w_1 = 0.6967 \ \& \ w_2 = 0.4379$
12	*if* $F(-1) = A_5 \wedge F(-2) = A_4$	*then* $w_1 = 0.6039 \ \& \ w_2 = 0.3549$
13	*if* $F(-1) = A_4 \wedge F(-2) = A_5 \wedge F(-3) = A_4$	*then* $w_1 = 0.5443, w_2 = 0.2956 \ \& \ w_3 = 0.0974$
14	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_5 \wedge F(-4) = A_4$	*then* $w_1 = 0.6041, w_2 = 0.3566, \ w_3 = 0.1678$ & $w_4 = 0.0011$
15	*if* $F(-1) = A_6 \wedge F(-2) = A_3$	*then* $w_1 = 0.7511 \ \& \ w_2 = 0.5073$
16	*if* $F(-1) = A_9 \wedge F(-2) = A_6$	*then* $w_1 = 0.5385 \ \& \ w_2 = 0.2886$
17	*if* $F(-1) = A_5 \wedge F(-2) = A_9$	*then* $w_1 = 0.5791 \ \& \ w_2 = 0.3354$
18	*if* $F(-1) = A_5 \wedge F(-2) = A_5 \wedge F(-3) = A_9$	*then* $w_1 = 0.5959, w_2 = 0.3464 \ \& \ w_3 = 0.1474$
19	*if* $F(-1) = A_7 \wedge F(-2) = A_5$	*then* $w_1 = 0.6887 \ \& \ w_2 = 0.4398$
20	*if* $F(-1) = A_8 \wedge F(-2) = A_7$	*then* $w_1 = 0.4603 \ \& \ w_2 = 0.2108$
21	*if* $F(-1) = A_2 \wedge F(-2) = A_8$	*then* $w_1 = 0.6366 \ \& \ w_2 = 0.3872$
22	*if* $F(-1) = A_5 \wedge F(-2) = A_2 \wedge F(-3) = A_8$	*then* $w_1 = 0.6474, w_2 = 0.3948 \ \& \ w_3 = 0.1945$
23	*if* $F(-1) = A_8 \wedge F(-2) = A_5$	*then* $w_1 = 0.7082 \ \& \ w_2 = 0.4565$
Rule	**Matching Part**	**Weights**

24	*if* $F(-1)=A_{10} \wedge F(-2)=A_8$	*then* $w_1 = 0.7416$ & $w_2 = 0.4788$
25	*if* $F(-1)=A_{12} \wedge F(-2)=A_{10}$	*then* $w_1 = 0.7169$ & $w_2 = 0.4733$
26	*if* $F(-1)=A_{13} \wedge F(-2)=A_{12}$	*then* $w_1 = 0.657$ & $w_2 = 0.41$
27	*if* $F(-1)=A_{13} \wedge F(-2)=A_{13} \wedge F(-3)=A_{12}$	*then* $w_1 = 0.5874$, $w_2 = 0.3374$ & $w_3 = 0.1381$
28	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{12}$	*then* $w_1 = 0.5444$, $w_2 = 0.3117$, $w_3 = 0.0974$ & $w_4 = 0.0032$
29	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{12}$	*then* $w_1 = 0.5674$, $w_2 = 0.3174$, $w_3 = 0.1181$, $w_4 = 0.0032$ & $w_5 = 0.0007$
30	*if* $F(-1)=A_{13} \wedge F(-2)=A_{13} \wedge F(-3)=A_{14}$	*then* $w_1 = 0.5874$, $w_2 = 0.3374$ & $w_3 = 0.1381$
31	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{14}$	*then* $w_1 = 0.5491$, $w_2 = 0.3008$, $w_3 = 0.1005$ & $w_4 = 0.0027$
32	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{14}$	*then* $w_1 = 0.4652$, $w_2 = 0.2174$, $w_3 = 0.0174$, $w_4 = 0.0052$ & $w_5 = 0.0001$
33	*if* $F(-1)=A_9 \wedge F(-2)=A_{13}$	*then* $w_1 = 0.4923$ & $w_2 = 0.2445$
34	*if* $F(-1)=A_6 \wedge F(-2)=A_9$	*then* $w_1 = 0.3774$ & $w_2 = 0.1294$
35	*if* $F(-1)=A_1 \wedge F(-2)=A_6$	*then* $w_1 = 0.5007$ & $w_2 = 0.2492$
36	*if* $F(-1)=A_1 \wedge F(-2)=A_1$	*then* $w_1 = 0.9500$ & $w_2 = 0.7000$
37	*if* $F(-1)=A_3 \wedge F(-2)=A_1$	*then* $w_1 = 0.8400$ & $w_2 = 0.5900$
38	*if* $F(-1)=A_7 \wedge F(-2)=A_3 \wedge F(-3)=A_1$	*then* $w_1 = 0.6250$, $w_2 = 0.3739$ & $w_3 = 1749$
39	*if* $F(-1)=A_7 \wedge F(-2)=A_7$	*then* $w_1 = 0.4859$ & $w_2 = 0.2287$
40	*if* $F(-1)=A_2 \wedge F(-2)=A_7 \wedge F(-3)=A_7$	*then* $w_1 = 0.5274$, $w_2 = 0.2737$ & $w_3 = 0.0783$
41	*if* $F(-1)=A_2 \wedge F(-2)=A_2 \wedge F(-3)=A_7 \wedge F(-4)=A_7$	*then* $w_1 = 0.6631$, $w_2 = 0.4149$, $w_3 = 0.2136$ & $w_4 = 0.0059$
42	*if* $F(-1)=A_7 \wedge F(-2)=A_2$	*then* $w_1 = 0.7800$ & $w_2 = 0.5300$
43	*if* $F(-1)=A_{10} \wedge F(-2)=A_7$	*then* $w_1 = 0.6884$ & $w_2 = 0.4378$
44	*if* $F(-1)=A_{11} \wedge F(-2)=A_{10}$	*then* $w_1 = 0.6965$ & $w_2 = 0.4467$
45	*if* $F(-1)=A_{12} \wedge F(-2)=A_{11}$	*then* $w_1 = 0.7698$ & $w_2 = 0.5034$
46	*if* $F(-1)=A_{14} \wedge F(-2)=A_{12}$	*then* $w_1 = 0.3859$ & $w_2 = 0.1362$
Rule	**Matching Part**	**Weights**

47	*if* $F(t-1) = A_4 \wedge F(t-2) = A_{14}$	*then* $w_1 = 0.5568$ & $w_2 = 0.3071$
48	*if* $F(t-1) = A_7 \wedge F(t-2) = A_4$	*then* $w_1 = 0.5570$ & $w_2 = 0.3159$
49	*if* $F(t-1) = A_3 \wedge F(t-2) = A_7$	*then* $w_1 = 0.5993$ & $w_2 = 0.3548$
50	*if* $F(t-1) = A_4 \wedge F(t-2) = A_3 \wedge F(t-3) = A_7$	*then* $w_1 = 0.5519$, $w_2 = 0.2974$ & $w_3 = 0.0963$
51	*if* $F(t-1) = A_3 \wedge F(t-2) = A_4 \wedge F(t-3) = A_3$	*then* $w_1 = 0.5362$, $w_2 = 0.4077$ & $w_3 = 0.2800$
52	*if* $F(t-1) = A_7 \wedge F(t-2) = A_3 \wedge F(t-3) = A_4$	*then* $w_1 = 0.5047$, $w_2 = 0.2564$ & $w_3 = 0.0552$
53	*if* $F(t-1) = A_2 \wedge F(t-2) = A_7 \wedge F(t-3) = A_3$	*then* $w_1 = 0.5374$, $w_2 = 0.2842$ & $w_3 = 0.1134$
54	*if* $F(t-1) = A_2 \wedge F(t-2) = A_2 \wedge F(t-3) = A_7 \wedge F(t-4) = A_3$	*then* $w_1 = 0.6131$, $w_2 = 0.3663$, $w_3 = 0.1674$ & $w_4 = 0.0059$
55	*if* $F(t-1) = A_5 \wedge F(t-2) = A_2 \wedge F(t-3) = A_2$	*then* $w_1 = 0.5968$, $w_2 = 0.3437$ & $w_3 = 0.1440$
56	*if* $F(t-1) = A_5 \wedge F(t-2) = A_5 \wedge F(t-3) = A_2$	*then* $w_1 = 0.5574$, $w_2 = 0.3074$ & $w_3 = 0.1038$
57	*if* $F(t-1) = A_5 \wedge F(t-2) = A_5 \wedge F(t-3) = A_5$	*then* $w_1 = 0.5374$, $w_2 = 0.2898$ & $w_3 = 0.0832$
58	*if* $F(t-1) = A_3 \wedge F(t-2) = A_5$	*then* $w_1 = 0.7364$ & $w_2 = 0.4805$
59	*if* $F(t-1) = A_7 \wedge F(t-2) = A_3 \wedge F(t-3) = A_5$	*then* $w_1 = ?$, $w_2 = ?$ & $w_3 = ?$

3.4.4 Deriving Forecasts

Equation (2.20) is utilized to compute the forecasts of voice traffic. It is important to note that fuzzy values a_{t-i}, are cluster centres of historical observations. To illustrate this, if forecast for $t = 9^{th}$ *December*, 2012, is to be found, then at least two most recent historical fuzzy sets should make up the rule; as shown in Label 7 of Table 3.5. The label number is the same as the matching rule number. Thus, from Table 3.8, the matching "if-then" rule for forecastingErlang traffic for $t = 9^{th}$ *December*, 2012, is Rule 7. Also, from Table 3.8, $a_{t-1} = A_3, a_{t-2} = A_4$ *and* $a_{t-3} = A_6$. Hence, the forecasted voice traffic is computed as:

$$Y(9^{th} \text{ December}, 2012) = (51.88 \times 0.5763) + (59.144 \times 0.3274) + (77.047 \times 0.123) = 161.409$$

3.5INVESTIGATING THE EFFECT OF REVERSEDWEIGHTS

In fuzzy time series (FTS) forecasting, it is assumed that a stronger relationship exist between a forecasted data and a more recent observation. It implies that a larger weight is assigned to a more recent historical data. Thus, in assigning optimal weights, $w_1 > w_2 > w_3 > ... > w_n$. Reversing the weights will be such that $w_1 < w_2 < w_3 < ... < w_n$. To investigate the effect of reversing the weights on the forecasting process, first, in any rule, the assigned weights should be reversed. Subsequently, the same optimization procedure is implemented. The result of the optimization process is shown in Table 3.9.

Table 3.9: Generated "if-then" Rules for Training Data Set (Reversed Weights)

Rule	Matching Part	Weights
1	*if* $F(-1) = A_4 \wedge F(-2) = A_5 \wedge F(-3) = \#$	*then* $w_1 = ?, w_2 = ? \ \& \ w_3 = ?$
2	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_5 \wedge F(-4) = \#$	*then* $w_1 = ?, w_2 = ?, \ w_3 = ? \ \& \ w_4 = ?$
3	*if* $F(-1) = A_4 \wedge F(-2) = A_3 \wedge F(-3) = A_4 \wedge F(-4) = A_5$	*then* $w_1 = 0.0020, w_2 = 0.150, \ w_3 = 0.3790$ & $w_4 = 0.5800$
4	*if* $F(-1) = A_6 \wedge F(-2) = A_4$	*then* $w_1 = 0.4035 \ \& \ w_2 = 0.6607$
5	*if* $F(-1) = A_6 \wedge F(-2) = A_6$	*then* $w_1 = 0.3169 \ \& \ w_2 = 0.5688$
6	*if* $F(-1) = A_4 \wedge F(-2) = A_6$	*then* $w_1 = 0.3316 \ \& \ w_2 = 0.5819$
7	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_6$	*then* $w_1 = 0.1007, w_2 = 0.3007 \ \& \ w_3 = 0.5485$
8	*if* $F(-1) = A_4 \wedge F(-2) = A_3 \wedge F(-3) = A_4 \wedge F(-4) = A_6$	*then* $w_1 = 0.0010, w_2 = 0.1200, \ w_3 = 0.3180$ & $w_4 = 0.5055$
9	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_4$	*then* $w_1 = 0.0786, w_2 = 0.2807, \ w_3 = 0.5329$
10	*if* $F(-1) = A_2 \wedge F(-2) = A_3$	*then* $w_1 = 0.4149 \ \& \ w_2 = 0.6700$
11	*if* $F(-1) = A_4 \wedge F(-2) = A_2$	*then* $w_1 = 0.4588 \ \& \ w_2 = 0.7061$
12	*if* $F(-1) = A_5 \wedge F(-2) = A_4$	*then* $w_1 = 0.3587 \ \& \ w_2 = 0.6083$
13	*if* $F(-1) = A_4 \wedge F(-2) = A_5 \wedge F(-3) = A_4$	*then* $w_1 = 0.0983, w_2 = 0.2986 \ \& \ w_3 = 0.5485$
14	*if* $F(-1) = A_3 \wedge F(-2) = A_4 \wedge F(-3) = A_5 \wedge F(-4) = A_4$	*then* $w_1 = 0.0084, w_2 = 0.1407, \ w_3 = 0.3532$ & $w_4 = 0.5907$
15	*if* $F(-1) = A_6 \wedge F(-2) = A_3$	*then* $w_1 = 0.5193 \ \& \ w_2 = 0.7700$
16	*if* $F(-1) = A_9 \wedge F(-2) = A_6$	*then* $w_1 = 0.3076 \ \& \ w_2 = 0.5594$
17	*if* $F(-1) = A_5 \wedge F(-2) = A_9$	*then* $w_1 = 0.3058 \ \& \ w_2 = 0.5553$
18	*if* $F(-1) = A_5 \wedge F(-2) = A_5 \wedge F(-3) = A_9$	*then* $w_1 = 0.1085, w_2 = 0.3195 \ \& \ w_3 = 0.5607$
19	*if* $F(-1) = A_7 \wedge F(-2) = A_5$	*then* $w_1 = 0.4700 \ \& \ w_2 = 0.6700$
20	*if* $F(-1) = A_8 \wedge F(-2) = A_7$	*then* $w_1 = 0.2191 \ \& \ w_2 = 0.4691$
21	*if* $F(-1) = A_2 \wedge F(-2) = A_8$	*then* $w_1 = 0.6366 \ \& \ w_2 = 0.3872$
22	*if* $F(-1) = A_5 \wedge F(-2) = A_2 \wedge F(-3) = A_8$	*then* $w_1 = 0.1682, w_2 = 0.3683 \ \& \ w_3 = 0.6182$
23	*if* $F(-1) = A_8 \wedge F(-2) = A_5$	*then* $w_1 = 0.4879 \ \& \ w_2 = 0.7328$
24	*if* $F(-1) = A_{10} \wedge F(-2) = A_8$	*then* $w_1 = 0.4942 \ \& \ w_2 = 0.7449$
25	*if* $F(-1) = A_{12} \wedge F(-2) = A_{10}$	*then* $w_1 = 0.4952 \ \& \ w_2 = 0.7446$
26	*if* $F(-1) = A_{13} \wedge F(-2) = A_{12}$	*then* $w_1 = 0.4254 \ \& \ w_2 = 0.6745$
Rule	**Matching Part**	**Weights**
27	*if* $F(-1) = A_{13} \wedge F(-2) = A_{13} \wedge F(-3) = A_{12}$	*then* $w_1 = 0.1706, w_2 = 0.3486 \ \& \ w_3 = 0.6007$

28	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{12}$	*then* $w_1 = 0.0190$, $w_2 = 0.1107$, $w_3 = 0.3107$ & $w_4 = 0.6258$
29	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{12}$	*then* $w_1 = 0.0003$, $w_2 = 0.0132$, $w_3 = 0.1822$, $w_4 = 0.2832$ & $w_5 = 0.6243$
30	*if* $F(-1)=A_{13} \wedge F(-2)=A_{13} \wedge F(-3)=A_{14}$	*then* $w_1 = 0.1196$, $w_2 = 0.3216$ & $w_3 = 0.5897$
31	*if* $F(-1)=A_{14} \wedge F(-2)=A_{13} \wedge F(-3)=A_{13} \wedge F(-4)=A_{14}$	*then* $w_1 = 0.0024$, $w_2 = 0.1008$, $w_3 = 0.3014$ & $w_4 = 0.5485$
32	*if* $F(-1)=A_{13} \wedge F(-2)=A_{14} \wedge F(-3)=A_{13} \wedge F(-4)=A_{13} \wedge F(-5)=A_{14}$	*then* $w_1 = 0.0003$, $w_2 = 0.0158$, $w_3 = 0.0158$, $w_4 = 0.2351$ & $w_5 = 0.4287$
33	*if* $F(-1)=A_9 \wedge F(-2)=A_{13}$	*then* $w_1 = 0.2028$ & $w_2 = 0.4533$
34	*if* $F(-1)=A_6 \wedge F(-2)=A_9$	*then* $w_1 = 0.1023$ & $w_2 = 0.3530$
35	*if* $F(-1)=A_1 \wedge F(-2)=A_6$	*then* $w_1 = 0.1783$ & $w_2 = 0.4228$
36	*if* $F(-1)=A_1 \wedge F(-2)=A_1$	*then* $w_1 = 0.7$ & $w_2 = 0.9478$
37	*if* $F(-1)=A_3 \wedge F(-2)=A_1$	*then* $w_1 = 0.6504$ & $w_2 = 0.9100$
38	*if* $F(-1)=A_7 \wedge F(-2)=A_3 \wedge F(-3)=A_1$	*then* $w_1 = 0.2692$, $w_2 = 0.4707$ & $w_3 = 0.7191$
39	*if* $F(-1)=A_7 \wedge F(-2)=A_7$	*then* $w_1 = 0.2283$ & $w_2 = 0.4789$
40	*if* $F(-1)=A_2 \wedge F(-2)=A_7 \wedge F(-3)=A_7$	*then* $w_1 = 0.0453$, $w_2 = 0.2296$ & $w_3 = 0.4787$
41	*if* $F(-1)=A_2 \wedge F(-2)=A_2 \wedge F(-3)=A_7 \wedge F(-4)=A_7$	*then* $w_1 = 0.0069$, $w_2 = 0.1600$, $w_3 = 0.3351$
42	*if* $F(-1)=A_7 \wedge F(-2)=A_2$	*then* $w_1 = 0.5700$ & $w_2 = 0.8200$
43	*if* $F(-1)=A_{10} \wedge F(-2)=A_7$	*then* $w_1 = 0.4530$ & $w_2 = 0.7033$
44	*if* $F(-1)=A_{11} \wedge F(-2)=A_{10}$	*then* $w_1 = 0.4507$ & $w_2 = 0.7000$
45	*if* $F(-1)=A_{12} \wedge F(-2)=A_{11}$	*then* $w_1 = 0.5300$ & $w_2 = 0.7800$
46	*if* $F(-1)=A_{14} \wedge F(-2)=A_{12}$	*then* $w_1 = 0.1668$ & $w_2 = 0.4055$
47	*if* $F(-1)=A_4 \wedge F(-2)=A_{14}$	*then* $w_1 = 0.2291$ & $w_2 = 0.4717$
48	*if* $F(-1)=A_7 \wedge F(-2)=A_4$	*then* $w_1 = 0.3257$ & $w_2 = 0.5787$
49	*if* $F(-1)=A_3 \wedge F(-2)=A_7$	*then* $w_1 = 0.3234$ & $w_2 = 0.5750$
50	*if* $F(-1)=A_4 \wedge F(-2)=A_3 \wedge F(-3)=A_7$	*then* $w_1 = 0.0689$, $w_2 = 0.2704$ & $w_3 = 0.5207$
51	*if* $F(-1)=A_3 \wedge F(-2)=A_4 \wedge F(-3)=A_3$	*then* $w_1 = 0.0837$, $w_2 = 0.2846$ & $w_3 = 0.5355$
52	*if* $F(-1)=A_7 \wedge F(-2)=A_3 \wedge F(-3)=A_4$	*then* $w_1 = 0.5047$, $w_2 = 0.2564$ & $w_3 = 0.0552$
53	*if* $F(-1)=A_2 \wedge F(-2)=A_7 \wedge F(-3)=A_3$	*then* $w_1 = 0.0807$, $w_2 = 0.2783$ & $w_3 = 0.5307$
Rule	**Matching Part**	**Weights**
54	*if* $F(-1)=A_2 \wedge F(-2)=A_2 \wedge F(-3)=A_7 \wedge F(-4)=A_3$	*then* $w_1 = 0.0460$, $w_2 = 0.1311$, $w_3 = 0.3211$

55	$if\ F(-1) = A_5 \wedge F(-2) = A_2 \wedge F(-3) = A_2$	$then\ \ w_1 = 0.1707, w_2 = 0.3707\ \&\ \ w_3 = 0.6279$
56	$if\ F(-1) = A_5 \wedge F(-2) = A_5 \wedge F(-3) = A_2$	$then\ \ w_1 = 0.1407, w_2 = 0.3386\ \&\ \ w_3 = 0.5907$
57	$if\ F(-1) = A_5 \wedge F(-2) = A_5 \wedge F(-3) = A_5$	$then\ \ w_1 = 0.0807, w_2 = 0.2992\ \&\ \ w_3 = 0.5286$
58	$if\ F(-1) = A_3 \wedge F(-2) = A_5$	$then\ \ w_1 = 0.4715\ \&\ w_2 = 0.7222$
59	$if\ F(-1) = A_7 \wedge F(-2) = A_3 \wedge F(-3) = A_5$	$then\ \ w_1 = ?, w_2 = ?\ \&\ \ w_3 = ?$

3.6 FORCASTING TEST DATA SET

Table 3.10 shows the test data set, of 30 records, to be forecasted.

Table 3.10: Erlang Traffic Observations (Test Set; 30 January, 2013 – 28 February, 2013)

Date	Traffic	Date	Traffic	Date	Traffic
30/01/13	171.264	**09/02/13**	230.679	**19/02/13**	152.082
31/01/13	166.890	**10/02/13**	204.754	**20/02/13**	88.029
01/02/13	176.522	**11/02/13**	137.068	**21/02/13**	178.357
02/02/13	96.502	**12/02/13**	113.689	**22/02/13**	176.959
03/02/13	135.146	**13/02/13**	148.392	**23/02/13**	244.465
04/02/13	136.067	**14/02/13**	202.684	**24/02/13**	145.498
05/02/13	115.478	**15/02/13**	236.503	**25/02/13**	153.308
06/02/13	171.586	**16/02/13**	72.496	**26/02/13**	190.242
07/02/13	201.352	**17/02/13**	164.978	**27/02/13**	169.463
08/02/13	181.668	**18/02/13**	138.881	**28/02/13**	186.015

First, the test data set is fuzzified by determining the Euclidean distances between a test observation and cluster centres of training data. A test observation will belong to a cluster centre if its distance, compared with it distances to other cluster centres, is smallest (minimum). Appendix E

shows the calculated distances. For instance, the calculated Euclidean distances for 30th January, 2013 are shown in the first row of Appendix E. A minimum distance (3.984) is obtained when cluster fourteen $\tilde{C}_{14}$ is used to compute the distance. Thus, the fuzzified result for the test observation (171.264) is A_5. Table 3.11 shows the complete results of the fuzzified test data set.

Table 3.11: Fuzzified Daily Erlang Traffic (Test Set; 30 January – 28 February, 2013)

Test Data Set	FUZZY SET	Test Data Set	FUZZY SET
171.264	A_5	202.684	A_8
166.89	A_5	236.503	A_{11}
176.522	A_6	72.496	A_1
96.502	A_1	164.978	A_5
135.146	A_2	138.881	A_2
136.067	A_2	152.082	A_3
115.478	A_2	88.029	A_1
171.586	A_5	178.357	A_6
201.352	A_8	176.959	A_6
181.668	A_6	244.465	A_{11}
230.679	A_{11}	145.498	A_3
204.754	A_8	153.308	A_3
137.068	A_2	190.242	A_7
113.689	A_1	169.463	A_5
148.392	A_3	186.015	A_7

Second, fuzzy set groups (FSGs) are determined for the test data set. Table 3.12 shows the fuzzy set groups (FSGs) for the test data.

Table 3.12: Established FSGs for Test Data Set

LABEL	Ambiguated FSGs	Disambiguated FSGs	LABEL	Ambiguated FSGs	Disambiguated FSGs
1	$\{A_5, A_5\}$	$\{A_5, A_5\}$	16	$\{A_8, A_{11}\}$	$\{A_8, A_{11}\}$
2	$\{A_5, A_6\}$	$\{A_5, A_6\}$	17	$\{A_{11}, A_1\}$	$\{A_{11}, A_1\}$
3	$\{A_6, A_1\}$	$\{A_6, A_1\}$	18	$\{A_1, A_5\}$	$\{A_1, A_5\}$
4	$\{A_1, A_2\}$	$\{A_1, A_2\}$	19	$\{A_5, A_2\}$	$\{A_5, A_2\}$
5	$\{A_2, A_2\}$	$\{A_1, A_2, A_2\}$	20	$\{A_2, A_3\}$	$\{A_2, A_3\}$
6	$\{A_2, A_2\}$	$\{A_2, A_2, A_2\}$	21	$\{A_3, A_1\}$	$\{A_3, A_1\}$
7	$\{A_2, A_5\}$	$\{A_2, A_5\}$	22	$\{A_1, A_6\}$	$\{A_1, A_6\}$
8	$\{A_5, A_8\}$	$\{A_5, A_8\}$	23	$\{A_6, A_6\}$	$\{A_6, A_6\}$
9	$\{A_8, A_6\}$	$\{A_8, A_6\}$	24	$\{A_6, A_{11}\}$	$\{A_6, A_6, A_{11}\}$
10	$\{A_6, A_{11}\}$	$\{A_8, A_6, A_{11}\}$	25	$\{A_{11}, A_3\}$	$\{A_{11}, A_3\}$
11	$\{A_{11}, A_8\}$	$\{A_{11}, A_8\}$	26	$\{A_3, A_3\}$	$\{A_3, A_3\}$
12	$\{A_8, A_2\}$	$\{A_8, A_2\}$	27	$\{A_3, A_7\}$	$\{A_3, A_7\}$
13	$\{A_2, A_1\}$	$\{A_2, A_1\}$	28	$\{A_7, A_5\}$	$\{A_7, A_5\}$
14	$\{A_1, A_3\}$	$\{A_1, A_3\}$	29	$\{A_5, A_7\}$	$\{A_5, A_7\}$
15	$\{A_3, A_8\}$	$\{A_3, A_8\}$			

Third, optimal weights are obtained for fuzzy sets in each fuzzy set group (FSG). The task here is to convert the test fuzzy set groups (FSGs) to "if-then" rules. Then, apply particle swarm optimization (PSO) to obtain optimal weights. Table 3.13 shows the complete "if-then" rules generated for the test data.

Table 3.13: Generated "if-then" Rules for Test Data Set

Rule	Matching Part	Weights
1	*if* $F(-1) = A_5 \wedge F(-2) = A_5$	*then* $w_1 = 0.6493$ & $w_2 = 0.4059$
2	*if* $F(-1) = A_6 \wedge F(-2) = A_5$	*then* $w_1 = 0.3958$ & $w_2 = 0.1388$
3	*if* $F(-1) = A_1 \wedge F(-2) = A_6$	*then* $w_1 = 0.6756$ & $w_2 = 0.4256$
4	*if* $F(-1) = A_2 \wedge F(-2) = A_1$	*then* $w_1 = 0.6482$ & $w_2 = 0.3934$
5	*if* $F(-1) = A_2 \wedge F(-2) = A_2 \wedge F(-3) = A_1$	*then* $w_1 = 0.5781$, $w_2 = 0.3301$ & $w_3 = 0.1281$
6	*if* $F(-1) = A_2 \wedge F(-2) = A_2 \wedge F(-3) = A_2$	*then* $w_1 = 0.6368$, $w_2 = 0.3863$ & $w_3 = 0.1864$
7	*if* $F(-1) = A_5 \wedge F(-2) = A_2$	*then* $w_1 = 0.77$ & $w_2 = 0.5200$
8	*if* $F(-1) = A_8 \wedge F(-2) = A_5$	*then* $w_1 = 0.5971$ & $w_2 = 0.3404$
9	*if* $F(-1) = A_6 \wedge F(-2) = A_8$	*then* $w_1 = 0.7472$ & $w_2 = 0.4960$
10	*if* $F(-1) = A_{11} \wedge F(-2) = A_6 \wedge F(-3) = A_8$	*then* $w_1 = 0.5558$, $w_2 = 0.3086$ & $w_3 = 0.0960$
11	*if* $F(-1) = A_5 \wedge F(-2) = A_2$	*then* $w_1 = 0.4505$ & $w_2 = 0.2009$
12	*if* $F(-1) = A_2 \wedge F(-2) = A_8$	*then* $w_1 = 0.4189$ & $w_2 = 0.1779$
13	*if* $F(-1) = A_1 \wedge F(-2) = A_2$	*then* $w_1 = 0.8031$ & $w_2 = 0.5600$
14	*if* $F(-1) = A_3 \wedge F(-2) = A_1$	*then* $w_1 = 0.9192$ & $w_2 = 0.6700$
15	*if* $F(-1) = A_8 \wedge F(-2) = A_3$	*then* $w_1 = 0.7600$ & $w_2 = 0.5100$
16	*if* $F(-1) = A_{11} \wedge F(-2) = A_8$	*then* $w_1 = 0.3309$ & $w_2 = 0.0813$
17	*if* $F(-1) = A_1 \wedge F(-2) = A_{11}$	*then* $w_1 = 0.6997$ & $w_2 = 0.4489$
18	*if* $F(-1) = A_5 \wedge F(-2) = A_1$	*then* $w_1 = 0.6235$ & $w_2 = 0.3745$
19	*if* $F(-1) = A_2 \wedge F(-2) = A_5$	*then* $w_1 = 0.6301$ & $w_2 = 0.3893$
20	*if* $F(-1) = A_3 \wedge F(-2) = A_2$	*then* $w_1 = 0.4404$ & $w_2 = 0.1926$
21	*if* $F(-1) = A_1 \wedge F(-2) = A_3$	*then* $w_1 = 0.8800$ & $w_2 = 0.6286$
22	*if* $F(-1) = A_6 \wedge F(-2) = A_1$	*then* $w_1 = 0.7495$ & $w_2 = 0.4998$
23	*if* $F(-1) = A_6 \wedge F(-2) = A_6$	*then* $w_1 = 0.7800$ & $w_2 = 0.5300$
24	*if* $F(-1) = A_{11} \wedge F(-2) = A_6 \wedge F(-3) = A_6$	*then* $w_1 = 0.4697$, $w_2 = 0.2181$ & $w_3 = 0.0181$
25	*if* $F(-1) = A_3 \wedge F(-2) = A_{11}$	*then* $w_1 = 0.5509$ & $w_2 = 0.3001$
27	*if* $F(-1) = A_7 \wedge F(-2) = A_3$	*then* $w_1 = 0.5999$ & $w_2 = 0.4970$
28	*if* $F(-1) = A_5 \wedge F(-2) = A_7$	*then* $w_1 = 0.6624$ & $w_2 = 0.4158$
Rule	**Matching Part**	**Weights**
29	*if* $F(-1) = A_7 \wedge F(-2) = A_5$	*then* $w_1 = ?$ & $w_2 = ?$

Finally, forecast and defuzzify the possible outcomes based on the generated "if-then" rule. Again, equation (2.20) is employed to compute forecast for the test data set.

3.7FORECASTING USING CHEN'S (1996) FUZZY TIME SERIES MODEL

The training data set was partitioned into $n = 14$ partitions. In the data set, the minimum and maximum records are $D_{\min} = 90.319$ and $D_{\max} = 319.248$ respectively. When b_0 and b_1 are set at 10.319 and 12.752 respectively, the universe of discourse defined is $U = [90,\ 332]$. The fuzzy sets defined are: $A_1 = [90,\ 98], A_2 = [98,\ 116], A_3 = [116,\ 134], ..., A_{14} = [314,\ 332]$. Appendix F and G show the fuzzification and established logical relationships respectively, for the Chen's (1996) model. Table 3.14 shows the forecasts obtained.

