A GENETIC - FUZZY SYSTEM MODELING OF TRIP DISTRIBUTION

Mert KOMPİL

İzmir Institute of Technology

December, 2010

A GENETIC - FUZZY SYSTEM MODELING OF TRIP DISTRIBUTION

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPY

in City Planning

by
Mert KOMPİL

December 2010
İZMİR

We approve the thesis of **Mert KOMPİL**

Assoc. Prof. Dr. H. Murat ÇELİK
Supervisor

Prof. Dr. Cemal Arkon
Committee Member

Assoc. Prof. Dr. Semahat Özdemir
Committee Member

Assoc. Prof. Dr. K. Mert ÇUBUKÇU
Committee Member

Assoc. Prof. Dr. Serhan TANYEL
Committee Member

20 December 2010

Assoc. Prof. Dr. Semahat ÖZDEMİR
Head of the Department of City and Regional Planning

Prof. Dr. Sedat Akkurt
Dean of the Graduate School of Engineering and Sciences

to my beloved father Salih Kompil
and beloved mother Nezahat Kompil,
who have trusted and supported me all the time

bana her zaman inanmış ve desteklemiş,
sevgili babam Salih Kompil
ve sevgili annem Nezahat Kompil'e

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my supervisor, Assoc. Prof. Dr. H. Murat Çelik for his continuous encouragement and support. This work would not have been possible without his wide knowledge and personal guidance. I also owe my deepest gratitude to him that his thought and ideals have had remarkable influence on me.

I am also deeply grateful to Prof. Dr. Cemal Arkon and Assoc. Prof. Dr Semahat Özdemir, for their valuable contributions to the present work, and for their unfailing support and encouragement throughout my graduate life.

I wish to express my warm and sincere thanks to Assoc. Prof. Dr. K. Mert Çubukçu, for his constructive comments with this work, and for his important support throughout my PhD. I would also like to show my gratitude to Assoc. Prof. Dr. Serhan Tanyel, for his valuable comments and suggestions.

I am deeply grateful to Istanbul Metropolitan Municipality Transportation Planning Department. This thesis would not have been possible unless their permission to use Istanbul Household Travel Survey data.

I am also indebted to the members of Izmir Institute of Technology and especially to the members of Department of City and Regional Planning with whom I have interacted during the course of my graduate studies. I also wish to express my warmest thanks to all of my colleagues and friends at the Department of City and Regional Planning, for their continuous moral support and rich friendship.

Last, but not least, I am truly indebted and forever thankful to my family for their unfailing encouragement and support in all respects. I would like show my warmest thanks to Ramazan İnce and Necla İnce, and to Mutlu Kompil Yaldız and Mahmut Yaldız, who were always nearby when I needed. I am also forever indebted to my beloved mother Nezahat Kompil, who has inspired, encouraged and supported me throughout my whole life. Finally and most importantly, I am deeply grateful to my lovely wife Esin İnce Kompil. The completion of this thesis would not have been possible without her precious support and help. I would like to give my special thanks to her and to my sweet son Oğuz Kompil, whose love make me the happiest man in the world.

ABSTRACT

A GENETIC - FUZZY SYSTEM MODELLING OF TRIP DISTRIBUTION

Trip distribution modelling is one of the most active parts of travel demand analysis. In recent years, use of soft computing techniques has introduced effective modelling approaches to the trip distribution problem. Fuzzy Rule-Based System (FRBS) and Genetic Fuzzy Rule-Based System (GFRBS: fuzzy system improved by a knowledge base learning process with genetic algorithms) modelling of trip distribution are two of these new approaches. However, much of the potential of these techniques has not been demonstrated so far. The present study explores the potential capabilities of these approaches in an urban trip distribution problem with some new features. For this purpose, a simple FRBS and a novel GFRBS were designed to model Istanbul intra-city passenger flows. Subsequently, their accuracy, applicability, and generalizability characteristics were evaluated against the well-known gravity and neural networks based trip distribution models. The overall results show that: i) traditional doubly constrained gravity models are still simple and efficient; ii) neural networks may not show expected performance when they are forced to satisfy production-attraction constraints; iii) simply-designed FRBSs, learning from observations and expertise, are both interpretable and efficient in forecasting trip interchanges even if the data is large and noisy; and iv) use of genetic algorithms in fuzzy rule base learning considerably increases modelling performance, although it brings additional computation costs.