Table 3.14: Results of Forecasting Using Chen's (1996) FTS Model

Date	Training Data Set	Forecast	Date	Training Data Set	Forecast
12/1/2012	170.232	-	1/1/2013	315.685	305
12/2/2012	160.343	164	1/2/2013	291.862	224
12/3/2012	150.965	170	1/3/2013	211.912	260
12/4/2012	163.151	170	1/4/2013	175.319	206
12/5/2012	179.86	170	1/5/2013	93.802	164
12/6/2012	176.526	164	1/6/2013	90.319	125
12/7/2012	155.55	164	1/7/2013	153.194	125
12/8/2012	154.571	170	1/8/2013	183.423	170
12/9/2012	160.874	170	1/9/2013	188.883	164
12/10/2012	148.72	170	1/10/2013	134.05	170
12/11/2012	138.522	170	1/11/2013	139.442	170
12/12/2012	159.171	170	1/12/2013	187.514	170
12/13/2012	169.367	170	1/13/2013	221.6	164
12/14/2012	156.205	170	1/14/2013	232.545	206
12/15/2012	152.556	170	1/15/2013	260.859	269
12/16/2012	176.442	170	1/16/2013	319.248	323
12/17/2012	209.617	164	1/17/2013	158.539	224
12/18/2012	164.002	206	1/18/2013	184.075	170
12/19/2012	168.123	170	1/19/2013	153.452	164
12/20/2012	189.797	170	1/20/2013	157.5	170
12/21/2012	202.416	170	1/21/2013	150.486	170
12/22/2012	131.758	170	1/22/2013	187.741	170
12/23/2012	166.487	161	1/23/2013	141.908	164
12/24/2012	202.214	170	1/24/2013	143.357	170
12/25/2012	219.48	170	1/25/2013	166.878	170
12/26/2012	259.722	206	1/26/2013	166.439	170
12/27/2012	291.306	287	1/27/2013	159.809	170
12/28/2012	300.841	260	1/28/2013	151.401	170
12/29/2012	308.587	305	1/29/2013	191.997	170
12/30/2012	292.317	305	1/30/2013	-	170
12/31/2012	301.779	260			

3.8 FORECASTING USING CHENG *et al* (2008) HYBRID MODEL

This is a hybrid model that integrates fuzzy C-means (FCM) clustering into fuzzy time series (FTS) forecasting model. The training data was partitioned into $c = 14$ clusters. The result of fuzzification is the same as that shown in Table 3.4. Appendix H shows the fuzzy logical relationships (FLRs) and fuzzy relationship groups (FLRGs) obtained for the hybrid model. After defuzzification, the results of the forecasts are shown in Table 3.15.

Table 3.15:Forecasted Training Data Set Using Cheng's (2008) FCM/ FTS Hybrid Model

Date	Training Data Set	Fuzzy Set	Forecast	Date	Training Data Set	Fuzzy Set	Forecast
12/1/2012	170.232	A_5	-	1/1/2013	315.685	A_{14}	273.411
12/2/2012	160.343	A_4	173.744	1/2/2013	291.862	A_{13}	226.925
12/3/2012	150.965	A_3	171.097	1/3/2013	211.912	A_9	273.411
12/4/2012	163.151	A_4	165.522	1/4/2013	175.319	A_6	172.164
12/5/2012	179.86	A_6	171.097	1/5/2013	93.802	A_1	159.753
12/6/2012	176.526	A_6	159.753	1/6/2013	90.319	A_1	121.976
12/7/2012	155.55	A_4	159.753	1/7/2013	153.194	A_3	121.976
12/8/2012	154.571	A_3	171.097	1/8/2013	183.423	A_7	165.522
12/9/2012	160.874	A_4	165.522	1/9/2013	188.883	A_7	180.109
12/10/2012	148.72	A_3	171.097	1/10/2013	134.05	A_2	180.109
12/11/2012	138.522	A_2	165.522	1/11/2013	139.442	A_2	163.08
12/12/2012	159.171	A_4	163.08	1/12/2013	187.514	A_7	163.08
12/13/2012	169.367	A_5	171.097	1/13/2013	221.6	A_{10}	180.109
12/14/2012	156.205	A_4	173.744	1/14/2013	232.545	A_{11}	246.431
12/15/2012	152.556	A_3	171.097	1/15/2013	260.859	A_{12}	260.314
12/16/2012	176.442	A_6	165.522	1/16/2013	319.248	A_{14}	304.742
12/17/2012	209.617	A_9	159.753	1/17/2013	158.539	A_4	226.925
12/18/2012	164.002	A_5	172.164	1/18/2013	184.075	A_7	171.097
12/19/2012	168.123	A_5	173.744	1/19/2013	153.452	A_3	180.109
12/20/2012	189.797	A_7	173.744	1/20/2013	157.5	A_4	165.522
12/21/2012	202.416	A_8	180.109	1/21/2013	150.486	A_3	171.097
12/22/2012	131.758	A_2	179.125	1/22/2013	187.741	A_7	165.522
12/23/2012	166.487	A_5	163.08	1/23/2013	141.908	A_2	180.109
12/24/2012	202.214	A_8	173.744	1/24/2013	143.357	A_2	163.08
12/25/2012	219.48	A_{10}	179.125	1/25/2013	166.878	A_5	163.08
12/26/2012	259.722	A_{12}	246.431	1/26/2013	166.439	A_5	173.744
12/27/2012	291.306	A_{13}	304.742	1/27/2013	159.809	A_5	173.744
12/28/2012	300.841	A_{13}	273.411	1/28/2013	151.401	A_3	173.744
12/29/2012	308.587	A_{14}	273.411	1/29/2013	191.997	A_7	165.522
12/30/2012	292.317	A_{13}	226.925	1/30/2013	-	-	180.109
12/31/2012	301.779	A_{13}	273.411				

To forecast the test data set, the test data set is first fuzzified. The result of fuzzification is the same as that shown in Table 3.11. Then, possible forecasts are obtained based on fuzzy logical

relationships of the training data set shown in appendix H. The calculated forecast is shown in Table 3.16.

Table 3.16: Forecasted Test Data Set Using Cheng (2008) Hybrid Model

Date	Actual	Forecast
1/30/2013	171.264	-
1/31/2013	166.89	173.744
2/1/2013	176.522	169.069
2/2/2013	96.502	159.753
2/3/2013	135.146	145.198
2/4/2013	136.067	163.08
2/5/2013	115.478	163.08
2/6/2013	171.586	163.08
2/7/2013	201.352	169.069
2/8/2013	181.668	191.961
2/9/2013	230.679	174.311
2/10/2013	204.754	176.625
2/11/2013	137.068	179.125
2/12/2013	113.689	149.378
2/13/2013	148.392	121.976
2/14/2013	202.684	158.324
2/15/2013	236.503	191.961
2/16/2013	72.496	176.625
2/17/2013	164.978	145.198
2/18/2013	138.881	169.069
2/19/2013	152.082	149.378
2/20/2013	88.029	158.324
2/21/2013	178.357	145.198
2/22/2013	176.959	159.753
2/23/2013	244.465	174.311
2/24/2013	145.498	176.625
2/25/2013	153.308	158.324
2/26/2013	190.242	165.522
2/27/2013	169.463	177.971
2/28/2013	186.015	169.069

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 INTRODUCTION

This section discusses the forecasting results obtained for the training and testing data set and the validation of the proposed hybrid forecasting model. The area of validation covered here is the forecasting performances of the developed hybrid model, a conventional fuzzy time series model (Chen's (1996) model) and an existing fuzzy time series model that integrates fuzzy c-means clustering algorithm (Cheng et al (2008) model).

4.2 FORECASTING RESULTS FOR TRAINING DATA SET

By utilizing the optimal weights, shown in Table 3.8 and equation (3.20) the forecast for the training data set, using the proposed hybrid model, are evaluated. The result obtained is presented in Table 4.1.

Table 4.1: Proposed Hybrid Model Forecasts for Training Data Set

Date	Actual	Forecasted	Date	Actual	Forecasted
12/1/2012	170.232	-	12/31/2012	301.779	303.056
12/2/2012	160.343	-	1/1/2013	315.685	316.014
12/3/2012	150.965	-	1/2/2013	291.862	291.96
12/4/2012	163.151	-	1/3/2013	211.912	212.222
12/5/2012	179.86	180.536	1/4/2013	175.319	175.807
12/6/2012	176.526	177.369	1/5/2013	93.802	94.088
12/7/2012	155.55	156.226	1/6/2013	90.319	90.22
12/8/2012	154.571	156.359	1/7/2013	153.194	151.917
12/9/2012	160.874	161.409	1/8/2013	183.423	181.901
12/10/2012	148.72	148.058	1/9/2013	188.883	190.505
12/11/2012	138.522	139.753	1/10/2013	134.05	134.475
12/12/2012	159.171	159.907	1/11/2013	139.442	138.87
12/13/2012	169.367	171.18	1/12/2013	187.514	189.761
12/14/2012	156.205	157.501	1/13/2013	221.6	219.77
12/15/2012	152.556	151.571	1/14/2013	232.545	234.203
12/16/2012	176.442	176.746	1/15/2013	260.859	260.482
12/17/2012	209.617	210.029	1/16/2013	319.248	317.454
12/18/2012	164.002	164.584	1/17/2013	158.539	156.928
12/19/2012	168.123	167.557	1/18/2013	184.075	185.28
12/20/2012	189.797	188.692	1/19/2013	153.452	155.091
12/21/2012	202.416	203.171	1/20/2013	157.5	157.789
12/22/2012	131.758	132.757	1/21/2013	150.486	151.123
12/23/2012	166.487	165.973	1/22/2013	187.741	188.847
12/24/2012	202.214	202.001	1/23/2013	141.908	142.702
12/25/2012	219.48	219.585	1/24/2013	143.357	144.711
12/26/2012	259.722	260.379	1/25/2013	166.878	167.274
12/27/2012	291.306	290.999	1/26/2013	166.439	166.995
12/28/2012	300.841	300.35	1/27/2013	159.809	158.958
12/29/2012	308.587	308.493	1/28/2013	151.401	152.292
12/30/2012	292.317	292.763	1/29/2013	191.997	192.222

Table 4.2 shows the result obtained when the weights of the training data set were reversed.

Table 4.2: Proposed Hybrid Model Forecasts for Training Data Set during Weights Reversal

Date	Actual	Forecasted	Date	Actual	Forecasted
12/1/2012	170.232	-	12/31/2012	301.779	303.913
12/2/2012	160.343	-	1/1/2013	315.685	315.649
12/3/2012	150.965	-	1/2/2013	291.862	291.942
12/4/2012	163.151	-	1/3/2013	211.912	210.298
12/5/2012	179.86	180.438	1/4/2013	175.319	176.329
12/6/2012	176.526	176.585	1/5/2013	93.802	92.506
12/7/2012	155.55	156.811	1/6/2013	90.319	91.277
12/8/2012	154.571	155.796	1/7/2013	153.194	151.715
12/9/2012	160.874	160.259	1/8/2013	183.423	182.567
12/10/2012	148.72	149.44	1/9/2013	188.883	188.357
12/11/2012	138.522	137.546	1/10/2013	134.05	133.082
12/12/2012	159.171	158.897	1/11/2013	139.442	139.528
12/13/2012	169.367	170.254	1/12/2013	187.514	188.829
12/14/2012	156.205	156.811	1/13/2013	221.6	220.188
12/15/2012	152.556	152.884	1/14/2013	232.545	232.251
12/16/2012	176.442	176.757	1/15/2013	260.859	259.184
12/17/2012	209.617	208.888	1/16/2013	319.248	319.353
12/18/2012	164.002	163.866	1/17/2013	158.539	158.062
12/19/2012	168.123	168.183	1/18/2013	184.075	184.941
12/20/2012	189.797	189.762	1/19/2013	153.452	153.388
12/21/2012	202.416	200.523	1/20/2013	157.5	157.323
12/22/2012	131.758	132.586	1/21/2013	150.486	150.02
12/23/2012	166.487	165.994	1/22/2013	187.741	139.336
12/24/2012	202.214	203.877	1/23/2013	141.908	141.696
12/25/2012	219.48	221.253	1/24/2013	143.357	144.087
12/26/2012	259.722	259.633	1/25/2013	166.878	166.146
12/27/2012	291.306	293.119	1/26/2013	166.439	166.075
12/28/2012	300.841	300.949	1/27/2013	159.809	161.524
12/29/2012	308.587	309.381	1/28/2013	151.401	151.974
12/30/2012	292.317	293.074	1/29/2013	191.997	192.421

As seen from Table 4.1 and 4.2, the first four years are not forecasted. This is due to the fact that at least two preceding historical data are required to forecast any future observation. Also the fuzzy set groups (FSGs) associated with the third and fourth future observations were extended to the third and fourth order, respectively to remove ambiguity. The plot for the forecasted training data set and actual training data set is shown in Figure 4.1.

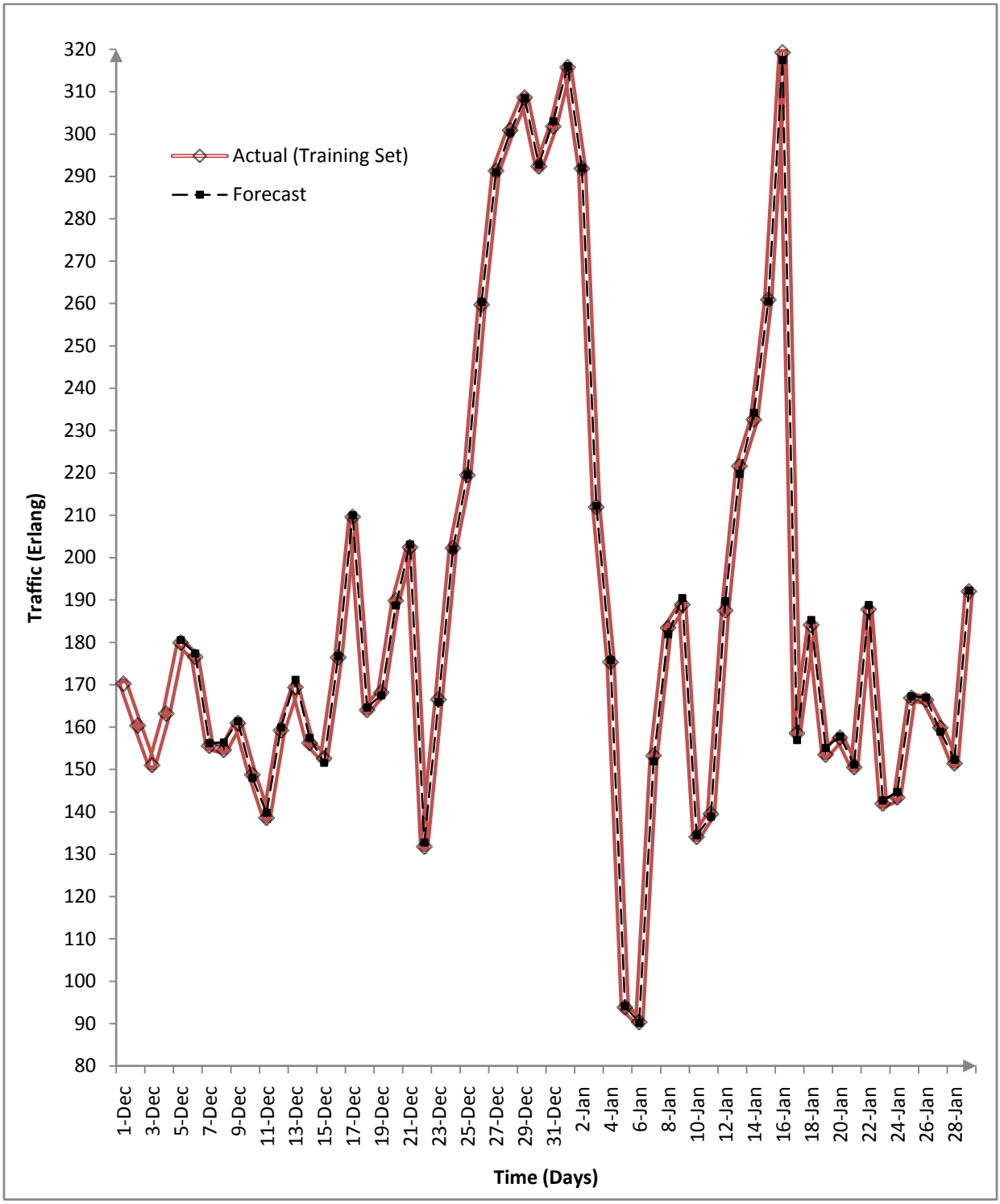


Figure 4.1: Plot of Actual and Forecasted Erlang Traffic against Time for Training Data Set

As seen from Figure 4.1, the actual and forecasted plots are very similar. Similarly, the plot of actual training data set and forecasted results when the weights are reversed is shown in Figure 4.2.

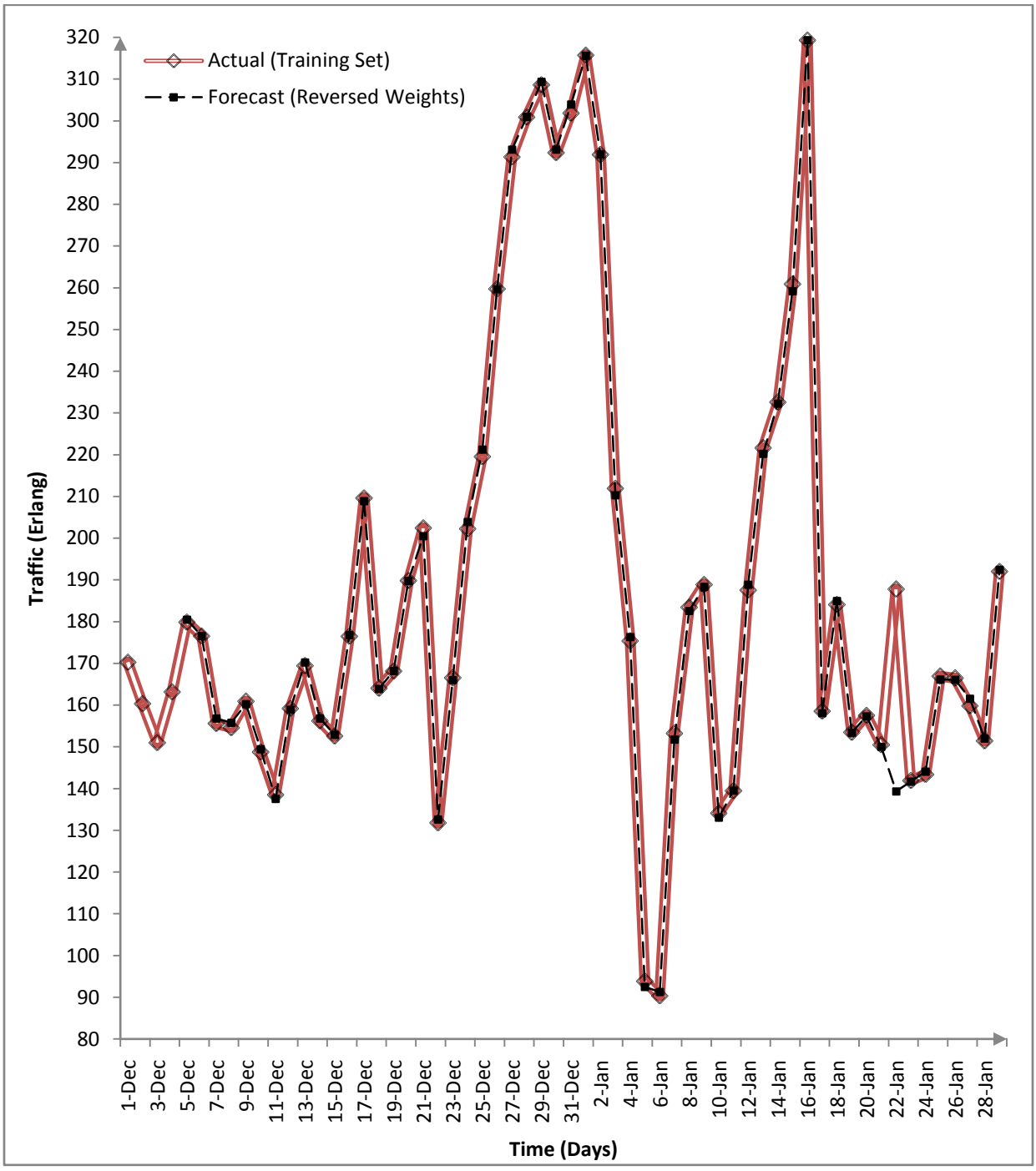


Figure 4.2: Plot of Actual Training Data Set and Forecasted (Reversed Weights) Erlang against Time

As seen from Figure 4.2, the plot of the forecasted data set for the reversed weights is similar to that of the actual training data plot. A deviation occurred a time 22nd January, 2013.

4.3 FORECASTING RESULT FOR TEST DATA SET

Table 4.3 shows the result of evaluating the forecast of the test data set.

Table 4.3: Forecasted Voice (Erlang) Traffic of Test Set for the Proposed Hybrid Model

Date	Actual	Forecast
1/30/2013	171.264	-
1/31/2013	166.89	-
2/1/2013	176.522	176.514
2/2/2013	96.502	93.294
2/3/2013	135.146	137.554
2/4/2013	136.067	138.783
2/5/2013	115.478	136.865
2/6/2013	171.586	166.564
2/7/2013	201.352	200.416
2/8/2013	181.668	177.696
2/9/2013	230.679	232.598
2/10/2013	204.754	203.301
2/11/2013	137.068	137.825
2/12/2013	113.689	93.665
2/13/2013	148.392	151.062
2/14/2013	202.684	201.296
2/15/2013	236.503	231.157
2/16/2013	72.496	93.391
2/17/2013	164.978	168.812
2/18/2013	138.881	138.78
2/19/2013	152.082	151.895
2/20/2013	88.029	93.411
2/21/2013	178.357	176.494
2/22/2013	176.959	178.714
2/23/2013	244.465	231.932
2/24/2013	145.498	151.046
2/25/2013	153.308	153.458
2/26/2013	190.242	189.06
2/27/2013	169.463	167.218
2/28/2013	186.015	189.052

To further verify the proposed model, a plot of the actual and the predicted Erlang traffic for the testing set is shown in Figure 4.3.

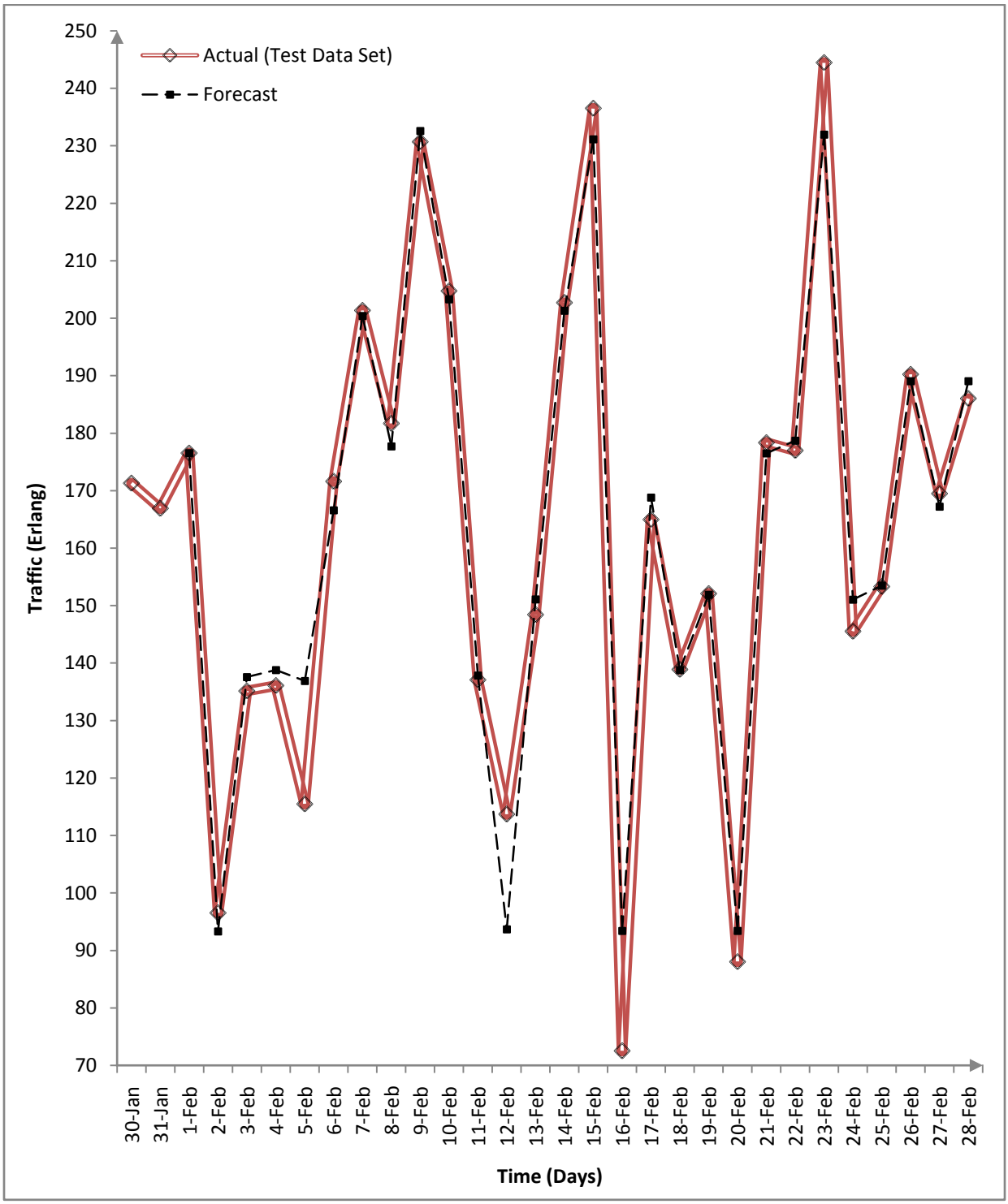


Figure 4.3: Plot of Actual and Forecasted Erlang against Time for Test Data Set

4.4 VALIDATION

To verify the proposed hybrid model forecasting accuracy, its performance measures of mean squared error (MSE) and mean absolute percentage error (MAPE) are determined. Subsequently, these measures for the proposed hybrid model are compared with those of some existing models. These existing models includes; Chen's (1996) model and Cheng et al (2008) hybrid model. Chen's (1996) model is a training model while Cheng et al (2008) hybrid model is both training and testing model. To determine the mean square error (MSE) and mean absolute (MAPE) of these models, equation (4.24) and (4.28), respectively, are utilized. Table 4.4 shows the calculation of the performance measures for the proposed hybrid model.

Table 4.4: Calculation of MSE and MAPE of Forecast forTraining the Proposed Model

Date	$actual_t$	$forecast_t$	SE_t	AE_t
12/1/2012	170.232	-	-	-
12/2/2012	160.343	-	-	-
12/3/2012	150.965	-	-	-
12/4/2012	163.151	-	-	-
12/5/2012	179.86	180.536	0.456976	0.003758479
12/6/2012	176.526	177.369	0.710649	0.0047755
12/7/2012	155.55	156.226	0.456976	0.004345869
12/8/2012	154.571	156.359	3.196944	0.0115675
12/9/2012	160.874	161.409	0.286225	0.003325584
12/10/2012	148.72	148.058	0.438244	0.004451318
12/11/2012	138.522	139.753	1.515361	0.008886675
12/12/2012	159.171	159.907	0.541696	0.004623958
12/13/2012	169.367	171.18	3.286969	0.010704565
12/14/2012	156.205	157.501	1.679616	0.008296789
12/15/2012	152.556	151.571	0.970225	0.006456645
12/16/2012	176.442	176.746	0.092416	0.001722946
12/17/2012	209.617	210.029	0.169744	0.001965489
12/18/2012	164.002	164.584	0.338724	0.003548737
12/19/2012	168.123	167.557	0.320356	0.003366583
12/20/2012	189.797	188.692	1.221025	0.00582201
12/21/2012	202.416	203.171	0.570025	0.003729942
12/22/2012	131.758	132.757	0.998001	0.007582082
12/23/2012	166.487	165.973	0.264196	0.003087328
12/24/2012	202.214	202.001	0.045369	0.00105334
12/25/2012	219.48	219.585	0.011025	0.000478403
12/26/2012	259.722	260.379	0.431649	0.002529628
12/27/2012	291.306	290.999	0.094249	0.001053875
12/28/2012	300.841	300.35	0.241081	0.001632091
12/29/2012	308.587	308.493	0.008836	0.000304614
12/30/2012	292.317	292.763	0.198916	0.001525741
12/31/2012	301.779	303.056	1.630729	0.004231573
1/1/2013	315.685	316.014	0.108241	0.001042178
1/2/2013	291.862	291.96	0.009604	0.000335775
1/3/2013	211.912	212.222	0.0961	0.001462871
1/4/2013	175.319	175.807	0.238144	0.002783498
1/5/2013	93.802	94.088	0.081796	0.003048976
1/6/2013	90.319	90.22	0.009801	0.001096115
1/7/2013	153.194	151.917	1.630729	0.008335836
1/8/2013	183.423	181.901	2.316484	0.00829776
1/9/2013	188.883	190.505	2.630884	0.008587327
1/10/2013	134.05	134.475	0.180625	0.003170459
1/11/2013	139.442	138.87	0.327184	0.004102064
1/12/2013	187.514	189.761	5.049009	0.011983105
1/13/2013	221.6	219.77	3.3489	0.008258123
1/14/2013	232.545	234.203	2.748964	0.007129803
1/15/2013	260.859	260.482	0.142129	0.001445225
1/16/2013	319.248	317.454	3.218436	0.005619456
1/17/2013	158.539	156.928	2.595321	0.010161538
1/18/2013	184.075	185.28	1.452025	0.006546245
1/19/2013	153.452	155.091	2.686321	0.010680864
1/20/2013	157.5	157.789	0.083521	0.001834921
1/21/2013	150.486	151.123	0.405769	0.004232952
1/22/2013	187.741	188.847	1.223236	0.005891095
1/23/2013	141.908	142.702	0.630436	0.005595174
1/24/2013	143.357	144.711	1.833316	0.009444952
1/25/2013	166.878	167.274	0.156816	0.002372991
1/26/2013	166.439	166.995	0.309136	0.003340563
1/27/2013	159.809	158.958	0.724201	0.005325107
1/28/2013	151.401	152.292	0.793881	0.005885034
1/29/2013	191.997	192.222	0.050625	0.001171893
			MSE = 0.9867	MAPE = 0.004714

The result shows that statistical measures of $MSE = 0.9867$ and $MAPE = 0.47\ \%$ were obtained for the training phase of the proposed hybrid model. As seen from Table 4.4, the mean square error value indicates a low quadratic loss while the mean absolute percentage error obtained indicated a low absolute loss.

Table 4.5 shows the calculation of the performance measures for the proposed hybrid model when the weights were reversed.

Table 4.5: MSE and MAPE Calculationfor Training the Propsed Model (Reversed Weights)

Date	$actual_t$	$forecast_t$	SE_t	AE_t
12/1/2012	170.232	-	-	-
12/2/2012	160.343	-	-	-
12/3/2012	150.965	-	-	-
12/4/2012	163.151	-	-	-
12/5/2012	179.86	180.438	0.334084	0.003213611
12/6/2012	176.526	176.585	0.003481	0.000334228
12/7/2012	155.55	156.811	1.590121	0.008106718
12/8/2012	154.571	155.796	1.500625	0.007925161
12/9/2012	160.874	160.259	0.378225	0.003822868
12/10/2012	148.72	149.44	0.5184	0.004841313
12/11/2012	138.522	137.546	0.952576	0.007045812
12/12/2012	159.171	158.897	0.075076	0.001721419
12/13/2012	169.367	170.254	0.786769	0.005237148
12/14/2012	156.205	156.811	0.367236	0.003879517
12/15/2012	152.556	152.884	0.107584	0.00215003
12/16/2012	176.442	176.757	0.099225	0.001785289
12/17/2012	209.617	208.888	0.531441	0.003477771
12/18/2012	164.002	163.866	0.018496	0.000829258
12/19/2012	168.123	168.183	0.0036	0.000356882
12/20/2012	189.797	189.762	0.001225	0.000184408
12/21/2012	202.416	200.523	3.583449	0.009352028
12/22/2012	131.758	132.586	0.685584	0.006284248
12/23/2012	166.487	165.994	0.243049	0.002961192
12/24/2012	202.214	203.877	2.765569	0.008223961
12/25/2012	219.48	221.253	3.143529	0.008078185
12/26/2012	259.722	259.633	0.007921	0.000342674
12/27/2012	291.306	293.119	3.286969	0.006223696
12/28/2012	300.841	300.949	0.011664	0.000358994
12/29/2012	308.587	309.381	0.630436	0.002573018
12/30/2012	292.317	293.074	0.573049	0.002589654
12/31/2012	301.779	303.913	4.553956	0.0070714
1/1/2013	315.685	315.649	0.001296	0.000114038
1/2/2013	291.862	291.942	0.0064	0.000274102
1/3/2013	211.912	210.298	2.604996	0.007616369
1/4/2013	175.319	176.329	1.0201	0.005760927
1/5/2013	93.802	92.506	1.679616	0.013816337
1/6/2013	90.319	91.277	0.917764	0.010606849
1/7/2013	153.194	151.715	2.187441	0.009654425
1/8/2013	183.423	182.567	0.732736	0.004666808
1/9/2013	188.883	188.357	0.276676	0.002784793
1/10/2013	134.05	133.082	0.937024	0.007221186
1/11/2013	139.442	139.528	0.007396	0.000616744
1/12/2013	187.514	188.829	1.729225	0.00701281
1/13/2013	221.6	220.188	1.993744	0.006371841
1/14/2013	232.545	232.251	0.086436	0.001264271
1/15/2013	260.859	259.184	2.805625	0.006421093
1/16/2013	319.248	319.353	0.011025	0.000328898
1/17/2013	158.539	158.062	0.227529	0.003008723
1/18/2013	184.075	184.941	0.749956	0.004704604
1/19/2013	153.452	153.388	0.004096	0.000417069
1/20/2013	157.5	157.323	0.031329	0.00112381
1/21/2013	150.486	150.02	0.217156	0.003096634
1/22/2013	187.741	139.336	2343.044025	0.257828604
1/23/2013	141.908	141.696	0.044944	0.001493926
1/24/2013	143.357	144.087	0.5329	0.005092182
1/25/2013	166.878	166.146	0.535824	0.004386438
1/26/2013	166.439	166.075	0.132496	0.002186987
1/27/2013	159.809	161.524	2.941225	0.010731561
1/28/2013	151.401	151.974	0.328329	0.003784651
1/29/2013	191.997	192.421	0.179776	0.002208368
			MSE = 42.7272	MAPE = 0.00881367

The effect of reversed weights investigated, based on the statistical measures of $MSE = 42.7272$ and $MAPE = 0.88\ \%$, indicated that it has to some extent, an effect on the forecasting accuracy.

Table 4.6 shows the calculation of the performance measures for the proposed hybrid model for the test data set.

Table 4.6: Calculation of MSE and MAPE for Testing theProposed Hybrid Model

Date	$Actual_t$	$Forecast_t$	SE_t	AE_t
1/30/2013	171.264	-	-	-
1/31/2013	166.89	-	-	-
2/1/2013	176.522	176.514	6.4E-05	4.53201E-05
2/2/2013	96.502	93.294	10.291264	0.033242834
2/3/2013	135.146	137.554	5.798464	0.017817767
2/4/2013	136.067	138.783	7.376656	0.019960755
2/5/2013	115.478	136.865	457.403769	0.185204108
2/6/2013	171.586	166.564	25.220484	0.029268122
2/7/2013	201.352	200.416	0.876096	0.004648576
2/8/2013	181.668	177.696	15.776784	0.02186406
2/9/2013	230.679	232.598	3.682561	0.008318919
2/10/2013	204.754	203.301	2.111209	0.00709632
2/11/2013	137.068	137.825	0.573049	0.005522806
2/12/2013	113.689	93.665	400.960576	0.176129617
2/13/2013	148.392	151.062	7.1289	0.017992884
2/14/2013	202.684	201.296	1.926544	0.006848099
2/15/2013	236.503	231.157	28.579716	0.022604364
2/16/2013	72.496	93.391	436.601025	0.288222798
2/17/2013	164.978	168.812	14.699556	0.023239462
2/18/2013	138.881	138.78	0.010201	0.000727241
2/19/2013	152.082	151.895	0.034969	0.0012296
2/20/2013	88.029	93.411	28.965924	0.061138943
2/21/2013	178.357	176.494	3.470769	0.010445343
2/22/2013	176.959	178.714	3.080025	0.009917552
2/23/2013	244.465	231.932	157.076089	0.051267053
2/24/2013	145.498	151.046	30.780304	0.038131108
2/25/2013	153.308	153.458	0.0225	0.000978423
2/26/2013	190.242	189.06	1.397124	0.006213139
2/27/2013	169.463	167.218	5.040025	0.01324773
2/28/2013	186.015	189.052	9.223369	0.01632664
			MSE = 59.21814	MAPE = 0.03849

As seen from Table 4.6, the forecasting performance of the proposed hybrid model on the test data set indicated measures of $MSE = 59.22$ and $MAPE = 3.85$ %.

Table 4.7 shows the calculation of the performance measures for the Chen's (1996) FTS model.

Table 4.7: Calculation of MSE and MAPE of Forecasts obtained for Chen's (1996) Model

Date	$actual_t$	$forecast_t$	SE_t	AE_t
12/1/2012	170.232	-	-	-
12/2/2012	160.343	164	13.373649	0.022807357
12/3/2012	150.965	170	362.331225	0.126088829
12/4/2012	163.151	170	46.908801	0.041979516
12/5/2012	179.86	170	97.2196	0.054820416
12/6/2012	176.526	164	156.900676	0.070958386
12/7/2012	155.55	164	71.4025	0.054323369
12/8/2012	154.571	170	238.054041	0.099818207
12/9/2012	160.874	170	83.283876	0.056727625
12/10/2012	148.72	170	452.8384	0.143087682
12/11/2012	138.522	170	990.864484	0.227241882
12/12/2012	159.171	170	117.267241	0.06803375
12/13/2012	169.367	170	0.400689	0.003737446
12/14/2012	156.205	170	190.302025	0.088313434
12/15/2012	152.556	170	304.293136	0.114344896
12/16/2012	176.442	170	41.499364	0.036510581
12/17/2012	209.617	164	2080.910689	0.217620708
12/18/2012	164.002	206	1763.832004	0.256082243
12/19/2012	168.123	170	3.523129	0.011164445
12/20/2012	189.797	170	391.921209	0.10430618
12/21/2012	202.416	170	1050.797056	0.160145443
12/22/2012	131.758	170	1462.450564	0.290244236
12/23/2012	166.487	161	30.107169	0.032957528
12/24/2012	202.214	170	1037.741796	0.159306477
12/25/2012	219.48	170	2448.2704	0.225441954
12/26/2012	259.722	206	2886.053284	0.206844241
12/27/2012	291.306	287	18.541636	0.014781707
12/28/2012	300.841	260	1667.987281	0.135756097
12/29/2012	308.587	305	12.866569	0.01162395
12/30/2012	292.317	305	160.858489	0.043387829
12/31/2012	301.779	260	1745.484841	0.13844237
1/1/2013	315.685	305	114.169225	0.033847031
1/2/2013	291.862	224	4605.251044	0.232513996
1/3/2013	211.912	260	2312.455744	0.226924384
1/4/2013	175.319	206	941.323761	0.175000998
1/5/2013	93.802	164	4927.759204	0.748363574
1/6/2013	90.319	125	1202.771761	0.383983436
1/7/2013	153.194	125	794.901636	0.18404115
1/8/2013	183.423	170	180.176929	0.073180572
1/9/2013	188.883	164	619.163689	0.131737637
1/10/2013	134.05	170	1292.4025	0.268183514
1/11/2013	139.442	170	933.791364	0.219144877
1/12/2013	187.514	170	306.740196	0.093401026
1/13/2013	221.6	164	3317.76	0.259927798
1/14/2013	232.545	206	704.637025	0.114149949
1/15/2013	260.859	269	66.275881	0.031208431
1/16/2013	319.248	323	14.077504	0.011752619
1/17/2013	158.539	224	4285.142521	0.412901557
1/18/2013	184.075	170	198.105625	0.076463398
1/19/2013	153.452	164	111.260304	0.068738107
1/20/2013	157.5	170	156.25	0.079365079
1/21/2013	150.486	170	380.796196	0.129673192
1/22/2013	187.741	170	314.743081	0.094497206
1/23/2013	141.908	164	488.056464	0.155678327
1/24/2013	143.357	170	709.849449	0.185850708
1/25/2013	166.878	170	9.746884	0.018708278
1/26/2013	166.439	170	12.680721	0.021395226
1/27/2013	159.809	170	103.856481	0.063769875
1/28/2013	151.401	170	345.922801	0.122845952
1/29/2013	191.997	170	483.868009	0.114569498
			MSE = 845.122	MAPE = 0.1347

As seen from Table 4.7, when the Chen's (1996) model was implemented in forecasting the training data set, the performance results indicated statistical measures of $MSE = 845.122$ and $MAPE = 13.47\ \%$. Figure 4.4 show the plot of the Chen's (1996) forecasts for the training data set against time.

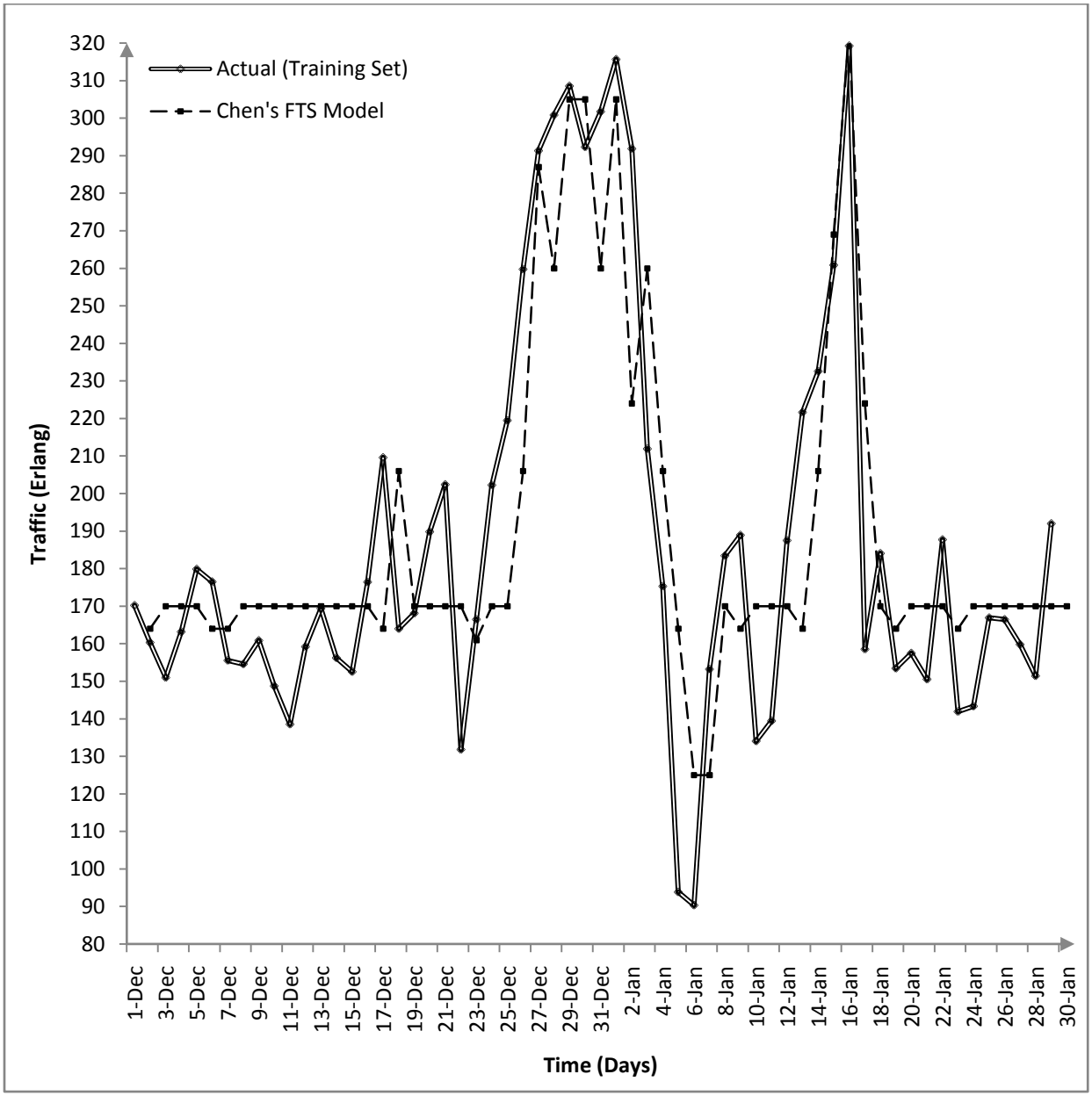


Figure 4.4: Plot of Actual (Training Set) and Forecasted Erlang (Chen's 1996 Model) against Time

Similarly, when the Cheng (2008) model was implemented in forecasting the training data set, Statistical performance measures of $MSE = 856.145$ and $MAPE = 13.37\ \%$ were obtained. Table 4.8 shows the calculation of the performance measures for the Cheng's (2008) FTS/ FCM hybrid model during the training phase.

Table 4.8: Calculation of MSE and MAPE for Forecast obtained for Cheng's (2008) Model Training

Date	$actual_t$	$forecast_t$	SE_t	AE_t
12/1/2012	170.232	–	-	-
12/2/2012	160.343	173.744	179.586801	0.083577082
12/3/2012	150.965	171.097	405.297424	0.133355414
12/4/2012	163.151	165.522	5.621641	0.01453255
12/5/2012	179.86	171.097	76.790169	0.048721228
12/6/2012	176.526	159.753	281.333529	0.095017165
12/7/2012	155.55	159.753	17.665209	0.027020251
12/8/2012	154.571	171.097	273.108676	0.106915269
12/9/2012	160.874	165.522	21.603904	0.028892176
12/10/2012	148.72	171.097	500.730129	0.150463959
12/11/2012	138.522	165.522	729	0.194914887
12/12/2012	159.171	163.08	15.280281	0.024558494
12/13/2012	169.367	171.097	2.9929	0.010214505
12/14/2012	156.205	173.744	307.616521	0.112281937
12/15/2012	152.556	171.097	343.768681	0.121535698
12/16/2012	176.442	165.522	119.2464	0.061890026
12/17/2012	209.617	159.753	2486.418496	0.23788147
12/18/2012	164.002	172.164	66.618244	0.049767686
12/19/2012	168.123	173.744	31.595641	0.033433855
12/20/2012	189.797	173.744	257.698809	0.084579841
12/21/2012	202.416	180.109	497.602249	0.110203739
12/22/2012	131.758	179.125	2243.632689	0.359499992
12/23/2012	166.487	163.08	11.607649	0.02046406
12/24/2012	202.214	173.744	810.5409	0.140791439
12/25/2012	219.48	179.125	1628.526025	0.183866412
12/26/2012	259.722	246.431	176.650681	0.051173948
12/27/2012	291.306	304.742	180.526096	0.04612332
12/28/2012	300.841	273.411	752.4049	0.091177732
12/29/2012	308.587	273.411	1237.350976	0.113990544
12/30/2012	292.317	226.925	4276.113664	0.223702351
12/31/2012	301.779	273.411	804.743424	0.094002565
1/1/2013	315.685	273.411	1787.091076	0.133911969
1/2/2013	291.862	226.925	4216.813969	0.222492137
1/3/2013	211.912	273.411	3782.127001	0.290210087
1/4/2013	175.319	172.164	9.954025	0.017995768
1/5/2013	93.802	159.753	4349.534401	0.703087354
1/6/2013	90.319	121.976	1002.165649	0.350502109
1/7/2013	153.194	121.976	974.563524	0.203780827
1/8/2013	183.423	165.522	320.445801	0.097594086
1/9/2013	188.883	180.109	76.983076	0.046452036
1/10/2013	134.05	180.109	2121.431481	0.343595673
1/11/2013	139.442	163.08	558.755044	0.169518509
1/12/2013	187.514	163.08	597.020356	0.130304937
1/13/2013	221.6	180.109	1721.503081	0.187233755
1/14/2013	232.545	246.431	192.820996	0.059713174
1/15/2013	260.859	260.314	0.297025	0.002089251
1/16/2013	319.248	304.742	210.424036	0.045438029
1/17/2013	158.539	226.925	4676.644996	0.431351276
1/18/2013	184.075	171.097	168.428484	0.070503871
1/19/2013	153.452	180.109	710.595649	0.173715559
1/20/2013	157.5	165.522	64.352484	0.050933333
1/21/2013	150.486	171.097	424.813321	0.136962907
1/22/2013	187.741	165.522	493.683961	0.118349215
1/23/2013	141.908	180.109	1459.316401	0.269195535
1/24/2013	143.357	163.08	388.996729	0.137579609
1/25/2013	166.878	163.08	14.424804	0.022759141
1/26/2013	166.439	173.744	53.363025	0.043889954
1/27/2013	159.809	173.744	194.184225	0.087197842
1/28/2013	151.401	173.744	499.209649	0.147574983
1/29/2013	191.997	165.522	700.925625	0.13789278
			MSE = 856.145	MAPE = 0.133667

Figure 4.5 shows the plot of the Cheng (2008) forecasts for the training data set against time.

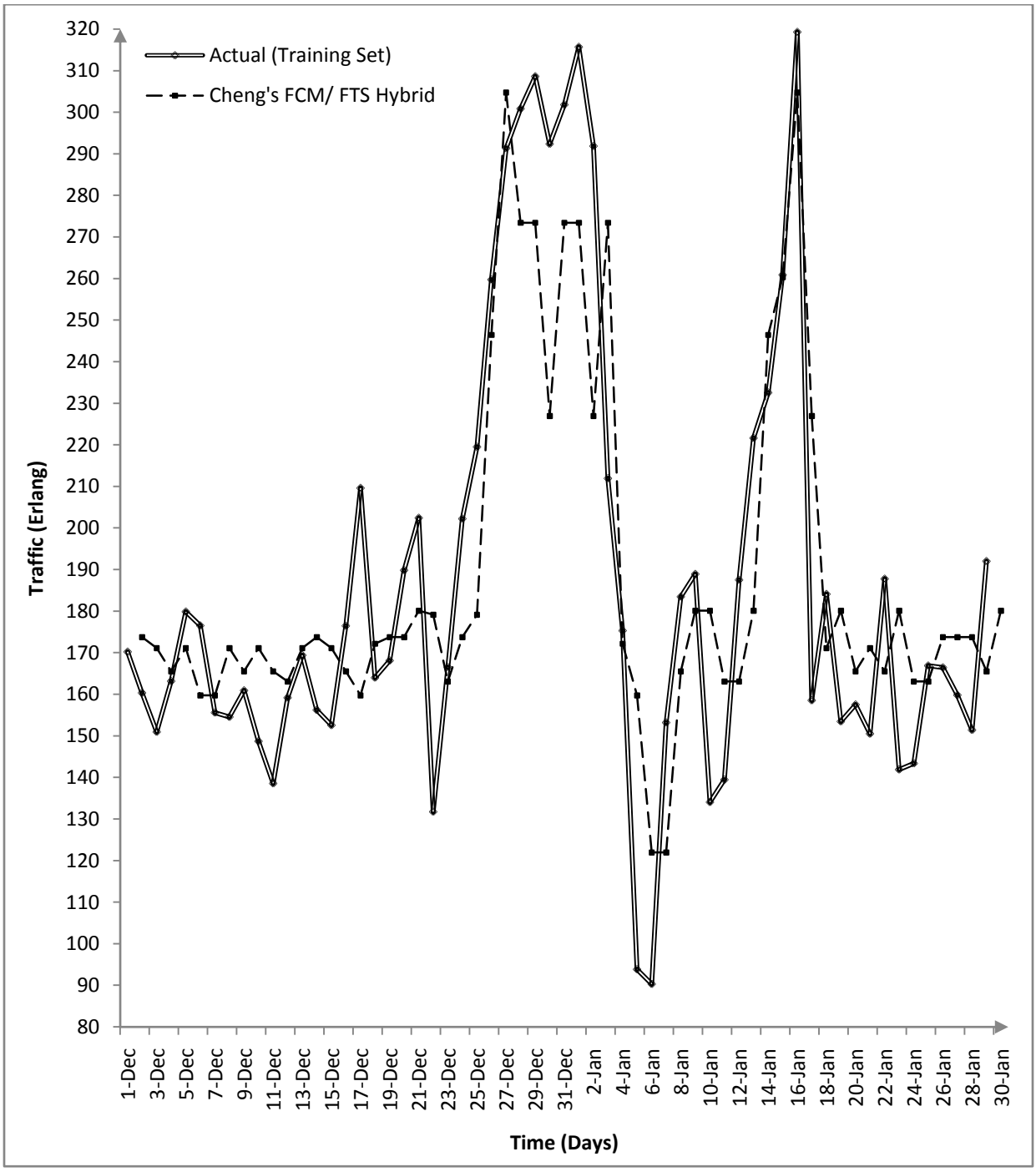


Figure 4.5: Plot of Actual (Training Set) and Forecasted Erlang (Cheng's 2008 Hybrid Model) against Time

Statistical measures of $MSE = 1567.4$ and $MAPE = 23.98\ \%$ were obtained when the Cheng (2008) model was implemented on the test data set. Table 4.9 shows the calculation of the performance measures for the Cheng's (2008) FTS/ FCM hybrid model during the testing phase.

Table 4.9: Calculation of MSE and MAPE for Forecast obtained for Cheng's (2008) Model Testing

Date	$Actual_t$	$Forecast_t$	SE_t	AE_t
1/30/2013	171.264	-	-	-
1/31/2013	166.89	173.744	46.977316	0.041068968
2/1/2013	176.522	169.069	55.547209	0.042221366
2/2/2013	96.502	159.753	4000.689001	0.655437193
2/3/2013	135.146	145.198	101.042704	0.07437882
2/4/2013	136.067	163.08	729.702169	0.198527196
2/5/2013	115.478	163.08	2265.950404	0.412217046
2/6/2013	171.586	163.08	72.352036	0.049572809
2/7/2013	201.352	169.069	1042.192089	0.160331161
2/8/2013	181.668	191.961	105.945849	0.0566583
2/9/2013	230.679	174.311	3177.351424	0.244356877
2/10/2013	204.754	176.625	791.240641	0.13737949
2/11/2013	137.068	179.125	1768.791249	0.306833105
2/12/2013	113.689	149.378	1273.704721	0.313917793
2/13/2013	148.392	121.976	697.805056	0.178014987
2/14/2013	202.684	158.324	1967.8096	0.21886286
2/15/2013	236.503	191.961	1983.989764	0.188335877
2/16/2013	72.496	176.625	10842.84864	1.436341315
2/17/2013	164.978	145.198	391.2484	0.119894774
2/18/2013	138.881	169.069	911.315344	0.217365946
2/19/2013	152.082	149.378	7.311616	0.017779882
2/20/2013	88.029	158.324	4941.387025	0.798543662
2/21/2013	178.357	145.198	1099.519281	0.185913645
2/22/2013	176.959	159.753	296.046436	0.097231562
2/23/2013	244.465	174.311	4921.583716	0.286969505
2/24/2013	145.498	176.625	968.890129	0.213934212
2/25/2013	153.308	158.324	25.160256	0.032718449
2/26/2013	190.242	165.522	611.0784	0.129939761
2/27/2013	169.463	177.971	72.386064	0.05020565
2/28/2013	186.015	169.069	287.166916	0.09110018
			MSE = 1567.48	MAPE = 0.239863

Figure 4.6 shows the plot of the Cheng (2008) forecasts for the test data set against time.

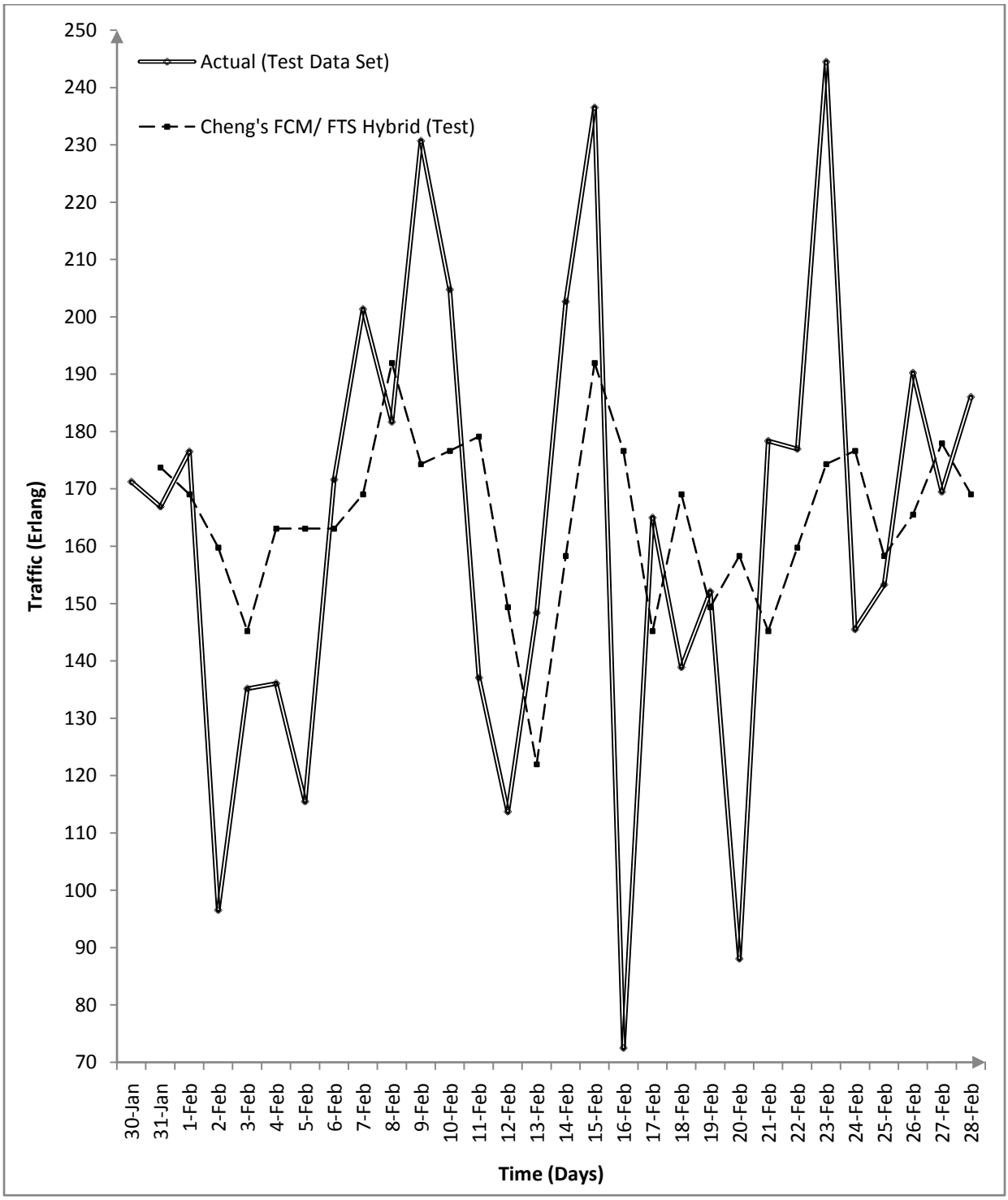


Figure 4.6: Plot of Actual (Test Set) and Forecasted Erlang (Cheng's 2008 Hybrid Model) against Time

4.5 SIGNIFICANCE OF FORECASTING RESULTS

Based on the validation of the proposed hybrid model with the two statistical performance measures of mean square error (MSE) and mean absolute percentage error (MAPE) and its comparison with the Chen's (1996) $(MSE = 845.122, \quad MAPE = 13.47\%)$ and Cheng (2008) $(MSE = 856.145, \quad MAPE = 13.37\%)$ models, the proposed hybrid model has a higher forecasting accuracy $(MSE = 0.9867, \quad MAPE = 0.47\%)$ in predicting Airtel voice (Erlang) taffic, Abuja region. This is because the proposed hybrid has the lowest square loss and absolute loss. In terms of mismatch, the proposed hybrid model has the lowest mismatch between the actual traffic and the forecasted traffic. This is clearly noticed when its plots (Figures 4.1 and 4.3) are compared with the Chen's (1996) (Figure 4.4) and Cheng's (2008) (Figure 4.5) plots.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1SUMMARY

In this research, ahybrid model that integrates fuzzy C-means (FCM) clustering and particle swarm optimization (PSO) into fuzzy time series (FTS) forecasting model to improve forecasting accuracy was developed. The developed model was then applied to forecast voice (Erlang) traffic of Airtel Call Centre, Abuja region. Fuzzy c-means (FCM) clustering algorithm was coded in c# and applied to generate fourteen unequal partitions (clusters centres) for data obtained from Airtel, Abuja.Based on the unequal partitions and membership degrees (partition matrix) obtained, a total of fifty – nine fuzzy "if" – rules were generated. By utilizing particle swarm optimization (PSO) coded in c#, weights of forecasting rules were tuned to match the future data they represent.Finally, the forecasts were obtained and forecasting perfomances of the proposed model was compared with that of Chen's (1996) and Cheng *et al* (2008) models.

5.2CONCLUSION

It has been observed that objectively partitioning the universe of discourse and optimizing the defuzzification process affects forecasting accuracy. The research has developed a novel fuzzy time series forecasting model which considered eliminating the need to define the universe of discourse, learning memberships in hidden data structures, partitioning the universe of discourse objectively, optimizing the defuzzification process, and reducing cost of computation. Fuzzy C-means (FCM) clustering algorithm was developed to objectively partition the universe of discourse and learn memberships in Erlang data obtained from Airtel, Abuja call centre. To obtain unique fuzzy

relations, fuzzy set groups (FSGs) were generated for the Erlang data and converted to 'if – then' forecasting rule. Then, particle swarm optimization algorithm was developed to assign optimal weights to the content of a forecasting rule. Thus, the defuzzification process was optimized. To reduce cost of computation, GUIs of fuzzy C-means clustering and particle swarm optimization was developed using c#. A defuzzification operator, which is a function of tuned weights and historical fuzzy values, was introduced to allow for the training and testing of the proposed hybrid model. By comparing the proposed hybrid model's forecasting results with that of Chen's (1996) and Cheng's (2008) forecasting models, using MSE and MAPE criteria, it was observed that the proposed forecasting model provides more accurate forecasts.

5.3 LIMITATIONS

The following were the limitations of the research work:

1. Fuzzy C-means (FCM) clustering is inherently dependent on the choice of fuzzy index, β, and cluster centre initialization. In this work, fuzzy index, β, was chosen to be a value of 2 to for simplicity. Thus, the results of clustering will change when these parameters changes.

2. The choice of the most suitable cluster validity index was not empirical. Thus, the most appropriate validity index may result to another optimal number of clusters.

5.4RECOMMENDATIONS FOR FURTHER WORKS

Further works should consider the following areas:

1) More comparisons should be made with other existing models to further validate the proposed hybrid model.
2) An optimal means of determining fuzzy index should be investigated.

3) Also, other fuzzy clustering techniques should be applied and compared with fuzzy C-means (FCM) clustering to determine the most suitable clustering technique for clustering voice (Erlang) traffic.

REFERENCES

Aladag C. H, Yolcu U., Egrioglu E. and Dalar A. Z, (2012), "A new time – invariant fuzzy time series forecasting method based on particle swarm optimization", Applied Soft Computing, vol. 12, pp. 3291 – 3299.

Bezdek J. C., (1981), "Pattern recognition with fuzzy objective function algorithms", New York, Plenum.

Birek L., Petrovic D. and Boylan J., (2014), "Water leakage forecasting: the application of a modified fuzzy evolving algorithm", Applied Soft Computing, vol. 14, pp. 305 – 315.

Chen S. M., (1996), "Forecasting enrolments based on fuzzy time series", Fuzzy Sets and Systems, vol. 81, pp. 311 – 319.

Chen S. M., (2002), "Forecasting enrolments based on higher order fuzzy time series", Cybernetics and Systems, vol. 33, pp. 1 – 16.

Cheng C. H., Cheng G. W. and Wang J. W., (2008), "Multi-attribute fuzzy time series based on fuzzy clustering", Expert Systems with Applications, vol. 34, pp. 1235 – 1242.

Chromy E., Misuth T. and Kavacky M., (2011), "Erlang C formula and its use in call centres", Information and Communication Technologies and Services, vol. 9, no. 1, pp. 7 – 13.

Dunn J. C. (1974), "A fuzzy relative of ISODATA process and its use in detecting compact well – separated clusters", Cybernetics, vol. 3, pp. 32 – 57.

Elbeltagi E., Hegazy T. and Grierson D., (2005), "Comparison among evolutionary – based optimization algorithms", Advance Engineering Informatics, vol. 19, pp. 43 – 53.

Eleruja S. A., (2012), "Forecasting GSM traffic using fuzzy time series (TFS): A case study of mobile telephone network (MTN) voice traffic", Thesis, http://kubanni.abu.edu.ng, (22/6/2013).

Eleruja S. A., Mu'azu M. B. and Dajab D. D., (2012), "Application of trapezoidal fuzzification approach (TFA) and particle swarm optimization (PSO) in fuzzy time series (FTS)

forecasting", proceedings of the International Conference on Artificial Intelligence, vol. 1, pp. 80 – 89.

Hao C. (2010), "Comparative study of C, C++, C# and Java programming languages", Thesis, http://publications.theseus.fi/bitstream/handle/10024/16995/Chen_Hao.pdf, (22/5/2014).

Huarng K., (2001), "Effective lengths of intervals to improve fuzzy time series forecasting", Fuzzy Sets and Systems, vol. 123, pp. 387 – 394.

Huarng K. and Yu T. H. K., (2006). "Ratio – based lengths of intervals to improve fuzzy time series forecasting", IEEE Transactions on Systems, Cybernetics, vol.36, pp. 328 – 340.

Jafar O. A. M and Sivakumar R., (2013), "A comparative study of hard and fuzzy data clustering algorithms with cluster validity indices", Proceedings of International Conference on Emerging Research in Computing, Information, Communication and Applications, pp. 775 – 782.

Kennedy J. and Eberhart R., (1995), "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Network, pp. 1942 – 1948.

Kuo I. H., Horng S. J., Kao T. W., Lin T. L., Lee C. L. and Pan Y., (2009), "An improved method for forecasting enrolments based on fuzzy time series and particle swarm optimization", Expert Systems with Applications, vol. 36, pp. 6108 – 6117.

Li S. T., Cheng Y. C. and Lin S. Y., (2008), "A FCM – based deterministic forecasting model for fuzzy time series", Computer and Mathematics with Applications, vol. 56, pp. 3052 – 3063.

Magaret R., (2007), "Erlang", http://www.whatis.com.htm.

Mu'azu M. B. and Adeola K. O., (2008), "The effect of interval length and model bases on fuzzy time series electric load forecasting", NETec, vol. 1, pp. 382 – 388.

Nikhil R. P., Kuhu P., Keller J. M., and Bezdek J. C., (2005), "A possibilistic fuzzy c-means clustering algorithm", IEEE transactions on Fuzzy Systems, vol. 13, No. 4, pp. 517 – 530.

Poulsen J. R., (2009), "Fuzzy time series forecasting – Developing a new forecasting model based on high order fuzzy time series", Thesis, http://projekter.aau.dk/projekter/files/18603950/FTS_rapport_pdf.pdf, (8/6/2013).

Schildt H., (2009), "C# 3.0: A beginners guide", McGraw – Hill

Shcherbakov M. V., Brebels A., Shcherbakova N. L., Tyukov A. P., Janovsky T. A., and Kamaev V. A., (2013), "A survey of forecasting error measures", World Applied Sciences, vol. 24, pp. 171 – 176.

Song Q. and Chissom B. S., (1993), "Forecasting enrolments with fuzzy time series – part I", Fuzzy Sets and Systems, vol. 54, pp. 1 – 9.

Tamhane, A. C. and Dunlop, D. D., (2000). *Statistics and data analysis : from elementary to intermediate*. Prentice Hall ; London : Prentice-Hall International (UK), Upper Saddle River, N.J.

Wang W. and Zhang Y., (2007) "On fuzzy cluster validity indices", Fuzzy Sets and Systems, vol. 158, pp. 2095 – 2117.

Woschnagg E. and Cipan J., (2004) "Evaluating forecast accuracy" http://homepage.univie.ac.at/robert.kunst/procip.pdf, (14/ 10/ 2014).

Xu S., Hu L., Yang X. and Liu X., (2013) "A cluster Number adaptive fuzzy c-means algorithm for image segmentation", International Journal of Image Processing and Pattern Recognition, vol. 6, pp. 191 – 204.

Yolcu U., Egrioglu E., Uslu V. R., Basran M. and Aladag C. H., (2009), "A new approach for determining the length of intervals for fuzzy time series", Applied Soft Computing, vol. 9, pp. 1 – 9.

Zadeh L. A., (1965), "The concept of linguistic variables and its application to approximate reasoning, part 1", Information Science, vol. 8, pp. 199 – 249.

APPENDIX A

POSTPAID AND PREPAID CALLS, ABUJA CALL CENTER

Date	Answered By Agents	AHT	ASA	Average Talk Time	Average Hold Time	Average Waiting Time	Maximum Wait Time	Service Level	Calls Offered	Erlang Traffic
1-Dec-2012	74925	0:02:10	0:00:57	0:02:10	0:00:05	0:00:43	0:59:56	0.6723	113139	170.232
2-Dec-2012	54501	0:02:07	0:00:45	0:02:17	0:00:04	0:00:33	1:50:08	61.61	109084	160.343
3-Dec-2012	82973	0:02:06	0:00:36	0:02:07	0:00:04	0:00:28	0:59:10	0.7508	103519	150.965
4-Dec-2012	79481	0:02:11	0:00:36	0:02:11	0:00:03	0:00:26	0:31:55	0.8003	107605	163.151
5-Dec-2012	71176	0:02:18	0:01:09	0:02:18	0:00:04	0:00:49	1:00:07	0.6951	112608	179.86
6-Dec-2012	88508	0:02:20	0:01:09	0:01:43	0:00:05	0:00:36	0:59:29	0.6185	108942	176.526
7-Dec-2012	72535	0:02:15	0:00:44	0:02:15	0:00:04	0:00:37	0:59:33	0.6992	99552	155.55
8-Dec-2012	49611	0:02:14	0:01:03	0:02:34	0:00:05	0:01:06	0:30:03	0.5341	99664	154.571
9-Dec-2012	69284	0:02:14	0:00:51	0:01:58	0:00:05	0:00:31	0:52:27	0.7037	103728	160.874
10-Dec-2012	67375	0:02:11	0:00:57	0:02:11	0:00:05	0:00:38	0:59:58	0.7075	98087	148.72
11-Dec-2012	77031	0:02:11	0:00:42	0:01:53	0:00:04	0:00:29	0:29:13	0.7775	91361	138.522
12-Dec-2012	75958	0:02:05	0:00:38	0:02:05	0:00:04	0:00:30	0:55:29	0.7203	110019	159.171
13-Dec-2012	70544	0:02:14	0:00:47	0:02:14	0:00:04	0:00:34	1:00:00	0.7461	109204	169.367
14-Dec-2012	51904	0:02:10	0:00:43	0:02:30	0:00:05	0:00:43	0:59:54	0.4357	103816	156.205
15-Dec-2012	46110	0:02:12	0:01:05	0:02:28	0:00:06	0:00:46	0:24:24	0.4692	99855	152.556
16-Dec-2012	59647	0:02:13	0:01:20	0:02:13	0:00:05	0:00:58	1:00:07	0.6024	114621	176.442
17-Dec-2012	83299	0:02:11	0:01:09	0:02:11	0:00:05	0:00:46	0:59:47	0.654	138251	209.617
18-Dec-2012	64364	0:02:12	0:01:16	0:02:12	0:00:04	0:00:47	1:39:25	0.6798	107347	164.002
19-Dec-2012	89850	0:02:10	0:00:51	0:02:00	0:00:05	0:00:36	0:59:47	0.7472	111737	168.123
20-Dec-2012	61921	0:02:12	0:00:58	0:02:32	0:00:05	0:01:20	1:00:04	0.6313	124231	189.797
21-Dec-2012	74041	0:02:17	0:01:17	0:02:17	0:00:06	0:00:46	1:00:07	0.7573	127655	202.416
22-Dec-2012	69814	0:02:11	0:00:54	0:02:03	0:00:06	0:00:31	0:40:58	0.7832	86900	131.758
23-Dec-2012	63084	0:02:14	0:01:21	0:02:14	0:00:06	0:00:23	0:50:47	0.7582	107347	166.487
24-Dec-2012	65639	0:02:14	0:01:18	0:02:14	0:00:06	0:01:00	1:00:08	0.6944	130383	202.214
25-Dec-2012	75038	0:02:15	0:01:51	0:02:15	0:00:06	0:01:08	1:02:19	0.6035	140467	219.48
26-Dec-2012	74673	0:02:21	0:01:52	0:02:21	0:00:08	0:01:13	1:04:24	0.5419	159149	259.722
27-Dec-2012	76897	0:02:19	0:01:41	0:02:19	0:00:07	0:01:07	1:00:13	0.4953	181071	291.306
28-Dec-2012	80406	0:02:20	0:02:09	0:02:20	0:00:08	0:01:27	1:00:10	0.6264	185662	300.841
29-Dec-2012	69857	0:02:18	0:02:14	0:02:18	0:00:08	0:01:22	1:00:09	0.5155	193202	308.587
30-Dec-2012	75065	0:02:19	0:02:25	0:02:19	0:00:08	0:01:22	1:01:57	0.5661	181699	292.317

POSTPAID AND PREPAID CALLS, ABUJA CALL CENTER

Date	Answered By Agents	AHT	ASA	Average Talk Time	Average Hold Time	Average Waiting Time	Maximum Wait Time	Service Level	Calls Offered	Erlang Traffic
1-Jan-2013	107778	0:02:07	0:00:45	0:01:58	0:00:04	0:00:53	0:55:29	0.4625	214765	315.685
2-Jan-2013	105360	0:02:06	0:00:36	0:02:03	0:00:04	0:01:05	1:00:00	0.6347	200134	291.862
3-Jan-2013	78654	0:02:11	0:00:36	0:02:47	0:00:03	0:00:48	0:59:54	0.5188	139765	211.912
4-Jan-2013	89675	0:02:18	0:01:09	0:01:44	0:00:04	0:00:44	0:24:24	0.53	109765	175.319
5-Jan-2013	50117	0:02:20	0:01:09	0:01:50	0:00:05	0:00:46	1:00:07	0.6613	57889	93.8016
6-Jan-2013	53000	0:02:15	0:00:44	0:02:10	0:00:04	0:00:12	0:39:47	0.96	57804	90.3188
7-Jan-2013	80978	0:02:14	0:01:03	0:02:13	0:00:06	0:00:25	0:49:25	0.9	98776	153.194
8-Jan-2013	98734	0:02:17	0:01:17	0:02:10	0:00:06	0:00:20	0:59:47	0.7533	115677	183.423
9-Jan-2013	67545	0:02:11	0:00:54	0:02:04	0:00:08	0:00:31	1:00:04	0.5633	124576	188.883
10-Jan-2013	60987	0:02:14	0:01:21	0:02:24	0:00:07	0:00:16	0:40:07	0.7143	86432	134.05
11-Jan-2013	48968	0:02:14	0:01:18	0:01:52	0:00:08	0:00:41	0:40:58	0.7253	89909	139.442
12-Jan-2013	99103	0:02:15	0:01:51	0:02:04	0:00:08	0:00:35	0:50:47	0.7368	120009	187.514
13-Jan-2013	78996	0:02:21	0:01:52	0:02:28	0:00:08	0:00:26	1:00:08	0.6923	135789	221.6
14-Jan-2013	56890	0:02:19	0:01:41	0:02:10	0:00:05	0:01:09	0:32:19	0.7294	144546	232.545
15-Jan-2013	98765	0:02:20	0:02:09	0:02:45	0:00:05	0:00:40	1:04:24	0.6842	160987	260.859
16-Jan-2013	129676	0:02:18	0:02:14	0:02:21	0:00:05	0:00:34	1:00:13	0.875	199877	319.248
17-Jan-2013	79619	0:02:19	0:02:25	0:01:56	0:00:04	0:00:24	1:00:10	0.8247	98545	158.539
18-Jan-2013	89724	0:02:15	0:00:44	0:02:13	0:00:04	0:00:33	0:45:09	0.6818	117808	184.075
19-Jan-2013	70008	0:02:14	0:01:03	0:03:04	0:00:04	0:01:18	1:01:57	0.3667	98942	153.452
20-Jan-2013	26754	0:02:14	0:00:51	0:02:28	0:00:05	0:01:08	0:59:56	0.4619	101552	157.5
21-Jan-2013	27885	0:02:11	0:00:57	0:02:36	0:00:06	0:00:55	1:50:08	0.6571	99252	150.486
22-Jan-2013	68734	0:02:11	0:00:42	0:01:10	0:00:05	0:00:40	0:49:10	0.7298	123823	187.741
23-Jan-2013	79861	0:02:05	0:00:38	0:01:46	0:00:05	0:00:30	0:31:55	0.9286	98087	141.908
24-Jan-2013	45969	0:02:14	0:00:47	0:02:01	0:00:04	0:01:04	1:00:07	0.5726	92433	143.357
25-Jan-2013	90245	0:02:10	0:00:43	0:02:00	0:00:05	0:00:33	0:59:29	0.9021	110910	166.878
26-Jan-2013	71508	0:02:12	0:01:05	0:02:16	0:00:05	0:01:05	0:59:33	0.8583	108942	166.439
27-Jan-2013	76949	0:02:13	0:01:20	0:01:54	0:00:06	0:00:28	0:30:03	0.8667	103816	159.809
28-Jan-2013	80991	0:02:11	0:01:09	0:02:03	0:00:06	0:00:18	0:52:27	0.8333	99855	151.401
29-Jan-2013	66823	0:02:12	0:01:16	0:02:00	0:00:08	0:00:35	0:59:58	0.8353	125671	191.997
30-Jan-2013	69383	0:02:10	0:00:51	0:01:56	0:00:08	0:00:17	0:29:13	0.6471	113825	171.264
31-Jan-2013	89685	0:02:12	0:00:58	0:01:55	0:00:05	0:00:48	0:55:29	0.808	109237	166.89

POSTPAID AND PREPAID CALLS, ABUJA CALL CENTER

Date	Answered By Agents	AHT	ASA	Average Talk Time	Average Hold Time	Average Waiting Time	Maximum Wait Time	Service Level	Calls Offered	Erlang Traffic
1-Feb-2013	64678	0:02:17	0:01:17	0:02:34	0:00:06	0:00:23	1:00:00	75.6	111325	176.522
2-Feb-2013	88336	0:01:18	0:00:57	0:01:59	0:00:05	0:00:19	0:59:54	86.67	106894	96.5015
3-Feb-2013	54366	0:01:51	0:00:42	0:02:35	0:00:05	0:00:46	0:24:24	60.1	105195	135.146
4-Feb-2013	56864	0:01:52	0:00:38	0:02:20	0:00:04	0:00:38	0:40:07	75	104966	136.067
5-Feb-2013	49205	0:01:41	0:00:47	0:02:18	0:00:05	0:00:54	0:59:29	57.33	98785	115.478
6-Feb-2013	85662	0:02:09	0:00:43	0:02:00	0:00:05	0:00:28	0:59:33	78.57	114923	171.586
7-Feb-2013	65903	0:02:14	0:02:14	0:02:23	0:00:06	0:00:44	0:30:03	53.85	129827	201.352
8-Feb-2013	54834	0:02:25	0:02:25	0:02:41	0:00:06	0:00:51	0:52:27	50	108249	181.668
9-Feb-2013	106596	0:02:14	0:00:44	0:02:32	0:00:06	0:01:03	0:59:58	50	148736	230.679
10-Feb-2013	79656	0:01:51	0:01:03	0:02:03	0:00:06	0:00:56	0:59:47	55.56	159376	204.754
11-Feb-2013	96876	0:01:52	0:00:51	0:01:58	0:00:06	0:00:44	1:00:04	85.71	105738	137.068
12-Feb-2013	86567	0:01:41	0:00:57	0:01:54	0:00:08	0:00:24	0:40:07	91.09	97255	113.689
13-Feb-2013	65655	0:02:09	0:00:42	0:02:15	0:00:07	0:00:45	0:40:58	69.65	99388	148.392
14-Feb-2013	43657	0:02:14	0:00:38	0:02:47	0:00:07	0:01:37	0:50:47	41.38	130686	202.684
15-Feb-2013	76543	0:02:25	0:00:47	0:02:30	0:00:08	0:00:17	1:00:08	77.3	140923	236.503
16-Feb-2013	115893	0:00:44	0:00:43	0:02:16	0:00:08	0:00:58	0:59:33	77.07	142355	72.4956
17-Feb-2013	56834	0:02:15	0:02:14	0:01:59	0:00:08	0:00:16	0:30:03	46.26	105586	164.978
18-Feb-2013	80543	0:02:14	0:02:25	0:02:08	0:00:05	0:00:12	0:52:27	70.9	89547	138.881
19-Feb-2013	55738	0:02:14	0:00:44	0:02:12	0:00:05	0:00:18	0:59:58	78.98	98059	152.082
20-Feb-2013	53987	0:02:11	0:01:03	0:02:00	0:00:05	0:00:30	0:59:47	87.65	58059	88.0293
21-Feb-2013	100056	0:02:11	0:00:51	0:02:05	0:00:04	0:00:27	1:39:25	87.36	117634	178.357
22-Feb-2013	98158	0:02:05	0:00:57	0:02:13	0:00:04	0:00:50	0:59:47	87.46	122314	176.959
23-Feb-2013	97677	0:02:14	0:00:42	0:01:53	0:00:04	0:00:34	1:00:04	74.12	157625	244.465
24-Feb-2013	81343	0:02:10	0:00:38	0:02:08	0:00:05	0:00:19	0:59:33	83.67	96700	145.498
25-Feb-2013	50152	0:02:12	0:00:47	0:02:24	0:00:06	0:00:49	0:30:03	49.09	100347	153.308

APPENDIX B

Cluster Validity Calculation Using MSE_c

Training Data Set	Centre c = 7	SE_7	Centre c = 8	SE_8	Centre c =9	SE_9	Centre c = 10	SE_{10}	Centre c = 11	SE_{11}	Centre c = 12	SE_{12}	Centre c = 13	SE_{13}	Centre c = 14	SE_{14}
90.319	93.556	10.4782	92.173	3.4373	92.097	3.16128	92.086	3.1223	92.142	3.32333	92.084	3.11523	92.084	3.11523	92.071	3.0695
93.802	93.556	0.06052	92.173	2.6536	92.097	2.90703	92.086	2.9447	92.142	2.7556	92.084	2.95152	92.084	2.95152	92.071	2.9964
131.758	152.72	439.405	140.789	81.559	138.342	43.3491	138.271	42.419	140.099	69.5723	138.266	42.3541	138.257	42.237	137.713	35.462
134.05	152.72	348.569	140.789	45.414	138.342	18.4213	138.271	17.817	140.099	36.5904	138.266	17.7747	138.257	17.6988	137.713	13.418
138.522	152.72	201.583	140.789	5.1393	138.342	0.0324	138.271	0.063	140.099	2.48693	138.266	0.06554	138.257	0.07022	137.713	0.6545
139.442	152.72	176.305	140.789	1.8144	138.342	1.21	138.271	1.3712	140.099	0.43165	138.266	1.38298	138.257	1.40423	137.713	2.9894
141.908	152.72	116.899	140.789	1.2522	138.342	12.7164	138.271	13.228	140.099	3.27248	138.266	13.2642	138.257	13.3298	137.713	17.598
143.357	152.72	87.6658	140.789	6.5946	138.342	25.1502	138.271	25.867	140.099	10.6146	138.266	25.9183	138.257	26.01	137.713	31.855
148.72	152.72	16	140.789	62.901	154.859	37.6873	154.612	34.716	140.099	74.3216	154.563	34.1406	154.505	33.4662	151.88	9.9856
150.486	152.72	4.99076	159.97	89.946	154.859	19.1231	154.612	17.024	159.102	74.2355	154.563	16.6219	154.505	16.1524	151.88	1.9432
150.965	152.72	3.08002	159.97	81.09	154.859	15.1632	154.612	13.301	159.102	66.2108	154.563	12.9456	154.505	12.5316	151.88	0.8372
151.401	152.72	1.73976	159.97	73.428	154.859	11.9578	154.612	10.311	159.102	59.3054	154.563	9.99824	154.505	9.63482	151.88	0.2294
152.556	152.72	0.0269	159.97	54.967	154.859	5.30381	154.612	4.2271	159.102	42.8501	154.563	4.02805	154.505	3.7986	151.88	0.457
153.194	152.72	0.22468	159.97	45.914	154.859	2.77223	154.612	2.0107	159.102	34.9045	154.563	1.87416	154.505	1.71872	151.88	1.7266
153.452	152.72	0.53582	159.97	42.484	154.859	1.97965	154.612	1.3456	159.102	31.9225	154.563	1.23432	154.505	1.10881	151.88	2.4712
154.571	152.72	3.4262	159.97	29.149	154.859	0.08294	154.612	0.0017	159.102	20.53	154.563	6.4E-05	154.505	0.00436	151.88	7.2415
155.55	152.72	8.0089	159.97	19.536	154.859	0.47748	154.612	0.8798	159.102	12.6167	154.563	0.97417	154.505	1.09203	159.144	12.917
156.205	152.72	12.1452	159.97	14.175	154.859	1.81172	154.612	2.5376	159.102	8.39261	154.563	2.69616	154.505	2.89	159.144	8.6377
157.5	152.72	22.8484	159.97	6.1009	154.859	6.97488	154.612	8.3405	159.102	2.5664	154.563	8.62597	154.505	8.97003	159.144	2.7027
158.539	152.72	33.8608	159.97	2.0478	154.859	13.5424	154.612	15.421	159.102	0.31697	154.563	15.8086	154.505	16.2732	159.144	0.366
159.171	152.72	41.6154	159.97	0.6384	154.859	18.5933	154.612	20.784	159.102	0.00476	154.563	21.2337	154.505	21.7716	159.144	0.0007
159.809	152.72	50.2539	159.97	0.0259	154.859	24.5025	154.612	27.009	159.102	0.49985	154.563	27.5205	154.505	28.1324	159.144	0.4422
160.343	152.72	58.1101	159.97	0.1391	154.859	30.0743	154.612	32.844	159.102	1.54008	154.563	33.4084	154.505	34.0822	159.144	1.4376
160.874	152.72	66.4877	159.97	0.8172	154.859	36.1802	154.612	39.213	159.102	3.13998	154.563	39.8287	154.505	40.5642	159.144	2.9929
163.151	152.72	108.806	159.97	10.119	168.18	25.2908	167.523	19.114	159.102	16.3944	167.397	18.0285	167.261	16.8921	159.144	16.056
164.002	152.72	127.284	159.97	16.257	168.18	17.4557	167.523	12.397	159.102	24.01	167.397	11.526	167.261	10.6211	167.28	10.745
166.439	178.893	155.102	159.97	41.848	168.18	3.03108	167.523	1.1751	159.102	53.8316	167.397	0.91776	167.261	0.67568	167.28	0.7073
166.487	178.893	153.909	159.97	42.471	168.18	2.86625	167.523	1.0733	159.102	54.5382	167.397	0.8281	167.261	0.59908	167.28	0.6288
166.878	178.893	144.36	159.97	47.72	168.18	1.6952	167.523	0.416	159.102	60.4662	167.397	0.26936	167.261	0.14669	167.28	0.1616
168.123	178.893	115.993	159.97	66.471	168.18	0.00325	167.523	0.36	159.102	81.3784	167.397	0.52708	167.261	0.74304	167.28	0.7106
169.367	178.893	90.7447	159.97	88.304	168.18	1.40897	167.523	3.4003	159.102	105.37	167.397	3.8809	167.261	4.43524	167.28	4.3556
170.232	178.893	75.0129	159.97	105.31	168.18	4.2107	167.523	7.3387	159.102	123.877	167.397	8.03723	167.261	8.82684	167.28	8.7143
175.319	178.893	12.7735	184.568	85.544	168.18	50.9653	167.523	60.778	182.55	52.2874	167.397	62.7581	167.261	64.9314	177.047	2.986
176.442	178.893	6.0074	184.568	66.032	168.18	68.2606	167.523	79.549	182.55	37.3077	167.397	81.812	167.261	84.2908	177.047	0.366
176.526	178.893	5.60269	184.568	64.674	168.18	69.6557	167.523	81.054	182.55	36.2886	167.397	83.3386	167.261	85.8402	177.047	0.2714
179.86	178.893	0.93509	184.568	22.165	187.508	58.4919	186.281	41.229	182.55	7.2361	186.141	39.451	185.899	36.4695	177.047	7.913
183.423	178.893	20.5209	184.568	1.311	187.508	16.6872	186.281	8.1682	182.55	0.76213	186.141	7.38752	185.899	6.13058	188.182	22.648
184.075	178.893	26.8531	184.568	0.243	187.508	11.7855	186.281	4.8664	182.55	2.32562	186.141	4.26836	185.899	3.32698	188.182	16.867
187.514	178.893	74.3216	184.568	8.6789	187.508	3.6E-05	186.281	1.5203	182.55	24.6413	186.141	1.88513	185.899	2.60823	188.182	0.4462
187.741	178.893	78.2871	184.568	10.068	187.508	0.05429	186.281	2.1316	182.55	26.9465	186.141	2.56	185.899	3.39296	188.182	0.1945
188.883	178.893	99.8001	184.568	18.619	187.508	1.89063	186.281	6.7704	182.55	40.1069	186.141	7.51856	185.899	8.90426	188.182	0.4914
189.797	178.893	118.897	184.568	27.342	187.508	5.23952	186.281	12.362	182.55	52.519	186.141	13.3663	185.899	15.1944	188.182	2.6082
191.997	178.893	171.715	184.568	55.19	187.508	20.1511	186.281	32.673	182.55	89.2458	186.141	34.2927	185.899	37.1856	188.182	14.554
202.214	211.843	92.7176	214.275	145.47	215.613	179.533	205.817	12.982	204.158	3.77914	203.882	2.78222	201.907	0.09425	202.234	0.0004
202.416	211.843	88.8683	214.275	140.64	215.613	174.161	205.817	11.567	204.158	3.03456	203.882	2.14916	201.907	0.25908	202.234	0.0331
209.617	211.843	4.95508	214.275	21.697	215.613	35.952	205.817	14.44	204.158	29.8007	203.882	32.8902	210.967	1.8225	210.748	1.2792
211.912	211.843	0.00476	214.275	5.5838	215.613	13.6974	205.817	37.149	204.158	60.1245	219.273	54.1843	210.967	0.89302	210.748	1.3549
219.48	211.843	58.3238	214.275	27.092	215.613	14.9537	223.058	12.802	222.345	8.20823	219.273	0.04285	223.22	13.9876	220.536	1.1151
221.6	211.843	95.199	214.275	53.656	215.613	35.8442	223.058	2.1258	222.345	0.55503	219.273	5.41493	223.22	2.6244	220.536	1.1321
232.545	211.843	428.573	214.275	333.79	215.613	286.693	223.058	90.003	222.345	104.04	232.367	0.03168	223.22	86.9556	232.547	4E-06
259.722	258.827	0.80102	259.383	0.1149	259.622	0.01	260.201	0.2294	260.131	0.16728	260.281	0.31248	260.209	0.23717	260.314	0.3505
260.859	258.827	4.12902	259.383	2.1786	259.622	1.53017	260.201	0.433	260.131	0.52998	260.281	0.33408	260.209	0.4225	260.314	0.297
291.306	294.641	11.1222	294.701	11.526	294.717	11.6349	294.749	11.854	291.813	0.25705	291.864	0.31136	291.822	0.26626	294.705	11.553
291.862	294.641	7.72284	294.701	8.0599	294.717	8.15102	294.749	8.3348	291.813	0.0024	291.864	4E-06	291.822	0.0016	294.705	8.0826
292.317	294.641	5.40098	294.701	5.6835	294.717	5.76	294.749	5.9146	291.813	0.25402	291.864	0.20521	291.822	0.24503	294.705	5.7025
300.841	294.641	38.44	294.701	37.7	294.717	37.5034	294.749	37.112	301.295	0.20612	302.431	2.5281	301.305	0.2153	294.705	37.65
301.779	294.641	50.951	294.701	50.098	294.717	49.8718	294.749	49.421	301.295	0.23426	302.431	0.4251	301.305	0.22468	294.705	50.041
308.587	314.591	36.048	314.663	36.918	314.692	37.271	314.745	37.921	308.599	0.00014	302.431	37.8963	308.611	0.00058	314.779	38.341
315.685	314.591	1.19684	314.663	1.0445	314.692	0.98605	314.745	0.8836	317.544	3.45588	317.01	1.75562	317.554	3.49316	314.779	0.8208
319.248	314.591	21.6876	314.663	21.022	314.692	20.7571	314.745	20.277	317.544	2.90362	317.01	5.00864	317.554	2.86964	314.779	19.972
	MSE_7 = 70.623		MSE_8 = 39.198		MSE_9 = 26.777		MSE_{10} = 17.777		MSE_{11} = 27.82		MSE_{12} = 14.445		MSE_{13} = 14.081		MSE_{14} = 7.526	

APPENDIX C

Training Data Set Partition Matrix (Membership Degrees) for $c = 14$

Date	Traffic	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
1/6/2013	90.319	3.19E-04	7.31E-05	4.06E-04	2.10E-04	0.9949	8.05E-04	1.80E-04	6.06E-05	1.06E-04	1.06E-04	6.44E-04	1.36E-03	2.44E-04	5.15E-04
1/5/2013	93.802	3.35E-04	7.39E-05	4.30E-04	2.18E-04	0.9945	8.84E-04	1.86E-04	6.11E-05	1.08E-04	6.98E-04	1.55E-03	1.55E-04	2.54E-04	5.52E-04
12/22/2012	131.76	8.99E-03	1.08E-03	1.40E-02	4.59E-03	1.82E-02	7.07E-02	3.63E-03	8.54E-04	1.73E-03	3.82E-02	0.8069	2.82E-03	5.76E-03	2.27E-02
1/10/2013	134.05	4.14E-03	4.70E-04	6.57E-03	2.06E-03	6.89E-03	3.82E-02	1.62E-03	3.72E-04	7.62E-04	1.93E-02	0.9048	1.25E-03	2.61E-03	1.10E-02
12/11/2012	138.52	4.06E-16	4.10E-17	6.74E-16	1.92E-16	4.63E-16	5.60E-15	1.49E-16	3.21E-17	6.74E-17	2.35E-15	0.9999	1.13E-16	2.46E-16	1.21E-15
1/11/2013	139.44	1.21E-03	1.19E-04	2.04E-03	5.66E-04	1.28E-03	1.86E-02	4.38E-04	9.36E-05	1.97E-04	7.42E-03	0.9633	3.32E-04	7.30E-04	3.71E-03
1/23/2013	141.91	6.28E-03	5.76E-03	1.09E-02	2.84E-03	5.41E-03	0.1352	2.17E-03	4.50E-04	9.59E-04	4.52E-02	0.7638	1.64E-03	3.69E-03	2.09E-02
1/24/2013	143.36	9.28E-03	8.14E-04	1.64E-02	4.11E-03	7.09E-03	0.2567	3.13E-03	6.34E-04	1.36E-03	7.48E-02	0.5854	2.34E-03	5.38E-03	3.26E-02
12/10/2012	148.72	5.19E-03	3.79E-04	1.01E-02	2.10E-03	2.52E-03	0.8089	1.57E-03	2.93E-04	6.48E-04	7.43E-02	6.67E-02	1.15E-03	2.82E-03	2.34E-02
1/21/2013	150.49	1.30E-03	8.87E-05	2.62E-03	5.09E-04	5.41E-04	0.9509	3.76E-04	6.84E-05	1.53E-04	2.46E-02	1.13E-02	2.74E-04	6.90E-04	6.54E-03
12/3/2012	150.97	7.22E-16	4.84E-17	1.47E-15	2.80E-16	2.88E-16	0.9999	2.07E-16	3.73E-17	8.36E-17	1.49E-14	5.69E-15	1.50E-16	3.80E-16	3.76E-15
1/28/2013	151.4	7.39E-16	4.87E-17	1.52E-15	2.84E-16	2.84E-16	0.9999	2.09E-16	3.75E-17	8.43E-17	1.67E-14	5.34E-15	1.52E-16	3.87E-16	3.97E-15
12/15/2012	152.56	7.88E-16	4.95E-17	1.67E-15	2.95E-16	2.73E-16	0.9999	2.16E-16	3.80E-17	8.61E-17	2.30E-14	4.54E-15	1.56E-16	4.05E-16	4.61E-15
1/7/2013	153.19	1.32E-03	8.05E-05	2.83E-03	4.87E-04	4.31E-04	0.933	3.55E-04	6.17E-05	1.40E-04	4.55E-02	6.73E-03	2.56E-04	6.70E-04	8.12E-03
1/19/2013	153.45	1.85E-03	1.12E-04	4.00E-03	6.79E-04	5.91E-04	0.9013	4.95E-04	8.56E-05	1.95E-04	6.88E-02	8.99E-03	3.56E-04	9.36E-04	1.17E-02
12/8/2012	154.57	4.43E-03	2.55E-04	9.90E-03	1.58E-03	1.28E-03	0.6902	1.15E-03	1.95E-04	4.47E-04	0.2391	1.76E-02	8.22E-04	2.20E-03	3.10E-02
12/7/2012	155.55	5.63E-03	3.10E-04	1.30E-02	1.97E-03	1.49E-03	0.4451	1.42E-03	2.36E-04	5.46E-04	0.4641	1.88E-02	1.01E-03	2.75E-03	4.36E-02
12/14/2012	156.21	5.28E-03	2.81E-04	1.24E-02	1.81E-03	1.31E-03	0.2886	1.30E-03	2.15E-04	4.98E-04	0.625	1.58E-02	9.26E-04	2.55E-03	4.40E-02
1/20/2013	157.5	2.53E-03	1.27E-04	6.23E-03	8.40E-04	5.56E-04	7.54E-02	5.99E-04	9.63E-05	2.25E-04	0.8808	6.08E-03	4.23E-04	1.19E-03	2.49E-02
1/17/2013	158.54	1.14E-15	5.39E-17	2.92E-15	3.67E-16	2.26E-16	2.25E-14	2.60E-16	4.10E-17	9.65E-17	0.9999	2.31E-15	1.83E-16	5.24E-16	1.31E-14
12/12/2012	159.17	1.19E-15	5.44E-17	3.13E-15	3.76E-16	2.22E-16	1.88E-14	2.66E-16	4.13E-17	9.78E-17	0.9999	2.17E-15	1.86E-16	5.39E-16	1.52E-14
1/27/2013	159.81	1.24E-15	5.50E-17	3.37E-15	3.85E-16	2.18E-16	1.69E-14	2.71E-16	4.16E-17	9.90E-17	0.9999	2.05E-15	1.89E-16	5.56E-16	1.79E-14
12/2/2012	160.34	1.75E-03	7.49E-05	4.84E-03	5.32E-04	2.90E-04	1.89E-02	3.73E-04	5.67E-05	1.35E-04	0.9413	2.64E-03	2.59E-04	7.71E-04	2.82E-02
12/9/2012	160.87	3.53E-03	1.47E-04	1.01E-02	1.06E-03	5.56E-04	3.25E-02	7.40E-04	1.11E-04	2.66E-04	0.8799	4.91E-03	5.12E-04	1.54E-03	6.41E-02
12/4/2012	163.15	1.15E-02	4.15E-04	3.72E-02	3.17E-03	1.42E-03	5.66E-02	2.18E-03	3.13E-04	7.61E-04	0.4477	1.11E-02	1.49E-03	4.70E-03	0.4215
12/18/2012	164	1.11E-02	3.82E-04	3.83E-02	2.98E-03	1.26E-03	4.44E-02	2.04E-03	2.87E-04	7.03E-04	0.2764	9.44E-03	1.39E-03	4.46E-03	0.6068
1/26/2013	166.44	2.12E-15	6.08E-17	8.89E-15	5.09E-15	1.81E-16	4.72E-15	3.42E-16	4.54E-17	1.13E-16	1.88E-14	1.21E-15	2.29E-16	7.80E-16	0.9999
12/23/2012	166.49	2.12E-15	6.08E-17	8.97E-15	5.10E-15	1.81E-16	4.69E-15	3.42E-16	4.55E-17	1.14E-16	1.85E-14	1.21E-15	2.29E-16	7.83E-16	0.9999
1/25/2013	166.88	2.20E-15	6.12E-17	9.67E-15	5.19E-16	1.79E-16	4.45E-15	3.47E-16	4.57E-17	1.15E-16	1.67E-14	1.18E-15	2.32E-16	8.00E-16	0.9999
12/19/2012	168.12	2.49E-15	6.24E-17	1.26E-14	5.50E-16	1.73E-16	3.79E-15	3.64E-16	4.65E-17	1.18E-16	1.24E-14	1.08E-15	2.41E-16	8.59E-16	0.9999
12/13/2012	169.37	1.06E-02	2.40E-04	6.38E-02	2.20E-03	6.30E-02	1.23E-02	1.44E-03	1.78E-04	4.55E-04	3.60E-02	3.76E-03	9.43E-04	3.48E-03	0.8639
12/1/2012	170.23	2.01E-02	4.19E-04	0.1397	3.95E-03	1.06E-03	1.93E-02	2.56E-03	3.11E-04	8.00E-04	5.28E-02	6.14E-03	1.67E-03	6.34E-03	0.7448
1/4/2013	175.32	1.65E-02	1.92E-04	0.9147	2.17E-03	3.94E-04	4.97E-03	1.34E-03	1.40E-04	3.78E-04	1.04E-02	1.93E-03	8.34E-04	3.77E-03	4.22E-02
12/16/2012	176.44	7.26E-15	7.15E-17	0.9999	8.50E-16	1.40E-16	1.66E-15	5.14E-16	5.22E-17	1.42E-16	3.34E-15	6.67E-16	3.18E-16	1.50E-15	1.19E-14
12/6/2012	176.53	7.36E-15	7.16E-17	0.9999	8.54E-16	1.40E-16	1.65E-15	5.16E-16	5.23E-17	1.42E-16	3.31E-15	6.64E-16	3.19E-16	1.51E-15	1.17E-14
12/5/2012	179.86	9.27E-02	4.87E-04	0.8115	6.73E-03	8.33E-04	8.20E-03	3.88E-03	3.53E-04	9.92E-04	1.50E-02	3.62E-03	2.31E-03	1.28E-02	4.06E-02
1/8/2013	183.42	0.5418	9.91E-04	0.3018	1.64E-02	1.47E-03	1.23E-02	8.91E-03	7.11E-04	2.08E-03	2.08E-02	5.87E-03	5.08E-03	3.47E-02	4.71E-02
1/18/2013	184.08	0.6433	8.86E-04	0.2196	1.52E-02	1.28E-03	1.05E-02	8.16E-03	6.35E-04	1.87E-03	1.75E-02	5.05E-03	4.62E-03	3.29E-02	3.85E-02
1/12/2013	187.51	0.9999	8.70E-17	9.13E-15	1.85E-15	1.10E-16	7.88E-16	9.17E-16	6.17E-17	1.89E-16	1.22E-15	4.00E-16	4.98E-16	4.76E-15	2.39E-15
1/22/2013	187.74	0.9999	8.74E-17	8.74E-15	1.89E-15	1.09E-16	7.78E-16	9.30E-16	6.20E-17	1.90E-16	1.22E-15	4.00E-16	4.98E-16	4.76E-15	2.39E-15
1/9/2013	188.88	0.9999	8.93E-17	7.14E-15	2.09E-15	1.07E-16	7.30E-16	9.98E-16	6.31E-17	1.96E-16	1.13E-15	3.82E-16	5.25E-16	5.61E-15	2.14E-15
12/20/2012	189.8	0.9479	2.25E-04	1.52E-02	5.64E-03	2.59E-04	1.72E-03	2.62E-03	1.58E-04	4.97E-04	2.63E-03	9.12E-04	1.35E-03	1.60E-02	4.88E-03
1/29/2013	192	0.7517	1.04E-03	4.90E-02	3.11E-02	1.10E-03	6.80E-03	1.34E-02	7.26E-04	2.34E-03	1.01E-02	3.71E-03	6.65E-03	0.1044	1.79E-02
12/24/2012	202.21	5.07E-15	1.17E-16	1.58E-15	1.37E-14	8.24E-17	3.95E-16	2.98E-15	7.89E-17	2.96E-16	5.39E-16	2.40E-16	1.09E-15	0.9999	8.19E-16
12/21/2012	202.42	4.94E-15	1.17E-16	1.55E-15	1.44E-14	8.21E-17	3.92E-16	3.05E-15	7.92E-17	2.98E-16	5.34E-16	2.39E-16	1.10E-15	0.9999	8.10E-16
12/17/2012	209.62	2.67E-03	1.69E-04	1.16E-03	0.9585	8.87E-05	3.68E-04	1.03E-02	1.11E-04	4.77E-04	4.81E-04	2.37E-04	2.33E-03	2.25E-02	6.84E-04
1/3/2013	211.91	2.31E-03	1.90E-04	1.07E-03	0.9595	9.05E-05	3.61E-04	1.75E-02	1.23E-04	5.55E-04	4.67E-04	2.36E-04	3.05E-03	1.39E-02	6.53E-04
12/25/2012	219.48	1.11E-02	1.92E-04	6.02E-04	1.42E-02	6.68E-05	2.37E-04	0.9719	1.19E-04	6.50E-04	2.98E-04	1.62E-04	6.35E-03	3.65E-03	3.98E-04
1/13/2013	221.6	9.87E-04	2.06E-04	5.56E-04	9.36E-03	6.57E-05	2.27E-04	0.9748	1.27E-04	7.36E-04	2.83E-04	1.57E-04	9.20E-03	2.94E-03	3.74E-04
1/14/2013	232.55	5.08E-16	2.59E-16	3.25E-16	2.10E-15	5.07E-17	1.54E-16	6.93E-15	1.48E-16	1.30E-15	1.86E-16	1.11E-16	0.9999	1.09E-15	2.35E-16
12/26/2012	259.72	1.95E-16	8.17E-16	1.46E-16	4.17E-16	3.56E-17	8.60E-17	6.51E-16	3.30E-16	0.9999	9.89E-17	6.72E-17	1.35E-15	3.03E-16	1.17E-16
1/15/2013	260.86	1.89E-16	8.73E-16	1.42E-16	3.98E-16	3.51E-17	8.42E-17	6.15E-16	3.44E-16	0.9999	9.67E-17	6.59E-17	1.25E-15	2.91E-16	1.14E-16
12/27/2012	291.31	1.04E-03	0.9554	8.45E-04	1.70E-03	2.78E-04	5.68E-04	2.20E-03	2.00E-02	1.15E-02	6.32E-04	4.68E-04	3.20E-03	1.39E-03	7.18E-04
1/2/2013	291.86	7.27E-04	0.9681	5.94E-04	1.19E-03	1.96E-04	3.99E-04	1.54E-03	1.49E-02	7.86E-03	4.44E-04	3.29E-04	2.22E-03	9.74E-04	5.04E-04
12/30/2012	292.32	5.14E-04	0.9771	4.19E-04	8.37E-04	1.39E-04	2.82E-04	1.08E-03	1.10E-02	5.44E-03	3.14E-04	2.33E-04	1.56E-03	6.87E-04	3.56E-04
12/28/2012	300.84	2.37E-03	0.7984	1.96E-03	3.70E-03	6.90E-04	1.35E-03	4.66E-03	0.1547	1.83E-02	1.50E-03	1.13E-03	6.44E-03	3.09E-03	1.69E-03
12/31/2012	301.78	2.83E-03	7.29E-01	2.34E-03	4.40E-03	8.29E-04	1.62E-03	5.53E-03	0.2158	2.12E-02	1.79E-03	1.36E-03	7.61E-03	3.68E-03	2.02E-03
12/29/2012	308.59	2.12E-03	0.1596	1.78E-03	3.21E-03	6.56E-04	1.25E-03	3.97E-03	0.8022	1.32E-02	1.38E-03	1.05E-03	5.32E-03	2.72E-03	1.54E-03
1/1/2013	315.69	6.15E-17	2.27E-15	5.20E-17	9.08E-17	2.00E-17	3.72E-17	1.10E-05	0.9999	3.26E-16	4.08E-17	3.16E-17	1.45E-16	7.77E-17	4.54E-17
1/16/2013	319.25	1.10E-03	3.15E-02	9.39E-04	1.61E-03	3.68E-04	6.78E-04	1.95E-03	0.9503	5.46E-03	7.40E-04	5.76E-04	2.52E-03	1.39E-03	8.22E-04

APPENDIX D

Extension Process for obtaining Disambiguated Fuzzy Set Groups (FSGs)

Ambiguous FSGs				Disambiguated FSGs	
LABEL	FSG	Extension Process		LABEL	FSG
1	$\{A_5, A_4\}$	$\{\#, A_5, A_4\}$		1	$\{\#, A_5, A_4\}$
2	$\{A_4, A_3\}$	$\{A_5, A_4, A_3\}$	$\{\#, A_5, A_4, A_3\}$	2	$\{\#, A_5, A_4, A_3\}$
3	$\{A_3, A_4\}$	$\{A_4, A_3, A_4\}$	$\{A_5, A_4, A_3, A_4\}$	3	$\{A_5, A_4, A_3, A_4\}$
4	$\{A_4, A_6\}$			4	$\{A_4, A_6\}$
5	$\{A_6, A_6\}$			5	$\{A_6, A_6\}$
6	$\{A_6, A_4\}$			6	$\{A_6, A_4\}$
7	$\{A_4, A_3\}$	$\{A_6, A_4, A_3\}$		7	$\{A_6, A_4, A_3\}$
8	$\{A_3, A_4\}$	$\{A_4, A_3, A_4\}$	$\{A_6, A_4, A_3, A_4\}$	8	$\{A_6, A_4, A_3, A_4\}$
9	$\{A_4, A_3\}$	$\{A_4, A_4, A_3\}$		9	$\{A_4, A_4, A_3\}$
10	$\{A_3, A_2\}$			10	$\{A_3, A_2\}$
11	$\{A_2, A_4\}$			11	$\{A_2, A_4\}$
12	$\{A_4, A_5\}$			12	$\{A_4, A_5\}$
13	$\{A_5, A_4\}$	$\{A_4, A_5, A_4\}$		13	$\{A_4, A_5, A_4\}$
14	$\{A_4, A_3\}$	$\{A_5, A_4, A_3\}$	$\{A_4, A_5, A_4, A_3\}$	14	$\{A_4, A_5, A_4, A_3\}$
15	$\{A_3, A_6\}$			15	$\{A_3, A_6\}$
16	$\{A_6, A_9\}$			16	$\{A_6, A_9\}$
17	$\{A_9, A_5\}$			17	$\{A_9, A_5\}$
18	$\{A_5, A_5\}$	$\{A_9, A_5, A_5\}$		18	$\{A_9, A_5, A_5\}$
19	$\{A_5, A_7\}$			19	$\{A_5, A_7\}$
20	$\{A_7, A_8\}$			20	$\{A_7, A_8\}$
21	$\{A_8, A_2\}$			21	$\{A_8, A_2\}$
22	$\{A_2, A_5\}$	$\{A_8, A_2, A_5\}$		22	$\{A_8, A_2, A_5\}$
23	$\{A_5, A_8\}$			23	$\{A_5, A_8\}$
24	$\{A_8, A_{10}\}$			24	$\{A_8, A_{10}\}$
25	$\{A_{10}, A_{12}\}$			25	$\{A_{10}, A_{12}\}$
26	$\{A_{12}, A_{13}\}$			26	$\{A_{12}, A_{13}\}$
27	$\{A_{13}, A_{13}\}$	$\{A_{12}, A_{13}, A_{13}\}$		27	$\{A_{12}, A_{13}, A_{13}\}$
28	$\{A_{13}, A_{14}\}$	$\{A_{13}, A_{13}, A_{14}\}$	$\{A_{12}, A_{13}, A_{13}, A_{14}\}$	28	$\{A_{12}, A_{13}, A_{13}, A_{14}\}$
29	$\{A_{14}, A_{13}\}$	$\{A_{13}, A_{14}, A_{13}\}$	$\{A_{13}, A_{13}, A_{14}, A_{13}\}$	29	$\{A_{12}, A_{13}, A_{13}, A_{14}, A_{13}\}$
30	$\{A_{13}, A_{13}\}$	$\{A_{14}, A_{13}, A_{13}\}$		30	$\{A_{14}, A_{13}, A_{13}\}$

Extension Process for obtaining Disambiguated Fuzzy Set Groups (FSGs)

Ambiguous FSGs				Disambiguated FSGs	
LABEL	FSG	Extension Process		LABEL	FSG
31	$\{A_{13}, A_{14}\}$	$\{A_{13}, A_{13}, A_{14}\}$	$\{A_{14}, A_{13}, A_{13}, A_{14}\}$	31	$\{A_{14}, A_{13}, A_{13}, A_{14}\}$
32	$\{A_{14}, A_{13}\}$	$\{A_{13}, A_{14}, A_{13}\}$	$\{A_{13}, A_{13}, A_{14}, A_{13}\}$	32	$\{A_{14}, A_{13}, A_{13}, A_{14}, A_{13}\}$
33	$\{A_{13}, A_9\}$			33	$\{A_{13}, A_9\}$
34	$\{A_9, A_6\}$			34	$\{A_9, A_6\}$
35	$\{A_6, A_1\}$			35	$\{A_6, A_1\}$
36	$\{A_1, A_1\}$			36	$\{A_1, A_1\}$
37	$\{A_1, A_3\}$			37	$\{A_1, A_3\}$
38	$\{A_3, A_7\}$	$\{A_1, A_3, A_7\}$		38	$\{A_1, A_3, A_7\}$
39	$\{A_7, A_7\}$			39	$\{A_7, A_7\}$
40	$\{A_7, A_2\}$	$\{A_7, A_7, A_2\}$		40	$\{A_7, A_7, A_2\}$
41	$\{A_2, A_2\}$	$\{A_7, A_2, A_2\}$	$\{A_7, A_7, A_2, A_2\}$	41	$\{A_7, A_7, A_2, A_2\}$
42	$\{A_2, A_7\}$			42	$\{A_2, A_7\}$
43	$\{A_7, A_{10}\}$			43	$\{A_7, A_{10}\}$
44	$\{A_{10}, A_{11}\}$			44	$\{A_{10}, A_{11}\}$
45	$\{A_{11}, A_{12}\}$			45	$\{A_{11}, A_{12}\}$
46	$\{A_{12}, A_{14}\}$			46	$\{A_{12}, A_{14}\}$
47	$\{A_{14}, A_4\}$			47	$\{A_{14}, A_4\}$
48	$\{A_4, A_7\}$			48	$\{A_4, A_7\}$
49	$\{A_7, A_3\}$			49	$\{A_7, A_3\}$
50	$\{A_3, A_4\}$	$\{A_7, A_3, A_4\}$		50	$\{A_7, A_3, A_4\}$
51	$\{A_4, A_3\}$	$\{A_3, A_4, A_3\}$		51	$\{A_3, A_4, A_3\}$
52	$\{A_3, A_7\}$	$\{A_4, A_3, A_7\}$		52	$\{A_4, A_3, A_7\}$
53	$\{A_7, A_2\}$	$\{A_3, A_7, A_2\}$		53	$\{A_3, A_7, A_2\}$
54	$\{A_2, A_2\}$	$\{A_7, A_2, A_2\}$	$\{A_3, A_7, A_2, A_2\}$	54	$\{A_3, A_7, A_2, A_2\}$
55	$\{A_2, A_5\}$	$\{A_2, A_2, A_5\}$		55	$\{A_2, A_2, A_5\}$
56	$\{A_5, A_5\}$	$\{A_2, A_5, A_5\}$		56	$\{A_2, A_5, A_5\}$
57	$\{A_5, A_5\}$	$\{A_5, A_5, A_5\}$		57	$\{A_5, A_5, A_5\}$
58	$\{A_5, A_3\}$			58	$\{A_5, A_3\}$
59	$\{A_3, A_7\}$	$\{A_5, A_3, A_7\}$		59	$\{A_5, A_3, A_7\}$

APPENDIX E

Calculation of Distances for Test Data Set

		A7	A13	A6	A9	A1	A3	A10	A14
Date	Testing Data Set	$d(x_t; v_1)$	$d(x_t; v_2)$	$d(x_t; v_3)$	$d(x_t; v_4)$	$d(x_t; v_5)$	$d(x_t; v_6)$	$d(x_t; v_7)$	$d(x_t; v_8)$
1/30/2013	171.264	16.918	123.441	5.783	39.484	79.193	19.384	49.272	143.515
1/31/2013	166.89	21.292	127.815	10.157	43.858	74.819	15.01	53.646	147.889
2/1/2013	176.522	11.66	118.183	0.525	34.226	84.451	24.642	44.014	138.257
2/2/2013	96.502	91.68	198.203	80.545	114.246	4.431	55.378	124.034	218.277
2/3/2013	135.146	53.036	159.559	41.901	75.602	43.075	16.734	85.39	179.633
2/4/2013	136.067	52.115	158.638	40.98	74.681	43.996	15.813	84.469	178.712
2/5/2013	115.478	72.704	179.227	61.569	95.27	23.407	36.402	105.058	199.301
2/6/2013	171.586	16.596	123.119	5.461	39.162	79.515	19.706	48.95	143.193
2/7/2013	201.352	13.17	93.353	24.305	9.396	109.281	49.472	19.184	113.427
2/8/2013	181.668	6.514	113.037	4.621	29.08	89.597	29.788	38.868	133.111
2/9/2013	230.679	42.497	64.026	53.632	19.931	138.608	78.799	10.143	84.1
2/10/2013	204.754	16.572	89.951	27.707	5.994	112.683	52.874	15.782	110.025
2/11/2013	137.068	51.114	157.637	39.979	73.68	44.997	14.812	83.468	177.711
2/12/2013	113.689	74.493	181.016	63.358	97.059	21.618	38.191	106.847	201.09
2/13/2013	148.392	39.79	146.313	28.655	62.356	56.321	3.488	72.144	166.387
2/14/2013	202.684	14.502	92.021	25.637	8.064	110.613	50.804	17.852	112.095
2/15/2013	236.503	48.321	58.202	59.456	25.755	144.432	84.623	15.967	78.276
2/16/2013	72.496	115.686	222.209	104.551	138.252	19.575	79.384	148.04	242.283
2/17/2013	164.978	23.204	129.727	12.069	45.77	72.907	13.098	55.558	149.801
2/18/2013	138.881	49.301	155.824	38.166	71.867	46.81	12.999	81.655	175.898
2/19/2013	152.082	36.1	142.623	24.965	58.666	60.011	0.202	68.454	162.697
2/20/2013	88.029	100.153	206.676	89.018	122.719	4.042	63.851	132.507	226.75
2/21/2013	178.357	9.825	116.348	1.31	32.391	86.286	26.477	42.179	136.422
2/22/2013	176.959	11.223	117.746	0.088	33.789	84.888	25.079	43.577	137.82
2/23/2013	244.465	56.283	50.24	67.418	33.717	152.394	92.585	23.929	70.314
2/24/2013	145.498	42.684	149.207	31.549	65.25	53.427	6.382	75.038	169.281
2/25/2013	153.308	34.874	141.397	23.739	57.44	61.237	1.428	67.228	161.471
2/26/2013	190.242	2.06	104.463	13.195	20.506	98.171	38.362	30.294	124.537
2/27/2013	169.463	18.719	125.242	7.584	41.285	77.392	17.583	51.073	145.316
2/28/2013	186.015	2.167	108.69	8.968	24.733	93.944	34.135	34.521	128.764

Calculation of Distances for Test Data Set

		A12	A4	A2	A11	A8	A5
Date	Testing Data Set	$d(x_t; v_9)$	$d(x_t; v_{10})$	$d(x_t; v_{11})$	$d(x_t; v_{12})$	$d(x_t; v_{13})$	$d(x_t; v_{14})$
1/30/2013	171.264	89.05	12.12	33.551	61.283	30.97	3.984
1/31/2013	166.89	93.424	7.746	29.177	65.657	35.344	0.39
2/1/2013	176.522	83.792	17.378	38.809	56.025	25.712	9.242
2/2/2013	96.502	163.812	62.642	41.211	136.045	105.732	70.778
2/3/2013	135.146	125.168	23.998	2.567	97.401	67.088	32.134
2/4/2013	136.067	124.247	23.077	1.646	96.48	66.167	31.213
2/5/2013	115.478	144.836	43.666	22.235	117.069	86.756	51.802
2/6/2013	171.586	88.728	12.442	33.873	60.961	30.648	4.306
2/7/2013	201.352	58.962	42.208	63.639	31.195	0.882	34.072
2/8/2013	181.668	78.646	22.524	43.955	50.879	20.566	14.388
2/9/2013	230.679	29.635	71.535	92.966	1.868	28.445	63.399
2/10/2013	204.754	55.56	45.61	67.041	27.793	2.52	37.474
2/11/2013	137.068	123.246	22.076	0.645	95.479	65.166	30.212
2/12/2013	113.689	146.625	45.455	24.024	118.858	88.545	53.591
2/13/2013	148.392	111.922	10.752	10.679	84.155	53.842	18.888
2/14/2013	202.684	57.63	43.54	64.971	29.863	0.45	35.404
2/15/2013	236.503	23.811	77.359	98.79	3.956	34.269	69.223
2/16/2013	72.496	187.818	86.648	65.217	160.051	129.738	94.784
2/17/2013	164.978	95.336	5.834	27.265	67.569	37.256	2.302
2/18/2013	138.881	121.433	20.263	1.168	93.666	63.353	28.399
2/19/2013	152.082	108.232	7.062	14.369	80.465	50.152	15.198
2/20/2013	88.029	172.285	71.115	49.684	144.518	114.205	79.251
2/21/2013	178.357	81.957	19.213	40.644	54.19	23.877	11.077
2/22/2013	176.959	83.355	17.815	39.246	55.588	25.275	9.679
2/23/2013	244.465	15.849	85.321	106.752	11.918	42.231	77.185
2/24/2013	145.498	114.816	13.646	7.785	87.049	56.736	21.782
2/25/2013	153.308	107.006	5.836	15.595	79.239	48.926	13.972
2/26/2013	190.242	70.072	31.098	52.529	42.305	11.992	22.962
2/27/2013	169.463	90.851	10.319	31.75	63.084	32.771	2.183
2/28/2013	186.015	74.299	26.871	48.302	46.532	16.219	18.735

APPENDIX F

Fuzzified Daily Erlang Training Set Traffic (Chen's Model)

Date	Training Data Set	Fuzzy Set	Date	Training Data Set	Fuzzy Set
12/1/2012	170.232	A_6	12/31/2012	301.779	A_{13}
12/2/2012	160.343	A_5	1/1/2013	315.685	A_{14}
12/3/2012	150.965	A_4	1/2/2013	291.862	A_{12}
12/4/2012	163.151	A_5	1/3/2013	211.912	A_8
12/5/2012	179.86	A_6	1/4/2013	175.319	A_6
12/6/2012	176.526	A_6	1/5/2013	93.802	A_1
12/7/2012	155.55	A_5	1/6/2013	90.319	A_1
12/8/2012	154.571	A_5	1/7/2013	153.194	A_5
12/9/2012	160.874	A_5	1/8/2013	183.423	A_6
12/10/2012	148.72	A_4	1/9/2013	188.883	A_7
12/11/2012	138.522	A_4	1/10/2013	134.05	A_4
12/12/2012	159.171	A_5	1/11/2013	139.442	A_4
12/13/2012	169.367	A_5	1/12/2013	187.514	A_6
12/14/2012	156.205	A_5	1/13/2013	221.6	A_8
12/15/2012	152.556	A_5	1/14/2013	232.545	A_9
12/16/2012	176.442	A_6	1/15/2013	260.859	A_{11}
12/17/2012	209.617	A_8	1/16/2013	319.248	A_{14}
12/18/2012	164.002	A_5	1/17/2013	158.539	A_5
12/19/2012	168.123	A_5	1/18/2013	184.075	A_6
12/20/2012	189.797	A_7	1/19/2013	153.452	A_5
12/21/2012	202.416	A_7	1/20/2013	157.5	A_5
12/22/2012	131.758	A_3	1/21/2013	150.486	A_4
12/23/2012	166.487	A_5	1/22/2013	187.741	A_6
12/24/2012	202.214	A_7	1/23/2013	141.908	A_4
12/25/2012	219.48	A_8	1/24/2013	143.357	A_4
12/26/2012	259.722	A_{10}	1/25/2013	166.878	A_5
12/27/2012	291.306	A_{12}	1/26/2013	166.439	A_5
12/28/2012	300.841	A_{13}	1/27/2013	159.809	A_5
12/29/2012	308.587	A_{13}	1/28/2013	151.401	A_4
12/30/2012	292.317	A_{12}	1/29/2013	191.997	A_7

APPENDIX G

Fuzzy Logical Relationships (FLRs) of Erlang (Training) Traffic using Chen's (1996) Model

$A_6 \rightarrow A_5$ $A_5 \rightarrow A_4$ $A_4 \rightarrow A_5$ $A_5 \rightarrow A_6$ $A_6 \rightarrow A_6$ $A_5 \rightarrow A_5$ $A_4 \rightarrow A_4$ $A_6 \rightarrow A_8$ $A_8 \rightarrow A_5$
$A_5 \rightarrow A_7$ $A_7 \rightarrow A_7$ $A_7 \rightarrow A_3$ $A_7 \rightarrow A_8$ $A_8 \rightarrow A_{10}$ $A_{10} \rightarrow A_{12}$ $A_{12} \rightarrow A_{13}$ $A_{13} \rightarrow A_{13}$ $A_{13} \rightarrow A_{12}$
$A_{13} \rightarrow A_{14}$ $A_{14} \rightarrow A_{12}$ $A_{12} \rightarrow A_8$ $A_8 \rightarrow A_6$ $A_6 \rightarrow A_1$ $A_1 \rightarrow A_1$ $A_1 \rightarrow A_5$ $A_6 \rightarrow A_7$ $A_4 \rightarrow A_6$
$A_8 \rightarrow A_9$ $A_9 \rightarrow A_{11}$ $A_{11} \rightarrow A_{14}$ $A_{14} \rightarrow A_5$ $A_6 \rightarrow A_4$ $A_4 \rightarrow A_7$ $A_3 \rightarrow A_5$ $A_7 \rightarrow A_4$

Fuzzy Logical Relationship Groups (FLRGs) of Erlang (Training) Traffic using Chen's (1996) Model

Group 1:	$A_1 \rightarrow A_1$ $A_1 \rightarrow A_5$
Group 2:	$A_3 \rightarrow A_5$
Group 3:	$A_4 \rightarrow A_4$ $A_4 \rightarrow A_5$ $A_4 \rightarrow A_6$ $A_4 \rightarrow A_7$
Group 4:	$A_5 \rightarrow A_4$ $A_5 \rightarrow A_5$ $A_5 \rightarrow A_6$ $A_5 \rightarrow A_7$
Group 5:	$A_6 \rightarrow A_1$ $A_6 \rightarrow A_4$ $A_6 \rightarrow A_5$ $A_6 \rightarrow A_6$ $A_6 \rightarrow A_7$ $A_6 \rightarrow A_8$
Group 6:	$A_7 \rightarrow A_3$ $A_7 \rightarrow A_4$ $A_7 \rightarrow A_7$ $A_7 \rightarrow A_8$
Group 7:	$A_8 \rightarrow A_5$ $A_8 \rightarrow A_6$ $A_8 \rightarrow A_9$ $A_8 \rightarrow A_{10}$
Group 8:	$A_9 \rightarrow A_{11}$
Group 9:	$A_{10} \rightarrow A_{12}$
Group 10:	$A_{11} \rightarrow A_{14}$
Group 11:	$A_{12} \rightarrow A_8$ $A_{12} \rightarrow A_{13}$
Group 12:	$A_{13} \rightarrow A_{12}$ $A_{13} \rightarrow A_{13}$ $A_{13} \rightarrow A_{14}$
Group 13:	$A_{14} \rightarrow A_5$ $A_{14} \rightarrow A_{12}$

APPENDIXH

Fuzzy Logical Relationships (FLRs) of Erlang (Training) Traffic using Cheng et al (2008) Model

$A_5 \rightarrow A_4\ A_4 \rightarrow A_3\ A_3 \rightarrow A_4\ A_4 \rightarrow A_6\ A_6 \rightarrow A_6\ A_6 \rightarrow A_4\ A_3 \rightarrow A_2\ A_2 \rightarrow A_4\ A_4 \rightarrow A_5$
$A_3 \rightarrow A_6\ A_6 \rightarrow A_9\ A_9 \rightarrow A_5\ A_5 \rightarrow A_5\ A_5 \rightarrow A_7\ A_7 \rightarrow A_8\ A_8 \rightarrow A_2\ A_2 \rightarrow A_5\ A_5 \rightarrow A_8$
$A_8 \rightarrow A_{10}\ A_{10} \rightarrow A_{12}\ A_{12} \rightarrow A_{13}\ A_{13} \rightarrow A_{13}\ A_{13} \rightarrow A_{14}\ A_{14} \rightarrow A_{13}\ A_{13} \rightarrow A_9\ A_9 \rightarrow A_6$
$A_6 \rightarrow A_1\ A_1 \rightarrow A_1\ A_1 \rightarrow A_3\ A_3 \rightarrow A_7\ A_7 \rightarrow A_7\ A_7 \rightarrow A_2\ A_2 \rightarrow A_2\ A_2 \rightarrow A_7\ A_7 \rightarrow A_{10}$
$A_{10} \rightarrow A_{11}\ A_{11} \rightarrow A_{12}\ A_{12} \rightarrow A_{14}\ A_{14} \rightarrow A_4\ A_4 \rightarrow A_7\ A_7 \rightarrow A_3\ A_5 \rightarrow A_3$

Fuzzy Logical Relationship Groups (FLRGs) of Erlang (Training) Traffic using Cheng et al (2008) Model

Group 1:	$A_1 \rightarrow A_1\ A_1 \rightarrow A_3$
Group 2:	$A_2 \rightarrow A_2\ A_2 \rightarrow A_4\ A_2 \rightarrow A_5\ A_2 \rightarrow A_7$
Group 3:	$A_3 \rightarrow A_2\ A_3 \rightarrow A_4\ A_3 \rightarrow A_6\ A_3 \rightarrow A_7$
Group 4:	$A_4 \rightarrow A_3\ A_4 \rightarrow A_5\ A_4 \rightarrow A_6\ A_4 \rightarrow A_7$
Group 5:	$A_5 \rightarrow A_3\ A_5 \rightarrow A_4\ A_5 \rightarrow A_5\ A_5 \rightarrow A_7\ A_5 \rightarrow A_8$
Group 6:	$A_6 \rightarrow A_1\ A_6 \rightarrow A_4\ A_6 \rightarrow A_6\ A_6 \rightarrow A_9$
Group 7:	$A_7 \rightarrow A_2\ A_7 \rightarrow A_3\ A_7 \rightarrow A_7\ A_7 \rightarrow A_8\ A_7 \rightarrow A_{10}$
Group 8:	$A_8 \rightarrow A_2\ A_8 \rightarrow A_{10}$
Group 9:	$A_9 \rightarrow A_5\ A_9 \rightarrow A_6$
Group 10:	$A_{10} \rightarrow A_{11}\ A_{10} \rightarrow A_{12}$
Group 11:	$A_{11} \rightarrow A_{12}$
Group 12:	$A_{12} \rightarrow A_{13}\ A_{12} \rightarrow A_{14}$
Group 13:	$A_{13} \rightarrow A_9\ A_{13} \rightarrow A_{13}\ A_{13} \rightarrow A_{14}$
Group 14:	$A_{14} \rightarrow A_4\ A_{14} \rightarrow A_{13}$

APPENDIX I

FCM Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FuzzyCMeansClustering
{
       publicclassClusterCentroid//: ClusterPoint
       {
///<summary>
/// Basic constructor, used to initialize cluster centroid
///</summary>
///<param name="Index">
///<param name="Data">
///<param name="DataCount">
///<param name="DSum">
///<param name="MembershipSum">
publicdouble Index { get; set; }
publicdouble Data { get; set; }
publicdouble DataCount { get; set; }
publicdouble DSum { get; set; }
publicdouble MembershipSum { get; set; }
public ClusterCentroid(double index,double data)
         {
this.Index = index;
this.Data = data;
this.DSum = 0;
this.DataCount = 0;
this.MembershipSum = 0;
        }
       }


}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FuzzyCMeansClustering
{
       publicclassClusterPoint
       {
///<summary>
/// Gets or sets Index-coord of the datum value
///</summary>
publicdouble Index { get; set; }
///<summary>
/// Gets or sets Data-coord of the datum value
///</summary>
publicdouble Data { get; set; }
```

```
///<summary>
/// Gets or sets cluster index, the strongest membership value to a cluster
///</summary>
publicdouble ClusterIndex { get; set; }



///<summary>
/// Basic constructor
///</summary>
///<param name="index">Index-coord</param>
///<param name="data">Data-coord</param>
///<param name="z">ClusterIndex</param>
public ClusterPoint(double x, double data)
              {
                     this.Index = x;
this.Data = data;
this.ClusterIndex = -1;
              }

       }

}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace FuzzyCMeansClustering
{
publicclassFCM
    {

///<summary>
/// Array containing all points used by the algorithm
///</summary>
privateList<ClusterPoint> Points;

///<summary>
/// Array containing all clusters handled by the algorithm
///</summary>
privateList<ClusterCentroid> Clusters;

///<summary>
/// Array containing all clusters membership value of all points to each cluster
/// Fuzzy rules state that the sum of the membership of a point to all clusters     /// must
be 1
///</summary>
publicdouble[,] U;

publicList<double> myData;
privatebool isConverged = false;
privatedouble[] processedData;
```

```
publicdouble[] getProcessedData { get { return processedData; } }
publicbool Converged { get { return isConverged; } }

///<summary>
/// Gets or sets the current fuzzyness factor
///</summary>
privatedouble Fuzzyness;

///<summary>
/// Algorithm precision
///</summary>
privatedouble Eps = Math.Pow(10, -6);

///<summary>
/// Gets or sets objective function
///</summary>
publicdouble J { get; set; }

///<summary>
/// Gets or sets log message
///</summary>
publicint myDataLength;


///<summary>
/// Initialize the algorithm with data points and initial clusters
///</summary>
///<param name="points">The list of Points objects</param>
///<param name="clusters">The list of Clusters objects</param>
///<param name="fuzzy">The fuzzyness factor to be used, constant</param>
///<param name="myData">A working data, so that the GUI working data can be  ///
updated</param>
///<param name="numCluster">The number of clusters requested by the user from       /// the
GUI</param>
public FCM(List<ClusterPoint> points, List<ClusterCentroid> clusters, float  fuzzy, double[]
data, int numCluster)
        {
if (points == null)
            {
thrownewArgumentNullException("points");
            }

if (clusters == null)
            {
thrownewArgumentNullException("clusters");
            }


            processedData = data;


this.Points = points;
this.Clusters = clusters;
this.myDataLength = data.Length; //experimental
this.myData = newList<double>(this.myDataLength);
            U = newdouble[this.Points.Count, this.Clusters.Count];
this.Fuzzyness = fuzzy;

double diff;
```

```
// Iterate through all points to create initial U matrix
for (int i = 0; i <this.Points.Count; i++)
            {
ClusterPoint p = this.Points[i];
double sum = 0.0;

for (int j = 0; j <this.Clusters.Count; j++)
                {
ClusterCentroid c = this.Clusters[j];
                    diff = Math.Sqrt(Math.Pow(CalculateEuclideanDistance(p, c), 2.0));
                    U[i, j] = (diff == 0) ? Eps : diff;
                    sum += U[i, j];
                }
             }

this.RecalculateClusterMembershipValues();
        }

///<summary>
/// Private constructor
///</summary>
private FCM()
        {
        }

///<summary>
/// Recalculates the cluster membership values to ensure that
/// the sum of all membership values of a point to all clusters is 1
///</summary>
privatevoid RecalculateClusterMembershipValues()
        {

for (int i = 0; i <this.Points.Count; i++)
            {
double max = 0.0;
double min = 0.0;
double sum = 0.0;
double newmax = 0;
var p = this.Points[i];
//Normalize the entries
for (int j = 0; j <this.Clusters.Count; j++)
                {
                    max = U[i, j] > max ? U[i, j] : max;
                    min = U[i, j] < min ? U[i, j] : min;
                }
//Sets the values to the normalized values between 0 and 1
for (int j = 0; j <this.Clusters.Count; j++)
                {
                    U[i, j] = (U[i, j] - min) / (max - min);
                    sum += U[i, j];
                }
//Makes it so that the sum of all values is 1
for (int j = 0; j <this.Clusters.Count; j++)
                {
                    U[i, j] = U[i, j] / sum;
if (double.IsNaN(U[i, j]))
                    {
```

```
                ///Console.WriteLine("NAN value: point({0}) cluster({1}) sum {2} newmax
{3}", i, j, sum, newmax);
                    U[i, j] = 0.0;
                }
              //Console.WriteLine("ClusterIndex: point({0}) cluster({1}) min {2} max {3}
              value {4} p.ClusterIndex {5}", i, j, min, max, U[i, j], p.ClusterIndex);
                  newmax = U[i, j] > newmax ? U[i, j] : newmax;
              }
// ClusterIndex is used to store the strongest membership value to a cluster
               p.ClusterIndex = newmax;
           };
       }


///<summary>
/// Perform one step of the algorithm
///</summary>
publicvoid Step()
       {
for (int c = 0; c < Clusters.Count; c++)
           {
for (int h = 0; h < Points.Count; h++)
               {

double top;
                   top = CalculateEuclideanDistance(Points[h], Clusters[c]);
if (top < 1.0) top = Eps;

// sumTerms is the sum of distances from this data point to all
//clusters.
double sumTerms = 0.0;

for (int ck = 0; ck < Clusters.Count; ck++)
                   {
                    sumTerms += top / CalculateEuclideanDistance(Points[h],
            Clusters[ck]);

                   }
// Then the membership value can be calculated as...
                   U[h, c] = (double)(1.0 / Math.Pow(sumTerms, (2 / (this.Fuzzyness -
1))));
               }
           };


this.RecalculateClusterMembershipValues();
       }

///<summary>
/// Calculates Euclidean Distance distance between a point and a cluster centroid
///</summary>
///<param name="p">Point</param>
///<param name="c">Centroid</param>
///<returns>Calculated distance</returns>
privatedouble CalculateEuclideanDistance(ClusterPoint p, ClusterCentroid c)
       {
returnMath.Sqrt(Math.Pow(p.Data - c.Data, 2.0));
       }
```

```
///<summary>
/// Calculate the objective function
///</summary>
///<returns>The objective function as double value</returns>
publicdouble CalculateObjectiveFunction()
        {
double Jk = 0.0;

for (int i = 0; i <this.Points.Count;i++)
            {
for (int j = 0; j <this.Clusters.Count; j++)
                {
                    Jk += Math.Pow(U[i, j], this.Fuzzyness) *
       Math.Pow(this.CalculateEuclideanDistance(Points[i], Clusters[j]), 2);
                }
            }
return Jk;
        }

///<summary>
/// Calculates the centroids of the clusters
///</summary>
publicvoid CalculateClusterCentroids()
        {
//Console.WriteLine("Cluster Centroid calculation:");
for (int j = 0; j <this.Clusters.Count; j++)
            {
ClusterCentroid c = this.Clusters[j];
double l = 0.0;
                c.DataCount = 1;
                c.DSum = 0;
                c.MembershipSum = 0;

for (int i = 0; i <this.Points.Count; i++)
                {

ClusterPoint p = this.Points[i];
                    l = Math.Pow(U[i, j], this.Fuzzyness);
                    c.DSum += l * p.Data;
                    c.MembershipSum += l;

if (U[i, j] == p.ClusterIndex)
                    {
                        c.DataCount += 1;
                    }
                }

                c.Data = c.DSum / c.MembershipSum;
             }

//update the original data
double[] tempData = newdouble[this.myDataLength];

for (int j = 0; j <this.Points.Count; j++)
            {
for (int i = 0; i <this.Clusters.Count; i++)
                {
ClusterPoint p = this.Points[j];
```

```
if (U[j, i] == p.ClusterIndex)
                    {
//tempData.Set((int)p.X, this.Clusters[i].Data;
                        tempData[(int)p.Index] = this.Clusters[i].Data;
                    }
                }
            }

            processedData = tempData;
        }

///<summary>
/// Perform a complete run of the algorithm until the desired accuracy is
///achieved.
/// For demonstration issues, the maximum Iteration counter is set to 50.
///</summary>
///<param name="accuracy">Algorithm accuracy</param>
///<returns>The number of steps the algorithm needed to complete</returns>

    }

}
////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace FuzzyCMeansClustering
{
classDataHelper
    {

privatestring FileName { get; set; }

public DataHelper(string _fileName)
        {
this.FileName = _fileName;
        }

publicdouble[] GetData()
        {

List<double> dataList = newList<double>();

using (StreamReader _rdr = newStreamReader(FileName))
            {
string line = "";
while ((line = _rdr.ReadLine()) != null)
                {
double result;
string thisLine = line.Trim();
if (!double.TryParse(thisLine, out result))
                    {
returnnull;
                    }
                    dataList.Add(result);
                }
```

```
            }
return (from c in dataList
orderby c ascending
select c).ToArray();

        }
    }
}
////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using System.IO;
using System.Drawing.Imaging;
using System.Diagnostics;

namespace FuzzyCMeansClustering
{
publicdelegatevoidMyDelegate(string input);

delegatevoidSetTextCallback(string text);

publicdelegatevoidDelegateThreadFinished();



publicpartialclassForm1 : Form
    {
privatedouble[] Data { get; set; }


privateBackgroundWorker backgroundWorker;
publicStopwatch stopWatch;


public Form1()
        {
            InitializeComponent();

            backgroundWorker = newBackgroundWorker();
            backgroundWorker.WorkerReportsProgress = true;
            backgroundWorker.WorkerSupportsCancellation = true;
            backgroundWorker.DoWork += newDoWorkEventHandler(backgroundWorker1_DoWork);
            backgroundWorker.RunWorkerCompleted +=
newRunWorkerCompletedEventHandler(backgroundWorker1_RunWorkerCompleted);
            backgroundWorker.ProgressChanged +=
newProgressChangedEventHandler(backgroundWorker1_ProgressChanged);

            stopWatch = newStopwatch();

        }


privatevoid button2_Click(object sender, EventArgs e)
```

```
        {
            button2.Enabled = false;
            button3.Enabled = true;

            stopWatch.Reset();
            stopWatch.Start();
            backgroundWorker.RunWorkerAsync();
        }


privatevoid backgroundWorker1_ProgressChanged(object sender, ProgressChangedEventArgs e)
       {
              toolStripProgressBar1.Value = e.ProgressPercentage;
              toolStripStatusLabel1.Text = e.UserState asString;
       }

privatevoid backgroundWorker1_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs
e)
       {

if ((e.Cancelled == true))
           {
               toolStripStatusLabel1.Text = "Canceled!";
               button2.Enabled = true;
               button3.Enabled = false;
           }

elseif (!(e.Error == null))
           {
               toolStripStatusLabel1.Text = ("Error: " + e.Error.Message);
           }

            toolStripProgressBar1.Enabled = false;
this.button2.Enabled = true;
this.button3.Enabled = false;
       }



// This method will run on a thread other than the UI thread.
// Be sure not to manipulate any Windows Forms controls created
// on the UI thread from this method.
privatevoid backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            backgroundWorker.ReportProgress(0, "Working...");

int numClusters = (int)numericUpDown2.Value;
int maxIterations = (int)numericUpDown3.Value;
double accuracy = (double)numericUpDown4.Value;


List<ClusterPoint> points = newList<ClusterPoint>();


for (int index = 0; index <this.Data.Length; ++index)
                {
                    points.Add(newClusterPoint(index, this.Data[index]));
                }
```

```
List<ClusterCentroid> centroids = newList<ClusterCentroid>();

//Create random points to use a the cluster centroids
Random random = newRandom();
for (int i = 0; i < numClusters; i++)
            {
int randomNumber1 = random.Next(this.Data.Length);
int randomNumber2 = random.Next(this.Data.Length);
                centroids.Add(newClusterCentroid(randomNumber1,this.Data[randomNumber2]));
            }
FCM alg = newFCM(points, centroids, 2, this.Data,(int)numericUpDown2.Value);

int k = 0;
do
            {
if ((backgroundWorker.CancellationPending == true))
                {
                    e.Cancel = true;
break;
                }
else
                {

                    k++;
                    alg.J = alg.CalculateObjectiveFunction();
                    alg.CalculateClusterCentroids();
                    alg.Step();
double Jnew = alg.CalculateObjectiveFunction();
Console.WriteLine("Run method i={0} accuracy = {1} delta={2}", k, alg.J, Math.Abs(alg.J -
Jnew));
                    toolStripStatusLabel2.Text = "Precision " + Math.Abs(alg.J - Jnew);

// Format and display the TimeSpan value.
string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}", stopWatch.Elapsed.Hours,
stopWatch.Elapsed.Minutes, stopWatch.Elapsed.Seconds, stopWatch.Elapsed.Milliseconds / 10);
                    toolStripStatusLabel3.Text = "Duration: " + elapsedTime;


                    backgroundWorker.ReportProgress((100 * k) / maxIterations, "Iteration "
+ k);

if (Math.Abs(alg.J - Jnew) < accuracy) break;
                }
            }
while (maxIterations > k);
Console.WriteLine("Done.");

            stopWatch.Stop();
// Get the elapsed time as a TimeSpan value.
TimeSpan ts = stopWatch.Elapsed;

// Save the segmented data points
using (StreamWriter _swr = newStreamWriter("output.txt"))
            {
double[] processedData = alg.getProcessedData;
foreach (double d in processedData)
                {
                    _swr.WriteLine(d.ToString());
```

```
                }
            }
//   return;

// Create a new data for each cluster in order to extract the features from       // the
original data
double[,] Matrix = alg.U;
List<double[]> clusterData = newList<double[]>(centroids.Count);
for (int i = 0; i < centroids.Count; i++)
            {
                clusterData.Add(newdouble[points.Count]);
            }

for (int j = 0; j < points.Count; j++)
            {
for (int i = 0; i < centroids.Count; i++)
                {
ClusterPoint p = points[j];
if (Matrix[j, i] == p.ClusterIndex)
                    {
double[] particularCluster = clusterData[i];
                        particularCluster[(int)p.Index] = p.Data;

                    }
                }
            }

// Save the data for each segmented cluster
for (int i = 0; i < centroids.Count; i++)
            {
double[] thisCluster = clusterData[i];
using (StreamWriter writer = newStreamWriter("Cluster" + i + ".txt"))
                {
foreach (double d in thisCluster)
                    {
                        writer.WriteLine(d.ToString());
                    }
                }
            }

using (StreamWriter sw = newStreamWriter("u_matrix.txt"))
            {
for (int i = 0; i < alg.U.GetLength(0); i++)
                {
                    sw.WriteLine("Index i = " + i.ToString());
for (int j = 0; j < alg.U.GetLength(1); j++)
                    {
                        sw.WriteLine("Data: " + alg.U[i, j].ToString());
                    }
                }
            }


// Resource cleanup...more work to do here to avoid memory problems!!!
            backgroundWorker.ReportProgress(100, "Done in " + k + " iterations.");
////alg.Dispose();
for (int i = 0; i < points.Count; i++)
            {
                points[i] = null;
```

```
            }
for (int i = 0; i < centroids.Count; i++)
            {
                centroids[i] = null;
            }
            alg = null;
//centroids.Clear();
//points.Clear();
        }


// Open data file
privatevoid openToolStripMenuItem_Click(object sender, EventArgs e)
        {
if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
try
                {
this.Data = newDataHelper(openFileDialog.FileName).GetData();
if (this.Data == null) thrownewException("File contains invalid characters");
                    button2.Enabled = true;
                }

catch (Exception ef)
                {
MessageBox.Show("Failed loading data: " + ef.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

privatevoid button3_Click(object sender, EventArgs e)
        {
if (backgroundWorker != null)
            {
                backgroundWorker.CancelAsync();
            }

            toolStripStatusLabel1.Text = "Aborting, please wait...";

        }

privatevoid label1_Click(object sender, EventArgs e)
        {

        }

privatevoid Form1_Load(object sender, EventArgs e)
        {

        }

privatevoid numericUpDown4_ValueChanged(object sender, EventArgs e)
        {

        }

privatevoid fileToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
        }

privatevoid mainMenuStrip_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
        {

        }

privatevoid numericUpDown2_ValueChanged(object sender, EventArgs e)
        {

        }

privatevoid label3_Click(object sender, EventArgs e)
        {

        }

privatevoid numericUpDown3_ValueChanged(object sender, EventArgs e)
        {

        }

privatevoid label4_Click(object sender, EventArgs e)
        {

        }

privatevoid toolStripStatusLabel2_Click(object sender, EventArgs e)
        {

        }

privatevoid toolStripStatusLabel3_Click(object sender, EventArgs e)
        {

        }

privatevoid toolStripStatusLabel2_Click_1(object sender, EventArgs e)
        {

        }

privatevoid statusStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
        {

        }

privatevoid toolStripProgressBar1_Click(object sender, EventArgs e)
        {

        }

privatevoid toolStripStatusLabel1_Click(object sender, EventArgs e)
        {

        }
```

```
    }
}
//////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace FuzzyCMeansClustering
{
staticclassProgram
    {
///<summary>
/// The main entry point for the application.
///</summary>
        [STAThread]
staticvoid Main()
        {
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(newForm1());
        }
    }
}
```

APPENDIX J

PSO Code

```
using System;
using System.Collections.Generic;
using System.Text;

namespace PSOLib
{
publicstructBestValue
   {
publicfloat W1 { get; set; }
publicfloat W2 { get; set; }
publicfloat W3 { get; set; }
publicfloat W4 { get; set; }
publicfloat W5 { get; set; }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Text;

namespace PSOLib
{
publicclassParticle
   {
public Particle()
      {
Random __myRandom = newRandom(DateTime.Now.Millisecond *
newRandom(System.DateTime.Now.Second).Next(0,newRandom().Next(0,57)));

this.VelocityOne = (float)__myRandom.NextDouble() / 100;
         System.Threading.Thread.Sleep(__myRandom.Next(0, 300));
this.VelocityTwo = (float)__myRandom.NextDouble() / 100;
         System.Threading.Thread.Sleep(__myRandom.Next(0, 300));
this.VelocityThree = (float)__myRandom.NextDouble() / 100;
         System.Threading.Thread.Sleep(__myRandom.Next(0, 300));
this.VelocityFour = (float)__myRandom.NextDouble() / 100;
         System.Threading.Thread.Sleep(__myRandom.Next(0, 300));
this.VelocityFive = (float)__myRandom.NextDouble() / 100;

      }


publicdouble VelocityOne { get; set; }
publicdouble VelocityTwo { get; set; }
publicdouble VelocityThree { get; set; }
publicdouble VelocityFour { get; set; }
publicdouble VelocityFive { get; set; }

publicfloat CurrentPosition1 { get; set; }
publicfloat CurrentPosition2 { get; set; }
publicfloat CurrentPosition3 { get; set; }
publicfloat CurrentPosition4 { get; set; }
publicfloat CurrentPosition5 { get; set; }

publicBestValue LocalBest { get; set; }
publicstaticBestValue Globalbest { get; set; }

   }
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Text;

namespace PSOLib
{
publicclassPSOData
   {
public PSOData()
      {
         ParticleCount = 5;
         MinimumSE = 3;
         EnhancedComputations = false;
         EnhancedComputations4 = false;
         EnhancedComputations5 = false;
         NumberOfIterations = 500;
         C1 = 2.0F;
         C2 = 2.0F;
         InertialWtCfft = 1.4F;// declare and assign the variable Inertial cofficient
      }

publicint ParticleCount { get; set; }
publiculong MinimumSE { get; set; }
publicint NumberOfIterations { get; set; }

publicbool EnhancedComputations { get; set; }
publicbool EnhancedComputations4 { get; set; }
publicbool EnhancedComputations5 { get; set; }

publicdouble InertialWtCfft { get; set; } //previously was float

publicfloat T1 { get; set; } //previously were ints
publicfloat T2 { get; set; }
publicfloat T3 { get; set; }
publicfloat T4 { get; set; }
publicfloat T5 { get; set; }

publicfloat C1 { get; set; }
publicfloat C2 { get; set; }

publicfloat Data { get; set; }

publicfloat W1 { get; set; }
publicfloat W2 { get; set; }
publicfloat W3 { get; set; }
publicfloat W4 { get; set; }
publicfloat W5 { get; set; }

   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;


namespace PSOLib
{
publicstaticclassPSOEngine
   {
```

```
static PSOEngine()
      {

        InitEngine();
      }



privatestaticvoid InitEngine()
      {
double _r1, _r2;
         GetRandomValues(out _r1, out _r2);
         R1 = _r1;
         R2 = _r2;
      }


privatestaticdouble R1 { get; set; }
privatestaticdouble R2 { get; set; }


privatestaticvoid GetRandomValues(outdouble r1, outdouble r2)
      {
Random _rand = newRandom(System.DateTime.Now.Millisecond);
         r1 = _rand.NextDouble();
         r2 = _rand.NextDouble();

      }

publicstaticPSOResult Compute(PSOData __input)
      {

ulong[] oldSquaredErrors = newulong[__input.ParticleCount + 1];
ulong[] currentSquaredErrors = newulong[__input.ParticleCount];
//ulong initialSE;


ulong initialSE = long.MaxValue;
bool doGlobalUpdate = false;
Particle[] particles = newParticle[__input.ParticleCount];
double INERTIAL_WEIGHT_COEFFICIENT = __input.InertialWtCfft;// calls inertial coefficient field from PSOData parameter in
the PSOData.cs object for current velocity calculation
ulong dumpVal = 0;

double[,] initialVelocities;
int luckyParticle = -1;

ulong[] intermediateSE = newulong[5];

if (__input.EnhancedComputations)
         {
            initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3, __input.W1, __input.W2, __input.W3);
         }
elseif (__input.EnhancedComputations4)
         {
            initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3,__input.T4, __input.W1, __input.W2,
__input.W3,  __input.W4);
         }
elseif (__input.EnhancedComputations5)
         {
            initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3, __input.T4,__input.T5, __input.W1,
__input.W2, __input.W3, __input.W4, __input.W5);
```

```
        }
else
        {
          initialSE = GetSquaredErrors(__input.Data, __input.T1, __input.T2,__input.W1, __input.W2);
        }

        dumpVal = initialSE;

        Initialise(particles, __input, out initialVelocities);
        Initialise(oldSquaredErrors, initialSE);
        Initialise(currentSquaredErrors, 0);

for (int i = 0; i < intermediateSE.Length; i++)
        {
          intermediateSE[i] = initialSE;
        }

for (int computeIterations = 0; computeIterations < __input.NumberOfIterations; computeIterations++)
        {
          doGlobalUpdate = false;

for (int particleNumber = 0; particleNumber < __input.ParticleCount; particleNumber++)
          {
Particle currentParticle = particles[particleNumber];
            currentParticle.VelocityOne = ComputeVelocity(__input, currentParticle.LocalBest.W1, Particle.Globalbest.W1,
currentParticle.CurrentPosition1, currentParticle.VelocityOne);
            currentParticle.VelocityTwo = ComputeVelocity(__input, currentParticle.LocalBest.W2, Particle.Globalbest.W2,
currentParticle.CurrentPosition2, currentParticle.VelocityTwo);

if (__input.EnhancedComputations)
            {
              currentParticle.VelocityThree = ComputeVelocity(__input, currentParticle.LocalBest.W3, Particle.Globalbest.W3,
currentParticle.CurrentPosition3, currentParticle.VelocityThree);
            }

if (__input.EnhancedComputations4)
            {
              currentParticle.VelocityThree = ComputeVelocity(__input, currentParticle.LocalBest.W3, Particle.Globalbest.W3,
currentParticle.CurrentPosition3, currentParticle.VelocityThree);
              currentParticle.VelocityFour = ComputeVelocity(__input, currentParticle.LocalBest.W4, Particle.Globalbest.W4,
currentParticle.CurrentPosition4, currentParticle.VelocityFour);
            }

if (__input.EnhancedComputations5)
            {
              currentParticle.VelocityThree = ComputeVelocity(__input, currentParticle.LocalBest.W3, Particle.Globalbest.W3,
currentParticle.CurrentPosition3, currentParticle.VelocityThree);
              currentParticle.VelocityFour = ComputeVelocity(__input, currentParticle.LocalBest.W4, Particle.Globalbest.W4,
currentParticle.CurrentPosition4, currentParticle.VelocityFour);
              currentParticle.VelocityFive = ComputeVelocity(__input, currentParticle.LocalBest.W5, Particle.Globalbest.W5,
currentParticle.CurrentPosition5, currentParticle.VelocityFive);
            }

            currentParticle.VelocityOne = Normalize(currentParticle.VelocityOne);
            currentParticle.VelocityTwo = Normalize(currentParticle.VelocityTwo);

if (__input.EnhancedComputations)
            {
              currentParticle.VelocityThree = Normalize(currentParticle.VelocityThree);
            }

if (__input.EnhancedComputations4)
            {
```

```
                currentParticle.VelocityThree = Normalize(currentParticle.VelocityThree);
                currentParticle.VelocityFour = Normalize(currentParticle.VelocityFour);
            }

if (__input.EnhancedComputations5)
            {
                currentParticle.VelocityThree = Normalize(currentParticle.VelocityThree);
                currentParticle.VelocityFour = Normalize(currentParticle.VelocityFour);
                currentParticle.VelocityFive = Normalize(currentParticle.VelocityFive);
            }


            currentParticle.CurrentPosition1 = (float)(currentParticle.CurrentPosition1 + currentParticle.VelocityOne);
            currentParticle.CurrentPosition2 = (float)(currentParticle.CurrentPosition2 + currentParticle.VelocityTwo);

if (__input.EnhancedComputations)
            {
                currentParticle.CurrentPosition3 = (float)(currentParticle.CurrentPosition3 + currentParticle.VelocityThree);
            }

elseif (__input.EnhancedComputations4)
            {
                currentParticle.CurrentPosition3 = (float)(currentParticle.CurrentPosition3 + currentParticle.VelocityThree);
                currentParticle.CurrentPosition4 = (float)(currentParticle.CurrentPosition4 + currentParticle.VelocityFour);
            }

elseif (__input.EnhancedComputations5)
            {
                currentParticle.CurrentPosition3 = (float)(currentParticle.CurrentPosition3 + currentParticle.VelocityThree);
                currentParticle.CurrentPosition4 = (float)(currentParticle.CurrentPosition4 + currentParticle.VelocityFour);
                currentParticle.CurrentPosition5 = (float)(currentParticle.CurrentPosition5 + currentParticle.VelocityFive);
            }

        } //end update-for

for (int currPat = 0; currPat < __input.ParticleCount; currPat++) //Per-particle iterator
        {
if (!__input.EnhancedComputations)
            {

if (computeIterations == 1)
                {
//This is during the second iteration
                    intermediateSE[currPat] = currentSquaredErrors[currPat]; //keep for a rainy day.
                }


                currentSquaredErrors[currPat] = GetSquaredErrors(__input.Data, __input.T1, __input.T2,
particles[currPat].CurrentPosition1, particles[currPat].CurrentPosition2);


            }
elseif (__input.EnhancedComputations)
            {
if (computeIterations == 1)
                {
//This is during the second iteration
                    intermediateSE[currPat] = currentSquaredErrors[currPat]; //keep for a rainy day.
                }

                currentSquaredErrors[currPat] = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3,
particles[currPat].CurrentPosition1, particles[currPat].CurrentPosition2, particles[currPat].CurrentPosition3);
```

```
                }

elseif (__input.EnhancedComputations4)
                {
if (computeIterations == 1)
                  {
//This is during the second iteration
                    intermediateSE[currPat] = currentSquaredErrors[currPat]; //keep for a rainy day.
                  }

                  currentSquaredErrors[currPat] = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3,__input.T4,
particles[currPat].CurrentPosition1, particles[currPat].CurrentPosition2, particles[currPat].CurrentPosition3,
particles[currPat].CurrentPosition4);
                }

elseif (__input.EnhancedComputations5)
                {
if (computeIterations == 1)
                  {
//This is during the second iteration
                    intermediateSE[currPat] = currentSquaredErrors[currPat]; //keep for a rainy day.
                  }

                  currentSquaredErrors[currPat] = GetSquaredErrors(__input.Data, __input.T1, __input.T2, __input.T3, __input.T4,
__input.T5, particles[currPat].CurrentPosition1, particles[currPat].CurrentPosition2, particles[currPat].CurrentPosition3,
particles[currPat].CurrentPosition4, particles[currPat].CurrentPosition5);
                }


//Now determine if to perform  a global update.
if (currentSquaredErrors[currPat] < oldSquaredErrors[currPat])
                {
                  doGlobalUpdate = true;
                  oldSquaredErrors[currPat] = currentSquaredErrors[currPat]; //update the SE
                  particles[currPat].LocalBest = newBestValue { W1 = particles[currPat].CurrentPosition1, W2 =
particles[currPat].CurrentPosition2, W3 = particles[currPat].CurrentPosition3, W4 = particles[currPat].CurrentPosition4, W5 =
particles[currPat].CurrentPosition5 };

                }
              } //end updating the SE

if (doGlobalUpdate)
              {
                oldSquaredErrors[__input.ParticleCount] = currentSquaredErrors.Min();

for (int par = 0; par < __input.ParticleCount; par++)
                {
if (currentSquaredErrors[par] == oldSquaredErrors[__input.ParticleCount])
                  {
Particle.Globalbest = newBestValue { W1 = particles[par].CurrentPosition1, W2 = particles[par].CurrentPosition2, W3 =
particles[par].CurrentPosition3, W4 = particles[par].CurrentPosition4, W5= particles[par].CurrentPosition5 };

                    luckyParticle = par;
break;
                  }//this must be the particle who is the global best for this run.
                }

if (oldSquaredErrors[__input.ParticleCount] <= __input.MinimumSE)
                {
returnnewPSOResult { Successful = true,IntermediateSquareErrors = intermediateSE, SquaredErrorValues =
currentSquaredErrors,Particles = particles,InitialVelocities = initialVelocities, BestParticleIndex = luckyParticle, IterationCount =
++computeIterations, SEValue = oldSquaredErrors[__input.ParticleCount], DebugValue = dumpVal};
```

```
                }
            }


        } //the outer iterator
//This happens if the stopping criterion regarding the SE doesn't hold.

//  return new PSOResult { Successful = false, Result = Particle.Globalbest, IterationCount = __input.NumberOfIterations,SEValue =
oldSquaredErrors[__input.ParticleCount], DebugValue = dumpVal };
returnnewPSOResult { Successful = false, IntermediateSquareErrors = intermediateSE, SquaredErrorValues =
currentSquaredErrors,Particles = particles, InitialVelocities = initialVelocities, BestParticleIndex = luckyParticle, IterationCount =
__input.NumberOfIterations, SEValue = oldSquaredErrors[__input.ParticleCount], DebugValue = dumpVal };

    }

privatestaticvoid Initialise(Particle[] __particles, PSOData __data, outdouble[,] __initialVelocities)
    {
double[,] _velocities = newdouble[__particles.Length,5];


for (int i = 0; i < __particles.Length; i++)
        {
            __particles[i] = newParticle();
            __particles[i].CurrentPosition1 = __data.W1;
            __particles[i].CurrentPosition2 = __data.W2;
            __particles[i].CurrentPosition3 = __data.W3;
            __particles[i].CurrentPosition4 = __data.W4;
            __particles[i].CurrentPosition5 = __data.W5;

            __particles[i].LocalBest = newBestValue { W1 = __data.W1, W2 = __data.W2, W3 = __data.W3, W4 = __data.W4, W5 =
__data.W5 };


//At the same time, populate the relevant fields of the velocities array
            _velocities[i, 0] = __particles[i].VelocityOne;
            _velocities[i, 1] = __particles[i].VelocityTwo;
            _velocities[i, 2] = __particles[i].VelocityThree;
            _velocities[i, 3] = __particles[i].VelocityFour;
            _velocities[i, 4] = __particles[i].VelocityFive;

        }

Particle.Globalbest = newBestValue { W1 = __data.W1, W2 = __data.W2, W3 = __data.W3, W4 = __data.W4, W5 = __data.W5 };
//since we're neighbors.
        __initialVelocities = _velocities; //Here is the magic.
    }

privatestaticvoid Initialise(ulong[] __values, ulong seedValue)
    {
for (int i = 0; i < __values.Length; i++)
        {
            __values[i] = seedValue;

        }

    }

//supports t2 and w2
privatestaticulong GetSquaredErrors(float __data,float __T1, float __T2, float __W1, float __W2)
    {
checked//double insurance to ensure no overflows.
        {
```

```
return (ulong)(((__T1 * __W1) + (__T2 * __W2) - __data) * ((__T1 * __W1) + (__T2 * __W2) - __data));
            }
        }

//suports t3 and w3
privatestaticulong GetSquaredErrors(float __data, float __T1, float __T2, float __T3, float __W1, float __W2, float __W3)
        {
checked//double insurance to ensure no overflows.
            {
return (ulong)(((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3) - __data) * ((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3)
- __data));
            }
        }

//supports t4 and w4
privatestaticulong GetSquaredErrors(float __data, float __T1, float __T2, float __T3,float __T4, float __W1, float __W2, float
__W3, float __W4)
        {
checked//double insurance to ensure no overflows.
            {
return (ulong)(((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3) + (__T4 * __W4) - __data) * ((__T1 * __W1) + (__T2 * __W2)
+ (__T3 * __W3) + (__T4 * __W4) - __data));
            }
        }

//supports t5 and w5
privatestaticulong GetSquaredErrors(float __data, float __T1, float __T2, float __T3, float __T4, float __T5, float __W1, float __W2,
float __W3, float __W4, float __W5)
        {
checked//double insurance to ensure no overflows.
            {
return (ulong)(((__T1 * __W1) + (__T2 * __W2) + (__T3 * __W3) + (__T4 * __W4) +(__T5 * __W5) - __data) * ((__T1 * __W1)
+ (__T2 * __W2) + (__T3 * __W3) + (__T4 * __W4) + (__T5 * __W5) - __data));
            }
        }

privatestaticdouble Normalize(double __raw)
        {
if (__raw > 0.01D)
            {
return 0.01D;
            }
elseif (__raw < -0.01D)
            {
return -0.01D;
            }
else
            {
return __raw;
            }

        }

privatestaticdouble ComputeVelocity(PSOData __input, float __lbest, float __gbest, float __position, double __velocity)
        {
return (__input.InertialWtCfft * __velocity) + ((__input.C1 * R1) * (__lbest - __position)) + ((__input.C2 * R2) * (__gbest -
__position));
        }
    }


}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace PSOLib
{
publicclassPSOResult
   {
publicbool Successful { get; set; }
publicBestValue Result { get; set; } //I shall leave this here just in case I'll need it someday.
publicint IterationCount { get; set; }
publiculong SEValue { get; set; }
publiculong DebugValue { get; set; }



publicdouble[,] InitialVelocities { get; set; }
publicParticle[] Particles { get; set; }
publicint BestParticleIndex { get; set; }


publiculong[] SquaredErrorValues { get; set; }
publiculong[] IntermediateSquareErrors { get; set; }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace AppFramework
{
publicclassDataUtility
   {
publicstring FilePath { get; set; }

public DataUtility(string _filesPath)
      {
this.FilePath = _filesPath;
      }

publicList<PSOItem> ReadDataItems()
      {
List<PSOItem> _result = newList<PSOItem>();

using (FileStream _fStream = File.Open(this.FilePath,FileMode.Open))
         {
using (StreamReader _rd = newStreamReader(_fStream))
            {
string[] lines = _rd.ReadToEnd().Split(newstring[] {"\r\n"}, StringSplitOptions.RemoveEmptyEntries);
//  _fsgs = lines.ToList();
// _rawData = null;

foreach (string l in lines)
               {
string[] rawItemsSeq = l.Split('%');
                  _result.Add(newPSOItem() {Data = float.Parse(rawItemsSeq[0]),FuzzySet = rawItemsSeq[1]});

               }
            }
         }
```

```
return _result;

        }

publicbool WriteDataItems(List<float> _rawData, List<string> _fsgs, int dataSetNo)
        {
try
            {

if (Directory.Exists(this.FilePath))
                {
File.Delete(dataSetNo + ".dataset"); //remove the old data.
                }
else
                {
Directory.CreateDirectory(this.FilePath);
                }

using (FileStream _stream = File.Create(this.FilePath + dataSetNo.ToString() + ".dataset"))
                {
using (StreamWriter _wr = newStreamWriter(_stream))
                    {
for (int i = 0; i < _rawData.Count; i++)
                        {
                            _wr.Write(_rawData.ToArray()[i]);
                            _wr.Write('%');
                            _wr.Write(_fsgs.ToArray()[i]);
                            _wr.WriteLine('%');

                        }
                    }
                }

returntrue;
            }

catch
            {
returnfalse;
            }

        }

publicbool WriteRules(List<string> _rawRules, int ruleSetNo)
        {
try
            {

if (Directory.Exists(this.FilePath))
                {
File.Delete(ruleSetNo + ".ruleset"); //remove the old data.
                }
else
                {
Directory.CreateDirectory(this.FilePath);
                }

using (FileStream _stream = File.Create(this.FilePath + ruleSetNo.ToString() + ".ruleset"))
                {
using (StreamWriter _wr = newStreamWriter(_stream))
                    {
```

```
//magic
for (int i = 0; i < _rawRules.Count; i = i + 5)
                    {
                    _wr.Write(_rawRules.ToArray()[i]);
                    _wr.Write('%');
                    _wr.Write(_rawRules.ToArray()[i + 1]);
                    _wr.Write('%');
                    _wr.Write(_rawRules.ToArray()[i + 2]);
                    _wr.Write('%');
                    _wr.Write(_rawRules.ToArray()[i + 3]);
                    _wr.Write('%');
                    _wr.Write(_rawRules.ToArray()[i + 4]);
                    _wr.WriteLine('%');
                    }
                }
            }

returntrue;
        }

catch
        {
returnfalse;
        }
    }

publicList<Rule> ReadRules()
    {

List<Rule> _results = newList<Rule>();
short indexNo = 1;

using (FileStream _fs = File.Open(this.FilePath, FileMode.Open))
        {
using (StreamReader _rd = newStreamReader(_fs))
            {
string[] rawLines = _rd.ReadToEnd().Split(newstring[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

foreach (string r in rawLines)
                {
string[] oneLine = r.Split('%');
                    _results.Add(newRule(indexNo++, oneLine));

                }
            }
        }


return _results;
    }


  }

publicstaticclassUtilities
  {
publicstaticbool TryParseFuzzySet(string _item)
    {
return (System.Text.RegularExpressions.Regex.IsMatch(_item, @"^A\d+$"));
    }
  }
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Windows.Forms;
using PSOLib;

namespace AppFramework
{
publicclassGUIHelper
   {
privatestring _path = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
public GUIHelper()
      {
//ctor
      }

//Helper classes
//I need a method to list all the dataset files in the folder, and then give their names

publicstring[] GetAvailableDataSets()
      {

if (!Directory.Exists(_path))
         {
//There are available datasets.
returnnull; //calling code will catch this condition.
         }

var f = from i inDirectory.GetFiles(_path)
where i.Contains(".dataset")
select i; //I couldn't help myself! I needed to use it!!!!!!

return f.ToArray();

      }

publicbool LoadIfRuleExists(int _fileIndex, outList<Rule> _rules)
      {
string __targetFileName = _path + _fileIndex.ToString() + ".ruleset";

if (File.Exists(__targetFileName))
         {
DataUtility __dataUtil = newDataUtility(__targetFileName);
            _rules = __dataUtil.ReadRules();
returntrue;
         }

//File doesn't even exist.
         _rules = null;
returnfalse;
      }

publicbool ParseData(IEnumerable<TextBox> _ctrl)
      {
bool __isValid = true;
float __dummyF;
int __dummyI;
ulong __dummyU;
```

```
foreach (TextBox c in _ctrl)
        {
if (c.Enabled == false) //sometimes _txtW3 is disabled. If so, do not consider it.
            {
continue;
            }

switch ((string)c.Tag)
            {
case"float":
                __isValid = float.TryParse(c.Text.Trim(), out __dummyF) && c.Text.Trim() != string.Empty;
break;

case"int":
                __isValid = int.TryParse(c.Text.Trim(), out __dummyI) && c.Text.Trim() != string.Empty;
break;

case"ulong":
                 __isValid = ulong.TryParse(c.Text.Trim(), out __dummyU) && c.Text.Trim() != string.Empty;
break;

default:
//Something I didn't forsee
                __isValid = false;
break;

            }

if (__isValid == false)
            {
return __isValid; //stop here.
            }
        }

return __isValid;
    }

publicPSOData RetrieveAppropriateItem(Rule _criterion, List<PSOItem> _items)
    {
PSOItem[] _itemsArray = _items.ToArray();

if (_criterion.Operands[2] == string.Empty && _criterion.Operands[3] == string.Empty && _criterion.Operands[4] == string.Empty)
        {
//Then there is no third criterion. The search can start at element three, index 2

for (int i = 2; i < _itemsArray.Length; i++)
            {
//ORIGINAL CODE
//if (_itemsArray[i - 1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]))
if (_itemsArray[i-1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]))
                {
//this is the element
returnnewPSOData { Data = _itemsArray[i].Data, T1 = _itemsArray[i - 1].Data, T2 = _itemsArray[i - 2].Data, T3 = -100, T4 = -100,
T5 = -100 };
                }
            }

        }

elseif (_criterion.Operands[3] == string.Empty && _criterion.Operands[4] == string.Empty)
        {
// look for a third criterion but dont look for fourth and fifth.
```

```
for (int i = 3; i < _itemsArray.Length; i++)
            {
//ORIGINAL CODE
//if (_itemsArray[i - 1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]) &&
_itemsArray[i - 3].BelongsTo(_criterion.Operands[2]))
if (_itemsArray[i - 1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]) &&
_itemsArray[i - 3].BelongsTo(_criterion.Operands[2]))
                {
returnnewPSOData() { Data = _itemsArray[i].Data, T1 = _itemsArray[i - 1].Data, T2 = _itemsArray[i - 2].Data, T3 = _itemsArray[i -
3].Data, T4 = -100, T5 = -100 };
                }

            }

        }
elseif (_criterion.Operands[4] == string.Empty)
        {
// look for a fourth criterion but not fifth.
for (int i = 4; i < _itemsArray.Length; i++)
            {
//ORIGINAL CODE
//if (_itemsArray[i - 1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]) &&
_itemsArray[i - 3].BelongsTo(_criterion.Operands[2]) && _itemsArray[i - 4].BelongsTo(_criterion.Operands[3]))
if (_itemsArray[i-1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]) &&
_itemsArray[i - 3].BelongsTo(_criterion.Operands[2]) && _itemsArray[i - 4].BelongsTo(_criterion.Operands[3]))
                {
returnnewPSOData() { Data = _itemsArray[i].Data, T1 = _itemsArray[i - 1].Data, T2 = _itemsArray[i - 2].Data, T3 = _itemsArray[i -
3].Data, T4 = _itemsArray[i - 4].Data, T5 = -100 };
                }

            }

        }
else
        {
// look for a fifth criterion.
for (int i = 5; i < _itemsArray.Length; i++)
            {

if (_itemsArray[i-1].BelongsTo(_criterion.Operands[0]) && _itemsArray[i - 2].BelongsTo(_criterion.Operands[1]) &&
_itemsArray[i - 3].BelongsTo(_criterion.Operands[2]) && _itemsArray[i - 4].BelongsTo(_criterion.Operands[3]) && _itemsArray[i
-5].BelongsTo(_criterion.Operands[4]))
                {
returnnewPSOData() { Data = _itemsArray[i].Data, T1 = _itemsArray[i - 1].Data, T2 = _itemsArray[i - 2].Data, T3 = _itemsArray[i -
3].Data, T4 = _itemsArray[i - 4].Data, T5 = _itemsArray[i - 5].Data };
                }

            }

        }

returnnull;
      }

   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AppFramework
```

```
{
publicclassPSOItem
   {
public PSOItem()
      {

      }

publicstring FuzzySet { get; set; }
publicfloat Data { get; set; }

publicbool BelongsTo(string _fuzzySet)
      {

if ((_fuzzySet == string.Empty)) returntrue;

if (_fuzzySet.ToUpper().Trim() == this.FuzzySet.ToUpper().Trim())
         {
returntrue;
         }

returnfalse;
      }

publicoverridestring ToString()
      {
return"Data: " + this.Data + ", FuzzySet: " + this.FuzzySet;
      }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AppFramework
{
publicclassRule
   {
public Rule(short _index, string[] _operands)
      {
this.IndexNo = _index;
this.Operands = _operands;
      }

publicoverridestring ToString()
      {
return"Index Number: " + this.IndexNo + ", " + "Operands: " + Operands[0] + ", " + Operands[1] + ", " + Operands[2] + ", " +
Operands[3] + ", " + Operands[4];
      }

publicshort IndexNo { get; set; }
publicstring[] Operands { get; set; }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Windows.Forms;

namespace DataEditor
{
publicpartialclassCaptureParameters : Form
   {
privateForm _parentFormReference;

public CaptureParameters(Form _parentForm)
      {
         InitializeComponent();
         _txtDataItemCount.LostFocus += LostFocusEventHandler;
         _txtPSORuleCount.LostFocus += LostFocusEventHandler;
         _parentFormReference = _parentForm;

this.FormClosing += (sender, e) => {_parentFormReference.WindowState = FormWindowState.Normal; };
this.Load += (sender, e) => { _txtDataItemCount.Focus(); };

      }

privatevoid _btnShowCreateData_Click(object sender, EventArgs e)
      {

int dataCount = -1;

if (!int.TryParse(_txtDataItemCount.Text, out dataCount))
         {
MessageBox.Show("Please check that the value entered is a number..", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
return;
         }

if (dataCount < 1)
         {
MessageBox.Show("At least one data item is needed.");
return;
         }

newCreateData(_parentFormReference, dataCount).Show();
this.Dispose(); //this will not raise a close event.
      }

privatevoid LostFocusEventHandler(object sender, EventArgs e)
      {
TextBox interim = sender asTextBox;
if (interim != null)
         {
            interim.Text = interim.Text.Trim();
         }
      }

privatevoid _btnShowRules_Click(object sender, EventArgs e)
      {
int ruleCount = -1;

if (!int.TryParse(_txtPSORuleCount.Text, out ruleCount))
         {
MessageBox.Show("Please check that the value entered is a number.","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
return;
         }

if (ruleCount < 1)
         {
```

```
MessageBox.Show("At least one rule is needed.");
return;
            }

newCreateRules(_parentFormReference,ruleCount).Show();
this.Dispose();

//duplicate code things!
        }
    }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;

namespace DataEditor
{
publicpartialclassCreateData : Form
    {
privateForm _parentFormReference;
privateList<float> _dataItems;
privateList<string> _fsgItems;

public CreateData(Form _parentForm, int _dataCount)
        {
            InitializeComponent();

            DataCount = _dataCount;
            _parentFormReference = _parentForm;
this.Load += (sender, e) => { this.Focus(); };

this.FormClosing += (sender, e) => {

if (e.CloseReason == CloseReason.ApplicationExitCall)
                {
                    _parentFormReference.Close();
this.Close();
                }
DialogResult _result = MessageBox.Show("Return to Main page?", "Exit?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

if (_result == System.Windows.Forms.DialogResult.Yes)
                {
                    _parentFormReference.WindowState = FormWindowState.Normal;
                }
else
                {
Application.Exit();
                }
            }; //end anonymous method

            LoadControls();
            _dataItems = newList<float>();
            _fsgItems = newList<string>();

        }
```

```
publicint DataSetNumber { get; set; }

privatevoid LoadControls()
        {
//int x_pos = 5;
int y_pos = 60;


for (int i = 0; i < DataCount; i++)
            {
TextBox currData = newTextBox();
TextBox currFSG = newTextBox();
Label sno = newLabel();

                sno.Text = (i+1) + ".";
                sno.AutoSize = true;
                sno.Location = newPoint(25, y_pos + 2);

                currData.Location = newPoint(55, y_pos);
                currData.Height = 15;
                currData.Width = 40;
                currData.Tag = "float";

                currFSG.Location = newPoint(125, y_pos);
                currFSG.Height = 15;
                currFSG.Width = 40;
                currFSG.Tag = "string";

this.Controls.Add(sno);
this.Controls.Add(currData);
this.Controls.Add(currFSG);


                y_pos += 22;
            }

Button saveContinue = newButton();
            saveContinue.Location = newPoint(55, y_pos + 3);
            saveContinue.Text = "Save and Continue";
            saveContinue.Height = 15;
            saveContinue.Width = 110;
            saveContinue.AutoSize = true;
            saveContinue.Tag = "int";
            saveContinue.Click += SaveAndContinueEventHandler;
this.Controls.Add(saveContinue);
        }

privateint DataCount { get; set; }

privatebool VetUserInput()
        {
bool _allClear = true;
int intDummyVal = -1;
float floatDummyVal = 0.0F;

            _dataItems.Clear();
            _fsgItems.Clear();

foreach (Control x inthis.Controls)
            {

TextBox y = x asTextBox;
```

```
if (y == null) continue;

            y.Text = y.Text.Trim().ToUpper();

if (y.Text == string.Empty)
            {
                _allClear = false;
break;
            }

try
            {
switch ((string)y.Tag)
                {
case"int":
                        intDummyVal = int.Parse(y.Text);
this.DataSetNumber = intDummyVal;
break;
case"string": //this must be for a fuzzySet
                        _allClear = Utilities.TryParseFuzzySet(y.Text);

                        _fsgItems.Add(y.Text);

break;

case"float":
                        floatDummyVal = float.Parse(y.Text);

                        _dataItems.Add(floatDummyVal);

break;
                }
            } //end try

catch
            {
                _allClear = false; //something went wrong somewhere, so a value is wrong
continue;
            }
        }

return _allClear;
    }

privatevoid SaveAndContinueEventHandler(object sender, EventArgs e)
    {
if (VetUserInput())
        {
MessageBox.Show("Dataset vetted successfully.");
string _pathToSave = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
DataUtility _dataUtil = newDataUtility(_pathToSave);

if (_dataUtil.WriteDataItems(_dataItems, _fsgItems, this.DataSetNumber))
            {
MessageBox.Show("Dataset saved to Desktop successfully.");

                _parentFormReference.WindowState = FormWindowState.Normal;
this.Dispose();

            }
else
            {
```

```
MessageBox.Show("Error while saving dataset file. Probably a permissions problem.");
            }
return;

        }//end successful data vetting

MessageBox.Show("Data Set not vetted successfully. Please check your values.");
      }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;

namespace DataEditor
{
publicpartialclassCreateRules : Form
   {
privateForm _parentFormReference;
privateList<string> _ruleSet;

public CreateRules(Form _parentReference, int _ruleCount)
      {
         InitializeComponent();
         RuleCount = _ruleCount;
         _parentFormReference = _parentReference;
         LoadControls();
         _ruleSet = newList<string>();

this.FormClosing += (sender, e) => {

if (e.CloseReason == CloseReason.ApplicationExitCall)
            {
               _parentFormReference.Close();
this.Close();
            }
DialogResult _res = MessageBox.Show("Return to Main Form?", "Question", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

if (_res == System.Windows.Forms.DialogResult.Yes)
            {
               _parentFormReference.WindowState = FormWindowState.Normal;
this.Dispose();
            }
else
            {
Application.Exit();
            }

         };
      }

privateint RuleCount { get; set; }
privateint RuleSetNumber { get; set; }

privatevoid LoadControls()
```

```
      {
//int x_pos = 5;
int y_pos = 60;


for (int i = 0; i < RuleCount; i++)
          {
TextBox firstOperand = newTextBox();
TextBox secondOperand = newTextBox();
TextBox thirdOperand = newTextBox();
TextBox fourthOperand = newTextBox();
TextBox fifthOperand = newTextBox();
Label sno = newLabel();

              sno.Text = (i + 1) + ".";
              sno.AutoSize = true;
              sno.Location = newPoint(25, y_pos + 2);

              firstOperand.Location = newPoint(55, y_pos);
              firstOperand.Height = 15;
              firstOperand.Width = 40;
              firstOperand.Tag = "1st";

              secondOperand.Location = newPoint(100, y_pos);
              secondOperand.Height = 15;
              secondOperand.Width = 40;
              secondOperand.Tag = "2nd";

              thirdOperand.Location = newPoint(145, y_pos);
              thirdOperand.Height = 15;
              thirdOperand.Width = 40;
              thirdOperand.Tag = "3rd";

              fourthOperand.Location = newPoint(190, y_pos);
              fourthOperand.Height = 15;
              fourthOperand.Width = 40;
              fourthOperand.Tag = "4th";

              fifthOperand.Location = newPoint(235, y_pos);
              fifthOperand.Height = 15;
              fifthOperand.Width = 40;
              fifthOperand.Tag = "5th";

this.Controls.Add(sno);
this.Controls.Add(firstOperand);
this.Controls.Add(secondOperand);
this.Controls.Add(thirdOperand);
this.Controls.Add(fourthOperand);
this.Controls.Add(fifthOperand);

              y_pos += 22;
          }

Button saveContinue = newButton();
          saveContinue.Location = newPoint(55, y_pos + 3);
          saveContinue.Text = "Save ruleset";
          saveContinue.Height = 15;
          saveContinue.Width = 110;
          saveContinue.AutoSize = true;
          saveContinue.Tag = "int";
          saveContinue.Click += SaveEventHandler;
this.Controls.Add(saveContinue);
      }
```

```
privatevoid SaveEventHandler(object sender, EventArgs e)
      {
if (!VetRules())
         {
MessageBox.Show("Erroneous values detected. Please revise.");
return;
         }
else
         {
MessageBox.Show("Rules vetted successfully.");
string _pathToSave = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
DataUtility _dataUtil = newDataUtility(_pathToSave);

if (_dataUtil.WriteRules(_ruleSet, this.RuleSetNumber))
            {
MessageBox.Show("Data persisted successfully.","Success",MessageBoxButtons.OK,MessageBoxIcon.Information);
               _parentFormReference.WindowState = FormWindowState.Normal;
this.Dispose();
            }
else
            {
MessageBox.Show("Error while saving dataset file. Probably a permissions problem.");
            }


         }
      }

privatebool VetRules()
      {

bool _allClear = true;
int dummyInt = -1;

         _ruleSet.Clear();

try
         {
foreach (Control x inthis.Controls)
            {
TextBox curr = x asTextBox;
if (curr == null) continue;

               curr.Text = curr.Text.Trim().ToUpper();

switch ((string)curr.Tag) {

case"int":
if (curr.Text == string.Empty)
                  {
                     _allClear = false;
                  }
                  dummyInt = int.Parse(curr.Text);
this.RuleSetNumber = dummyInt;
break;

case"1st":
if (curr.Text == string.Empty || !Utilities.TryParseFuzzySet(curr.Text))
                  {
                     _allClear = false;
break;
                  }
```

```
                    _ruleSet.Add(curr.Text);
break;

case"2nd":
if (curr.Text == string.Empty || !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    _ruleSet.Add(curr.Text);
break;

case"3rd":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    _ruleSet.Add(curr.Text);
break;

case"4th":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    _ruleSet.Add(curr.Text);
break;

case"5th":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    _ruleSet.Add(curr.Text);
break;
                }//end switch
//if the code reaches here, there were no interruptions

                } //end of foreach

            }

catch
            {
                _allClear = false;
            }

return _allClear;
        }

privatevoid CreateRules_Load(object sender, EventArgs e)
        {

        }

privatevoid label2_Click(object sender, EventArgs e)
        {

        }
```

```
privatevoid label3_Click(object sender, EventArgs e)
      {

      }

privatevoid label4_Click(object sender, EventArgs e)
      {

      }

privatevoid label5_Click(object sender, EventArgs e)
      {

      }

privatevoid label6_Click(object sender, EventArgs e)
      {

      }

privatevoid label1_Click(object sender, EventArgs e)
      {

      }

privatevoid _txtRuleSetNumber_TextChanged(object sender, EventArgs e)
      {

      }


   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;

namespace DataEditor
{
publicpartialclassEditData : Form
   {

privateList<PSOItem> DSrc { get; set; }
privateint DataSetNumber { get; set; }

privateList<float> _dataItems;
privateList<string> _fsgItems;

public EditData(string safeFileName, List<PSOItem> dataSource)
      {
         InitializeComponent();
this.DSrc = dataSource;
this.DataSetNumber = int.Parse(safeFileName.Replace(".dataset", string.Empty));
         _lblCurrEditing.Text += this.DataSetNumber;

         LoadElements();
```

```
            _dataItems = newList<float>();
            _fsgItems = newList<string>();
        }

privatevoid LoadElements()
        {
//int x_pos = 5;
int y_pos = 60;
PSOItem[] arrayVersion = DSrc.ToArray();

for (int i = 0; i < arrayVersion.Length; i++)
            {
TextBox currData = newTextBox();
TextBox currFSG = newTextBox();
Label sno = newLabel();

                sno.Text = (i + 1) + ".";
                sno.AutoSize = true;
                sno.Location = newPoint(25, y_pos + 2);

                currData.Location = newPoint(55, y_pos);
                currData.Height = 15;
                currData.Width = 40;
                currData.Text = arrayVersion[i].Data.ToString() ;
                currData.Tag = "float";

                currFSG.Location = newPoint(125, y_pos);
                currFSG.Height = 15;
                currFSG.Width = 40;
                currFSG.Text = arrayVersion[i].FuzzySet;
                currFSG.Tag = "string";

this.Controls.Add(sno);
this.Controls.Add(currData);
this.Controls.Add(currFSG);


                y_pos += 22;
            }

Button saveContinue = newButton();
            saveContinue.Location = newPoint(55, y_pos + 3);
            saveContinue.Text = "Save Changes";
            saveContinue.Height = 25;
            saveContinue.Width = 110;
            saveContinue.Click += SaveEventHandler;
this.Controls.Add(saveContinue);
        }

privatevoid SaveEventHandler(object sender, EventArgs e)
        {
if (VetUserInput())
            {
MessageBox.Show("Dataset vetted successfully.");
string _pathToSave = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
DataUtility _dataUtil = newDataUtility(_pathToSave);

if (_dataUtil.WriteDataItems(_dataItems, _fsgItems, this.DataSetNumber))
                {
MessageBox.Show("Dataset saved to Desktop successfully.");
this.Dispose();
```

```
            }
else
            {
MessageBox.Show("Error while saving dataset file. Probably a permissions problem.");
            }
return;

        }//end successful data vetting

MessageBox.Show("Data Set not vetted successfully. Please check your values.");
      }

privatebool VetUserInput()
      {
bool _allClear = true;

float floatDummyVal = 0.0F;

          _dataItems.Clear();
          _fsgItems.Clear();

foreach (Control x inthis.Controls)
          {

TextBox y = x asTextBox;

if (y == null) continue;

              y.Text = y.Text.Trim().ToUpper();

if (y.Text == string.Empty)
              {
                 _allClear = false;
break;
              }

try
              {
switch ((string)y.Tag)
                 {

case"string": //this must be for a fuzzySet
                        _allClear = Utilities.TryParseFuzzySet(y.Text);
                        _fsgItems.Add(y.Text);
break;

case"float":
                        floatDummyVal = float.Parse(y.Text);
                        _dataItems.Add(floatDummyVal);
break;
                 }
              } //end try

catch
              {
                 _allClear = false; //something went wrong somewhere, so a value is wrong
continue;
              }
          }

return _allClear;
      }
   }
```

```
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;

namespace DataEditor
{
publicpartialclassEditRules : Form
   {

privateList<Rule> DSrc { get; set; }
privateList<string> RuleSet { get; set; }
privateint RuleSetNumber { get; set; }

public EditRules(string safeFileName, List<Rule> _dataSource)
      {
         InitializeComponent();
this.RuleSetNumber = int.Parse(safeFileName.Replace(".ruleset", string.Empty));
this.DSrc = _dataSource;
         _lblCurrEdit.Text += this.RuleSetNumber;
         LoadElements();
         RuleSet = newList<string>();

      }

privatevoid LoadElements()
      {
//int x_pos = 5;
int y_pos = 60;
Rule[] rules = this.DSrc.ToArray();


for (int i = 0; i < rules.Length; i++)
         {
TextBox firstOperand = newTextBox();
TextBox secondOperand = newTextBox();
TextBox thirdOperand = newTextBox();
TextBox fourthOperand = newTextBox();
TextBox fifthOperand = newTextBox();
Label sno = newLabel();

            sno.Text = (i + 1) + ".";
            sno.AutoSize = true;
            sno.Location = newPoint(25, y_pos + 2);

            firstOperand.Location = newPoint(55, y_pos);
            firstOperand.Height = 15;
            firstOperand.Width = 40;
            firstOperand.Tag = "1st";
            firstOperand.Text = rules[i].Operands[0];

            secondOperand.Location = newPoint(100, y_pos);
            secondOperand.Height = 15;
            secondOperand.Width = 40;
            secondOperand.Tag = "2nd";
            secondOperand.Text = rules[i].Operands[1];
```

```
            thirdOperand.Location = newPoint(145, y_pos);
            thirdOperand.Height = 15;
            thirdOperand.Width = 40;
            thirdOperand.Tag = "3rd";
            thirdOperand.Text = rules[i].Operands[2];

            fourthOperand.Location = newPoint(145, y_pos);
            fourthOperand.Height = 15;
            fourthOperand.Width = 40;
            fourthOperand.Tag = "4th";
            fourthOperand.Text = rules[i].Operands[3];

            fifthOperand.Location = newPoint(145, y_pos);
            fifthOperand.Height = 15;
            fifthOperand.Width = 40;
            fifthOperand.Tag = "5th";
            fifthOperand.Text = rules[i].Operands[4];


this.Controls.Add(sno);
this.Controls.Add(firstOperand);
this.Controls.Add(secondOperand);
this.Controls.Add(thirdOperand);
this.Controls.Add(fourthOperand);
this.Controls.Add(fifthOperand);


            y_pos += 22;
        }

Button saveContinue = newButton();
        saveContinue.Location = newPoint(55, y_pos + 3);
        saveContinue.Text = "Save Changes";
        saveContinue.Height = 25;
        saveContinue.Width = 110;
       saveContinue.Click += SaveEventHandler;
this.Controls.Add(saveContinue);
    }

privatevoid SaveEventHandler(object sender, EventArgs e)
    {
if (!VetRules())
        {
MessageBox.Show("Erroneous values detected. Please revise.");
return;
        }
else
        {
MessageBox.Show("Rules vetted successfully.");
string _pathToSave = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
DataUtility _dataUtil = newDataUtility(_pathToSave);

if (_dataUtil.WriteRules(RuleSet, this.RuleSetNumber))
            {
MessageBox.Show("Data persisted successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
this.Dispose();
            }
else
            {
MessageBox.Show("Error while saving dataset file. Probably a permissions problem.");
            }
```

```
        }
    }

privatebool VetRules()
    {

bool _allClear = true;

        RuleSet.Clear();

try
        {
foreach (Control x inthis.Controls)
            {
TextBox curr = x asTextBox;
if (curr == null) continue;

                curr.Text = curr.Text.Trim().ToUpper();

switch ((string)curr.Tag)
                {


case"1st":
if (curr.Text == string.Empty || !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    RuleSet.Add(curr.Text);
break;

case"2nd":
if (curr.Text == string.Empty || !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    RuleSet.Add(curr.Text);
break;

case"3rd":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    RuleSet.Add(curr.Text);
break;

case"4th":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
break;
                    }
                    RuleSet.Add(curr.Text);
break;

case"5th":
if (curr.Text != string.Empty && !Utilities.TryParseFuzzySet(curr.Text))
                    {
                        _allClear = false;
```

```
break;
                    }
                    RuleSet.Add(curr.Text);
break;



                }//end switch
//if the code reaches here, there were no interruptions

            } //end of foreach

        }

catch
        {
            _allClear = false;
        }

return _allClear;
    }

privatevoid _lblCurrEdit_Click(object sender, EventArgs e)
    {

    }
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;


//TODO: Cosmetic fixes.
namespace DataEditor
{
publicpartialclassMainForm : Form
  {
public MainForm()
    {
        InitializeComponent();
    }

privatevoid _btnCreateData_Click(object sender, EventArgs e)
    {
//Create a new capturedata form.
newCaptureParameters(this).Show();
this.WindowState = FormWindowState.Minimized;

    }

privatevoid _btnEditData_Click(object sender, EventArgs e)
    {
//Open a file dialog and then attempt to load it into the data boxes.
OpenFileDialog _openDlg = newOpenFileDialog();
        _openDlg.Filter = "Dataset files|*.dataset";
DialogResult _res = _openDlg.ShowDialog();
```

```
if (_res == System.Windows.Forms.DialogResult.OK)
        {
DataUtility _dataU = newDataUtility(_openDlg.FileName);
//Mine the data here.


List<PSOItem> results = _dataU.ReadDataItems();

//foreach (PSOItem s in results)
//{
//    MessageBox.Show(s.Data + ", " + s.FuzzySet);
//}

newEditData(_openDlg.SafeFileName, results).ShowDialog();


          }

       }

privatevoid _btnCreateNewRules_Click(object sender, EventArgs e)
       {
newCaptureParameters(this).Show();
this.WindowState = FormWindowState.Minimized;
       }

privatevoid _btnEditExistingRules_Click(object sender, EventArgs e)
       {
//Open a file dialog and then attempt to load it into the data boxes.
OpenFileDialog _openDlg = newOpenFileDialog();
          _openDlg.Filter = "Ruleset files|*.ruleset";
DialogResult _res = _openDlg.ShowDialog();

if (_res == System.Windows.Forms.DialogResult.OK)
        {
DataUtility _dataU = newDataUtility(_openDlg.FileName);
//Mine the data here.


List<Rule> _results = _dataU.ReadRules();

//foreach (Rule s in _results)
//{
//    MessageBox.Show("Number: " + s.IndexNo + " Operand1: " + s.Operands[0] + " Operand2: " + s.Operands[1] + " Operand3: " +
s.Operands[2] + " Operand4: " + s.Operands[3] + " Operand5: " + s.Operands[4]);
//}

newEditRules(_openDlg.SafeFileName, _results).ShowDialog();

          }

       }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace DataEditor
{
```

```
staticclassProgram
   {
///<summary>
/// The main entry point for the application.
///</summary>
      [STAThread]
staticvoid Main()
      {
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(newMainForm());
      }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace GUIComputer
{
staticclassProgram
   {
///<summary>
/// The main entry point for the application.
///</summary>
      [STAThread]
staticvoid Main()
      {
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(newRuleForm());
      }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AppFramework;
using PSOLib;

usingRule = AppFramework.Rule;

namespace GUIComputer
{
publicpartialclassPSOComputer : Form
   {
privateList<Rule> Rules { get; set; }
privateForm ParentFormReference { get; set; }
privatePSOData CurrentData { get; set; }
privatestring DataSetFileName { get; set; }
privateDataUtility _readUtil;
privatePSOResult PSOResult { get; set; }

privateGUIHelper _helper;
```

```
public PSOComputer(string dataSetFN, List<Rule> _rules, Form _parentForm)
        {
//TODO: Modify ctor to add a selected dataset to load appropriate rulesets, and determine how to render the GUI.
            InitializeComponent();
            ParentFormReference = _parentForm;
            Rules = _rules;
            DataSetFileName = dataSetFN;
            _helper = newGUIHelper();
            _readUtil = newDataUtility(DataSetFileName);
            LoadAvailableRules();

        }

privatevoid Form_Closing(object sender, FormClosingEventArgs e)
        {
DialogResult _res = MessageBox.Show("Return to Dataset selection?", "Return?", MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Question);

switch (_res)
            {
caseDialogResult.Cancel:
//do not close!
                e.Cancel = true;
break;

caseDialogResult.Yes:
                ParentFormReference.WindowState = FormWindowState.Normal;
// this.Dispose();
break;

caseDialogResult.No:
                ParentFormReference.Close();
// this.Dispose();
break;

            }

        }

privatevoid LoadAvailableRules()
        {
//  _cmbRules.Items.Add("-");

foreach (Rule r in Rules)
            {
                _cmbRules.Items.Add("Rule " + r.IndexNo.ToString());
            }
        }

privatevoid Item_Selected(object sender, EventArgs e)
        {
Rule rl = Rules.ElementAt(_cmbRules.SelectedIndex);

            CurrentData = _helper.RetrieveAppropriateItem(rl, _readUtil.ReadDataItems());
// List<PSOItem> _dataItems = _readUtil.ReadDataItems();

            _lblFuzzyOne.Text = "FuzzySet 1: " + rl.Operands[0];
            _lblFuzzyTwo.Text = "FuzzySet 2: " + rl.Operands[1];

if (rl.Operands[2] == string.Empty) //there is no third operand.
            {
                _lblFuzzyThree.Text = "FuzzySet 3: " + "-";
            }
```

```
else
        {
           _lblFuzzyThree.Text = "FuzzySet 3: " + rl.Operands[2];
        }

if (rl.Operands[3] == string.Empty) //there is no fourth operand.
        {
           _lblFuzzyFour.Text = "FuzzySet 4: " + "-";
        }
else
        {
           _lblFuzzyFour.Text = "FuzzySet 4: " + rl.Operands[3];
        }
if (rl.Operands[4] == string.Empty) //there is no fifth operand.
        {
           _lblFuzzyFive.Text = "FuzzySet 5: " + "-";
        }
else
        {
           _lblFuzzyFive.Text = "FuzzySet 5: " + rl.Operands[4];
        }

if (CurrentData == null)
        {
//it did not find data in the dataset matching the rule's criterion
           _lblT1.Text = "T1: " + "Not Found";
           _lblT2.Text = "T2: " + "Not Found";
           _lblT3.Text = "T3: " + "Not Found";
           _lblT4.Text = "T4: " + "Not Found";
           _lblT5.Text = "T5: " + "Not Found";
           LockUserInputs();
        }
else {
        _lblT1.Text = "T1: " + CurrentData.T1.ToString();
        _lblT2.Text = "T2: " + CurrentData.T2.ToString();
if (CurrentData.T3 >= 0) {
           _lblT3.Text = "T3: " + CurrentData.T3.ToString();
             }
else {
           _lblT3.Text = "T3: -";
           }


if (CurrentData.T4 >= 0)
        {
           _lblT4.Text = "T4: " + CurrentData.T4.ToString();
        }
else
        {
           _lblT4.Text = "T4: -";
        }


if (CurrentData.T5 >= 0)
        {
           _lblT5.Text = "T5: " + CurrentData.T5.ToString();
        }
else
        {
           _lblT5.Text = "T5: -";
        }

        FreeUserInputs(CurrentData.T3 >= 0, CurrentData.T4 >= 0, CurrentData.T5 >= 0); //free it if there's data for it
```

```
        }
    }

privatevoid Calculate_Solutions(object sender, EventArgs e)
    {
if (_cmbRules.SelectedIndex < 0)
        {
MessageBox.Show("Please select a rule first.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
return;
        }
//Call the parser method of the GUI helper to ensure there's no garbage data

        #region Disappear
List<TextBox> _textBoxes = newList<TextBox>();
foreach (Control c in _grpParams.Controls)
        {
if (c asTextBox != null)
            {
                _textBoxes.Add(c asTextBox);
            }
        }
        #endregion

if (!_helper.ParseData(_textBoxes))
        {
//Data is invalid
MessageBox.Show("Please ensure that the parameters entered are correct and
valid.","Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
return;
        }

//Data is valid. Let's do this. Merely modify the current PSOData object.
        PrepareForPSO();
//Now, compute and display the result in the window.
this.Cursor = Cursors.WaitCursor;
this.PSOResult = PSOEngine.Compute(CurrentData);
this.Cursor = Cursors.Default;
        WritePSODetails();
    }

privatevoid LockUserInputs()
    {
foreach (Control x in _grpParams.Controls)
        {
if (x asTextBox != null)
            {
                x.Enabled = false;
            }
        }

        _txtResult.Enabled = false;
    }

privatevoid FreeUserInputs(bool freeW3, bool freeW4, bool freeW5)
    {
foreach (Control x in _grpParams.Controls)
        {
if (x asTextBox != null)
            {
                x.Enabled = true;
            }
        }
```

```
if (freeW3)
        {
            _txtW3.Enabled = true;
        }
else
        {
//Lock it?
            _txtW3.Enabled = false;
        }

if (freeW4)
        {
            _txtW4.Enabled = true;
        }
else
        {
//Lock it?
            _txtW4.Enabled = false;
        }

if (freeW5)
        {
            _txtW5.Enabled = true;
        }
else
        {
//Lock it?
            _txtW5.Enabled = false;
        }

        _txtResult.Enabled = true;
    }

privatevoid PrepareForPSO()
    {
        CurrentData.C1 = float.Parse(_txtC1.Text.Trim());
        CurrentData.C2 = float.Parse(_txtC2.Text.Trim());
        CurrentData.EnhancedComputations = (CurrentData.T3 >= 0); //if true, then consider w3
        CurrentData.EnhancedComputations4 = (CurrentData.T4 >= 0); //if true, then consider w4
        CurrentData.EnhancedComputations5 = (CurrentData.T5 >= 0); //if true, then consider w5
        CurrentData.InertialWtCfft = float.Parse(_txtIWC.Text.Trim());
        CurrentData.MinimumSE = ulong.Parse(_txtMinSE.Text.Trim());
        CurrentData.NumberOfIterations = int.Parse(_txtMaxIterations.Text.Trim());
        CurrentData.ParticleCount = int.Parse(_txtParticleCount.Text.Trim());
        CurrentData.W1 = float.Parse(_txtW1.Text.Trim());
        CurrentData.W2 = float.Parse(_txtW2.Text.Trim());
if (CurrentData.EnhancedComputations) //only bother if there is actually even a T3
        {
            CurrentData.W3 = float.Parse(_txtW3.Text.Trim());
        }
if (CurrentData.EnhancedComputations4) //only bother if there is actually even a T4
        {
            CurrentData.W4 = float.Parse(_txtW4.Text.Trim());
        }
if (CurrentData.EnhancedComputations5) //only bother if there is actually even a T5
        {
            CurrentData.W5 = float.Parse(_txtW5.Text.Trim());
        }
    }

privatevoid WritePSODetails()
    {
        _txtResult.Clear();
```

```
List<string> lines = newList<string>();
        lines.Add("Terminated on SE condition: " + this.PSOResult.Successful.ToString());

if (this.PSOResult.BestParticleIndex < 0)
        {
           lines.Add("No best particle was encountered.");
        }
else
        {
           lines.Add("Best Particle: " + (this.PSOResult.BestParticleIndex + 1).ToString());
           lines.Add("Best W1 Value: " + this.PSOResult.Particles[this.PSOResult.BestParticleIndex].CurrentPosition1.ToString());
           lines.Add("Best W2 Value: " + this.PSOResult.Particles[this.PSOResult.BestParticleIndex].CurrentPosition2.ToString());
if (this.CurrentData.EnhancedComputations)
           {
              lines.Add("Best W3 Value: " +
this.PSOResult.Particles[this.PSOResult.BestParticleIndex].CurrentPosition3.ToString());
           }
if (this.CurrentData.EnhancedComputations4)
           {
              lines.Add("Best W4 Value: " +
this.PSOResult.Particles[this.PSOResult.BestParticleIndex].CurrentPosition4.ToString());
           }
if (this.CurrentData.EnhancedComputations5)
           {
              lines.Add("Best W5 Value: " +
this.PSOResult.Particles[this.PSOResult.BestParticleIndex].CurrentPosition5.ToString());
           }
        }
        lines.Add("Best SE value obtained: " + this.PSOResult.SEValue.ToString());
        lines.Add("Iterations used: " + this.PSOResult.IterationCount.ToString());

        lines.Add("Full Particle Details:");
        lines.Add("---------------------------------------");

for (int idx = 0; idx <this.PSOResult.Particles.Length; idx++)
        {

           lines.Add("Particle " + (idx + 1).ToString() + ":");
           lines.Add("Initial V1: " + this.PSOResult.InitialVelocities[idx,0]);
           lines.Add("Initial V2: " + this.PSOResult.InitialVelocities[idx, 1]);

if (this.CurrentData.EnhancedComputations)
           {
              lines.Add("Initial V3: " + this.PSOResult.InitialVelocities[idx, 2]);
           }

if (this.CurrentData.EnhancedComputations4)
           {
              lines.Add("Initial V4: " + this.PSOResult.InitialVelocities[idx, 3]);
           }

if (this.CurrentData.EnhancedComputations5)
           {
              lines.Add("Initial V5: " + this.PSOResult.InitialVelocities[idx, 4]);
           }

           lines.Add("SE after 1st Iteration: " + this.PSOResult.IntermediateSquareErrors[idx].ToString());
           lines.Add("SE at last iteration: " + this.PSOResult.SquaredErrorValues[idx].ToString());
           lines.Add("W1 at last iteration: " + this.PSOResult.Particles[idx].CurrentPosition1.ToString());
           lines.Add("W2 at last iteration: " + this.PSOResult.Particles[idx].CurrentPosition2.ToString());

if (this.CurrentData.EnhancedComputations)
           {
```

```
                lines.Add("W3 at last iteration: " + this.PSOResult.Particles[idx].CurrentPosition3.ToString());
            }

if (this.CurrentData.EnhancedComputations4)
            {
                lines.Add("W4 at last iteration: " + this.PSOResult.Particles[idx].CurrentPosition4.ToString());
            }
if (this.CurrentData.EnhancedComputations5)
            {
                lines.Add("W5 at last iteration: " + this.PSOResult.Particles[idx].CurrentPosition5.ToString());
            }
            lines.Add("----------------------------------------");
        }

        _txtResult.Lines = lines.ToArray();
    }

privatevoid Form_Load(object sender, EventArgs e)
    {
        _txtW1.Focus();
    }

privatevoid _txtResult_TextChanged(object sender, EventArgs e)
    {

    }

privatevoid _lblT3_Click(object sender, EventArgs e)
    {

    }

privatevoid _lblT5_Click(object sender, EventArgs e)
    {

    }
  }
}
////////////////////////////////////////////////////////////////////////////////////////////////////
using System;
using System.Windows.Forms;
using AppFramework;
using System.Collections.Generic;


namespace GUIComputer
{
publicpartialclassRuleForm : Form
  {
privateGUIHelper _helper;
privatestring _path = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";

public RuleForm()
    {
        InitializeComponent();
        _helper = newGUIHelper();
        LoadDataSetFileNames();

    }

publicvoid LoadDataSetFileNames()
    {
string[] fileNames = _helper.GetAvailableDataSets();
```

```
if (fileNames == null || fileNames.Length == 0)
        {
            _listDataSets.Items.Add("No Datasets found.");
return;
        }

foreach (string s in _helper.GetAvailableDataSets())
        {
string f = s.Replace(_path, string.Empty);
            _listDataSets.Items.Add(f);
        }

            _listDataSets.SelectedIndexChanged += Dataset_Selected; //assign if there are items.

      }

privatevoid Dataset_Selected(object sender, EventArgs e)
      {
string __selectedItem = (string)_listDataSets.Items[_listDataSets.SelectedIndex];
int __fileIndex = int.Parse(__selectedItem.Replace(".dataset", string.Empty).Trim());
DialogResult _res = MessageBox.Show("Dataset " + __fileIndex.ToString() + " is selected. Continue?","Load
Rules?",MessageBoxButtons.YesNo,MessageBoxIcon.Question);

if (_res == DialogResult.Yes)
        {
//Load the rules and populate the form accordingly.
//Determine if there are even rules to find.
List<Rule> __associatedRules;

if (_helper.LoadIfRuleExists(__fileIndex, out __associatedRules))
            {
//It existed, so the rules were loaded.
PSOComputer _computerForm = newPSOComputer(_path + __fileIndex + ".dataset",__associatedRules, this);
                _computerForm.Show();
this.WindowState = FormWindowState.Minimized;
return;
            }

MessageBox.Show("No associated ruleset found. Please create and retry.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);

         }

      }

privatevoid _listDataSets_SelectedIndexChanged(object sender, EventArgs e)
      {

      }

privatevoid label1_Click(object sender, EventArgs e)
      {

      }

privatevoid RuleForm_Load(object sender, EventArgs e)
      {

      }
   }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PSOLib;
using AppFramework; //this my namespace
using System.IO;

namespace TemperaturePSO
{
classProgram
   {

privatestaticList<Rule> _PSORules;
privatestaticList<PSOItem> _PSODataItems;

privatestaticbool _runAgain;

staticvoid Main(string[] args)
      {


do
         {

while (!LoadData());
Rule criterion;

while (!SelectRule(out criterion)) ;

            WriteCriterionDetails(criterion);
PSOData _data = SelectItemByCriterion(_PSODataItems,criterion);

if (_data == null)
            {
Console.WriteLine("No data corresponding to such a rule was found. Please revise the data in the dataset.");
Console.ReadLine();
Environment.Exit(0);
            }

            WriteDataItemDetails(_data);
            GetPSOParameters(_data);

Console.WriteLine("Hit the enter key to begin the PSO Process.");
Console.ReadLine();

PSOResult _computationResult;
do
            {
Console.WriteLine("Computing optimal solutions...");
               _computationResult = PSOEngine.Compute(_data);
               WriteResultDetails(_data, _computationResult);

if (_computationResult.Successful == false)
               {
Console.WriteLine("Failed to obtain optimal solutions.Retrying...");
               }
            }
while (!_computationResult.Successful);

Console.WriteLine("Hit the Enter key to continue program execution.");
Console.ReadLine();
            AskFinalQuestion();
```

```
        }
while (_runAgain); //main app loop

    }

privatestaticvoid AskFinalQuestion()
    {
string runAgain = string.Empty;

do//this is the final question loop
        {
Console.WriteLine("Run again? [Y/N]");
            runAgain = Console.ReadLine().Trim().ToUpper();
        }

while (!(runAgain == "Y") && !(runAgain == "N"));

if (runAgain == "Y")
        {
            _runAgain = true;
        }
else
        {
            _runAgain = false;
        }
    }

staticbool LoadData()
    {
string readIn = string.Empty;
int fileNumsToLoad = -1;
bool _result = true;

do
        {
Console.WriteLine("*****************************************************************");
Console.WriteLine("Enter the dataset number you'd like to use, or \"Q\" to exit.");
            readIn = Console.ReadLine();

            readIn = readIn.Trim().ToUpper();
if (readIn == "Q") Environment.Exit(0);
        }
while (!int.TryParse(readIn, out fileNumsToLoad));

string path = Environment.GetEnvironmentVariable("USERPROFILE") + @"\Desktop\Datasets\";
string dataFileName = fileNumsToLoad.ToString() + ".dataset";

if (File.Exists(path + dataFileName))
        {
Console.WriteLine("Found dataset file.");
        }
else
        {
Console.WriteLine("Dataset file not found.");
            _result = false;
return _result;
        }

string ruleFileName = fileNumsToLoad.ToString() + ".ruleset";
if (File.Exists(path + ruleFileName))
        {
Console.WriteLine("Found associated ruleset.");
        }
```

```
else
        {
Console.WriteLine("Ruleset file not found.");
            _result = false;
return _result;
        }

DataUtility _dataUtil = newDataUtility(path + dataFileName);
DataUtility _ruleUtil = newDataUtility(path + ruleFileName);

        _PSODataItems = _dataUtil.ReadDataItems();
        _PSORules = _ruleUtil.ReadRules();

return _result;

    }

staticbool SelectRule(outRule _appropriateRule)
    {

int ruleIndex;
string readIn = string.Empty;
bool result = false;
Console.WriteLine("Number of rules in ruleset file: " + _PSORules.Count);

do
        {
Console.WriteLine("Please enter a rule number to train, or Q to exit.");
            readIn = Console.ReadLine().Trim().ToUpper();
if (readIn == "Q") Environment.Exit(0);
        }
while (!int.TryParse(readIn, out ruleIndex));


if (ruleIndex < 0 || ruleIndex > _PSORules.Count)
        {
            _appropriateRule = null;
return result;
        }
//Now we're sure its a safe value.
        ruleIndex--; // this is now an actual rule index in the wouldbe array.
        _appropriateRule = _PSORules.ToArray()[ruleIndex];
returntrue;
    }

privatestaticvoid WriteCriterionDetails(Rule criterion)
    {
if (criterion.Operands[2] == string.Empty)
        {
for (int i = 0; i < 2; i++)
            {
Console.WriteLine("Operand " + i.ToString() + ": " + criterion.Operands[i]);
            }
        }
if (criterion.Operands[3] == string.Empty)
        {
for (int i = 0; i < 3; i++)
            {
Console.WriteLine("Operand " + i.ToString() + ": " + criterion.Operands[i]);
            }
        }

if (criterion.Operands[4] == string.Empty)
```

```
        {
for (int i = 0; i < 4; i++)
            {
Console.WriteLine("Operand " + i.ToString() + ": " + criterion.Operands[i]);
            }
        }

else
        {
for (int i = 0; i < 5; i++)
            {
Console.WriteLine("Operand " + i.ToString() + ": " + criterion.Operands[i]);
            }
        }
    }

staticPSOData SelectItemByCriterion(List<PSOItem> items, Rule criterion)
    {

PSOItem[] itemsAsArray = items.ToArray();

if (criterion.Operands[2] == string.Empty)
        {
//don't look for a third, fourth and criteria.
for (int i = 2; i < itemsAsArray.Length; i++)
            {
if (itemsAsArray[i - 1].BelongsTo(criterion.Operands[0]) && itemsAsArray[i - 2].BelongsTo(criterion.Operands[1]))
                {
returnnewPSOData() { Data = itemsAsArray[i].Data, T1 = itemsAsArray[i - 1].Data, T2 = itemsAsArray[i - 2].Data , T3 = -100};
                }

            }
        }
if (criterion.Operands[3] == string.Empty)
        {
// look for a third criterion but dont look for fourth and fifth.
for (int i = 3; i < itemsAsArray.Length; i++)
            {
if (itemsAsArray[i - 1].BelongsTo(criterion.Operands[0]) && itemsAsArray[i - 2].BelongsTo(criterion.Operands[1]) &&
itemsAsArray[i-3].BelongsTo(criterion.Operands[2]))
                {
returnnewPSOData() { Data = itemsAsArray[i].Data, T1 = itemsAsArray[i - 1].Data, T2 = itemsAsArray[i - 2].Data, T3 =
itemsAsArray[i-3].Data, T4 = -100 };
                }

            }

        }
if (criterion.Operands[4] == string.Empty)
        {
// look for a fourth criterion but not fifth.
for (int i = 4; i < itemsAsArray.Length; i++)
            {
if (itemsAsArray[i - 1].BelongsTo(criterion.Operands[0]) && itemsAsArray[i - 2].BelongsTo(criterion.Operands[1]) &&
itemsAsArray[i-3].BelongsTo(criterion.Operands[2]) && itemsAsArray[i-4].BelongsTo(criterion.Operands[3]))
                {
returnnewPSOData() { Data = itemsAsArray[i].Data, T1 = itemsAsArray[i - 1].Data, T2 = itemsAsArray[i - 2].Data, T3 =
itemsAsArray[i - 3].Data, T4 = itemsAsArray[i - 4].Data, T5 = -100 };
                }

            }

        }
```

```
else
        {
// look for a fifth criterion.
for (int i = 5; i < itemsAsArray.Length; i++)
            {
if (itemsAsArray[i - 1].BelongsTo(criterion.Operands[0]) && itemsAsArray[i - 2].BelongsTo(criterion.Operands[1]) &&
itemsAsArray[i-3].BelongsTo(criterion.Operands[2]) && itemsAsArray[i-4].BelongsTo(criterion.Operands[3]) && itemsAsArray[i-
5].BelongsTo(criterion.Operands[4]))
                {
returnnewPSOData() { Data = itemsAsArray[i].Data, T1 = itemsAsArray[i - 1].Data, T2 = itemsAsArray[i - 2].Data, T3 =
itemsAsArray[i-3].Data, T4 = itemsAsArray[i-4].Data, T5 = itemsAsArray[i-5].Data };
                }

            }

        }
returnnull;
    }

staticvoid WriteDataItemDetails(PSOData _data)
    {
Console.WriteLine("Data: " + _data.Data);
Console.WriteLine("f(t-1): " + _data.T1);
Console.WriteLine("f(t-2): " + _data.T2);

if ((_data.T3 != -100))
        {
Console.WriteLine("f(t-3): " + _data.T3);
        }

if ((_data.T4 != -100))
        {
Console.WriteLine("f(t-4): " + _data.T4);
        }

if ((_data.T5 != -100))
        {
Console.WriteLine("f(t-5): " + _data.T5);
        }
    }

staticvoid GetPSOParameters(PSOData _data)
    {
string _answer = string.Empty;
bool doAgain = false;

float c1;
float c2;
double iwc;
int numOfIterations;
int particleCount;
ulong mse;
float w1 = 0; ;
float w2 = 0;
float w3 = 0;
float w4 = 0;
float w5 = 0;

do
        {

do
            {
```

```
Console.WriteLine("Please enter a value for the constant C1.");
                _answer = Console.ReadLine();
            }
while (!float.TryParse(_answer, out c1));

do
            {
Console.WriteLine("Please enter a value for the constant C2.");
                _answer = Console.ReadLine();
            }
while (!float.TryParse(_answer, out c2));

do
            {
Console.WriteLine("Please enter a value for the Inertial Weight Coefficient.");
                _answer = Console.ReadLine();
            }
while (!double.TryParse(_answer, out iwc));

do
            {
Console.WriteLine("Please enter the number of iterations required.");
                _answer = Console.ReadLine();
            }
while (!int.TryParse(_answer, out numOfIterations));

do
            {
Console.WriteLine("Please enter the number of particles required.");
                _answer = Console.ReadLine();
            }
while (!int.TryParse(_answer, out particleCount));

do
            {
Console.WriteLine("Please enter the target mean squared error.");
                _answer = Console.ReadLine();
            }
while (!ulong.TryParse(_answer, out mse));

do
            {
Console.WriteLine("Please enter a value for the constant W1.");
                _answer = Console.ReadLine();
            }
while (!float.TryParse(_answer, out w1));

do
            {
Console.WriteLine("Please enter a value for the constant W2.");
                _answer = Console.ReadLine();
            }
while (!float.TryParse(_answer, out w2));

if (_data.T3 != -100)
            {
do
                {
Console.WriteLine("Please enter a value for the constant W3.");
                    _answer = Console.ReadLine();
                }
while (!float.TryParse(_answer, out w3));
            }
```

```
if (_data.T4 != -100)
            {
do
                {
Console.WriteLine("Please enter a value for the constant W4.");
                    _answer = Console.ReadLine();
                }
while (!float.TryParse(_answer, out w4));
            }

if (_data.T5 != -100)
            {
do
                {
Console.WriteLine("Please enter a value for the constant W5.");
                    _answer = Console.ReadLine();
                }
while (!float.TryParse(_answer, out w5));
            }

            _data.C1 = c1;
            _data.C2 = c2;
            _data.InertialWtCfft = iwc;
            _data.ParticleCount = particleCount;
            _data.NumberOfIterations = numOfIterations;
            _data.MinimumSE = mse;
            _data.W1 = w1;
            _data.W2 = w2;

if (_data.T3 != -100)
            {
                _data.EnhancedComputations = true;
                _data.W3 = w3;
            }

if (_data.T4 != -100)
            {
                _data.EnhancedComputations4 = true;
                _data.W4 = w4;
            }

if (_data.T5 != -100)
            {
                _data.EnhancedComputations5 = true;
                _data.W5 = w5;
            }


Console.WriteLine("C1: " + _data.C1);
Console.WriteLine("C2: " + _data.C2);
Console.WriteLine("Inertial Weight Coefficient: " + _data.InertialWtCfft);
Console.WriteLine("Particle Count: " + _data.ParticleCount);
Console.WriteLine("Iteration Count: " + _data.NumberOfIterations);
Console.WriteLine("Min. Squared Error: " + _data.MinimumSE);
Console.WriteLine("W1: " + _data.W1);
Console.WriteLine("W2: " + _data.W2);

if (_data.T3 != -100)
            {
Console.WriteLine("W3: " + _data.W3);
            }
```

```
if (_data.T4 != -100)
            {
Console.WriteLine("W4: " + _data.W4);
            }

if (_data.T5 != -100)
            {
Console.WriteLine("W5: " + _data.W5);
            }

string runAgain = string.Empty;
do
            {
Console.WriteLine("Are the above values satisfactory? [Y/N]");
               runAgain = Console.ReadLine().Trim().ToUpper();
            }

while (!(runAgain == "Y") && !(runAgain == "N"));

if (runAgain == "Y")
            {
               doAgain = false;
            }
else
            {
               doAgain = true;
            }


         } //main loop
while (doAgain);
      }

staticvoid WriteResultDetails(PSOData _data, PSOResult _res)
      {
Console.WriteLine("Results obtained: ");

for (int i = 0; i < _data.ParticleCount; i++)
         {
//Loop through the results
Console.WriteLine("Particle " + (i + 1).ToString() + ": ");
Console.WriteLine("W1=" + _res.Particles[i].CurrentPosition1);
Console.WriteLine("W2=" + _res.Particles[i].CurrentPosition2);
if (_data.T3 != -100)
            {
Console.WriteLine("W3=" + _res.Particles[i].CurrentPosition3);
            }

if (_data.T4 != -100)
            {
Console.WriteLine("W4=" + _res.Particles[i].CurrentPosition4);
            }

if (_data.T5 != -100)
            {
Console.WriteLine("W5=" + _res.Particles[i].CurrentPosition5);
            }

Console.WriteLine("Initial V1=" + _res.InitialVelocities[i,0]);
Console.WriteLine("Initial V2=" + _res.InitialVelocities[i,1]);
if (_data.T3 != -100)
            {
Console.WriteLine("Initial V3=" + _res.InitialVelocities[i,2]);
```

```
            }
if (_data.T4 != -100)
            {
Console.WriteLine("Initial V4=" + _res.InitialVelocities[i, 3]);
            }
if (_data.T5 != -100)
            {
Console.WriteLine("Initial V5=" + _res.InitialVelocities[i, 4]);
            }
Console.WriteLine("Intermediate Squared Error after 1st Iteration: " + _res.IntermediateSquareErrors[i].ToString());

Console.WriteLine("Squared Error Value at completion: " + _res.SquaredErrorValues[i].ToString());
        }
Console.WriteLine();
Console.WriteLine("The best particle was: Particle " + (_res.BestParticleIndex + 1).ToString());
Console.WriteLine("Squared Error Value: " + _res.SEValue);
Console.WriteLine("Iteration Count: " + _res.IterationCount);

if (_res.BestParticleIndex >= 0)
        {
//Then there was indeed a "best" particle. Sometimes there really isn't one.
Console.WriteLine("Best W1 Value: " + _res.Particles[_res.BestParticleIndex].CurrentPosition1);
Console.WriteLine("Best W2 Value: " + _res.Particles[_res.BestParticleIndex].CurrentPosition2);
if (_data.T3 != -100)
            {
Console.WriteLine("Best W3 Value: " + _res.Particles[_res.BestParticleIndex].CurrentPosition3);
            }
if (_data.T4 != -100)
            {
Console.WriteLine("Best W4 Value: " + _res.Particles[_res.BestParticleIndex].CurrentPosition4);
            }

if (_data.T5 != -100)
            {
Console.WriteLine("Best W5 Value: " + _res.Particles[_res.BestParticleIndex].CurrentPosition5);
            }
        }

    }
  }

}
```