ÖZET

BİR GENETİK - BULANIK SİSTEM ÖNERİSİ İLE SEYAHAT DAĞILIMI MODELLEMESİ

Geçmişten günümüze, seyahat dağılım modelleri, ulaşım talep analizinin en aktif kısımlarından biri olagelmiştir. Son yıllarda, hesaplamalı zeka tabanlı tekniklerin kullanımı, klasik seyahat dağılımı problemine yeni ve etkin çözümler getirmiştir. Bulanık Kural Tabanlı Sistemler (BKTS) ve Genetik Bulanık Tabanlı Sistemler (GBKTS: bulanık kural tabanının genetik algoritmalar yardımıyla öğrenildiği ya da iyileştirildiği sistemler) bu yeni yaklaşımlardan ikisidir. Ancak, bu iki yaklaşımın seyahat dağılımı modellemesindeki potansiyelleri bugüne kadar gerçek anlamda ortaya konamamıştır. Bu çalışmada, kentsel seyahat dağılımı problemine bu iki yaklaşımın potansiyel uygulanabilirliği araştırılmaktadır. Bu amaçla öncelikle, İstanbul sehir-içi seyahat dağılımını modellemek üzere basit bir BKTS ve orijinal bir GKBS tasarlanmıştır. Daha sonra tasarlanan bu modellerin doğruluğu, uygulanabilirliği ve genellenebilirliği gibi özellikleri, yaygın olarak kullanılan gravite ve sinir ağları tabanlı seyahat dağılımı modelleri ile kıyaslanmıştır. Çalışma ile ulaşılan sonuçları özetlemek gerekirse: i) klasik çift-kısıtlı gravite modelleri halen basit ve etkin modellerdir; ii) sinir ağı tabanlı modeller, üretim-çekim kısıtlarının yerine getirilmesi söz konusu olduğunda beklenen performansı göstermeyebilmektedir; iii) gözlemlerden ve uzman deneyiminden öğrenen, basitçe tasarlanmış BKTS'ler, hem yorumlanabilir hem de seyahat değişimlerini tahminlemede oldukça başarılıdır. Bu durum veri seti geniş ve yer yer problemli olsa bile geçerlidir; iv) hesaplama veya modelleme zorlukları içerse de, bulanık kural sisteminin oluşturulmasında genetik algoritmaların kullanımı modelleme performansını ciddi ölçüde arttırmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ARAE	Average Relative Absolute Error
ARV	Average Relative Variance
DCGM	Doubly-Constrained Gravity Model
FL	Fuzzy Logic
FRBS	Fuzzy Rule-Based Systems
GAs	Genetic Algorithms
GFRBS	Genetic Fuzzy Rule-Based Systems
GFS	Genetic Fuzzy Systems
GIS	Geographical Information Systems
HBO	Home-Based-Other
HBS	Home-Based-School
HBW	Home-Based-Work
MF	Membership Function
ML	Maximum Likelihood
MSE	Mean Square Error
MTCE	Mean Travel Cost Error
NHB	None-Home-Based
NNs	Neural Networks
NNTDM	Neural Network Based Trip Distribution Model
O-D	Origin-Destination
P-A	Production-Attraction
RMSE	Root Mean Square Error
SRMSE	Standardized Root Mean Square Error
TAZ	Traffic Analysis Zone
TLD	Trip Length Distribution
WLS	Weighted Least Squares

CHAPTER 1

INTRODUCTION

Travel demand modelling is crucial in transportation planning. Over several decades, a number of techniques have been proposed for each steps of the demand analysis to achieve more accurate and applicable solutions. Among them, trip distribution has probably been the most attracted field of travel demand analysis especially with the widespread use of gravity type of spatial interaction models.

Simply for a given trip purpose, any trip distribution model estimates the trips between given origins and destinations. From the early 1950s, modellers have used several different formulations to deal with this task. The gravity models in the mid-1950s and many other aggregate or disaggregate models have followed the initial growth factor techniques, such as the intervening opportunity models, random utility models, activity based models and several mixed and combined models. All models have a theoretical basing mainly on physics, statistics, economics and behavioural sciences.

With its well-known theoretical base and various application procedures, the gravity type of spatial interaction models have been by far the most commonly used aggregate trip distribution models. Recently, there has been increasing interest among both transportation researchers and practitioners in exploring the capability of computational intelligence based techniques to real transportation problems.

Research in more effective and predictive methodologies in spatial interaction and trip distribution modelling has also led to some pioneering studies in this area. Many scholars have proposed new modelling procedures to forecast aggregate interactions using Neural Networks (Openshaw, 1993; Fischer and Gopal, 1994; Black, 1995; Mozolin et al., 2000; Celik, 2004; Tillema et al., 2006; Tapkin and Ozdemir, 2009), Fuzzy Logic (Kalic and Teodorovic, 1996; 2003; Shafahi et al., 2008) and Evolutionary or Genetic Algorithms (Diplock and Openshaw, 1996; Leung, 2007).

The initial experiences with these techniques have been encouraging and the overall results offer that the Neural Networks (NNs), Fuzzy Logic (FL) and Genetic Algorithms (GAs) can successfully be used in spatial interaction models. Additionally,

they can produce more accurate results against the conventional models even though the efficiency, accuracy, applicability and interpretability of these approaches are still under investigation.

From our point of view, use of fuzzy set theory and FL is very promising in modelling spatial interactions for several reasons: i) they are simple, flexible, and equation-free; ii) they can be used under uncertain and imprecise conditions; iii) they provide an opportunity to incorporate expertise into modelling procedure, a process that may increase the interpretability of the analysed system; and iii) they are capable of increasing the accuracy of analysed system when hybridized with GAs or NNs.

A pioneering fuzzy logic approach to trip distribution modelling was introduced by Kalic and Teodorovic (1996; 2003). They (1996) estimated air passenger flows among some major industrial cities and tourist resorts using known productions and attractions as inputs. In comparison to non-fuzzy methods, the proposed Fuzzy Rule-Based System (FRBS) produced better results. In another study, they (2003) achieved better results using a Genetic Fuzzy Rule-Based System (GFRBS: fuzzy systems augmented by a learning process based on genetic algorithms search). Finally, Shafahi et. al. (2008) proposed a FRBS to predict the discretionary trips in Tahran showing its capability in predicting intra-city passenger flows. They used travel time as additional input and gained better results against the unconstrained type of gravity model.

However, in comparison with the approaches centred on gravity and NNs, the full potential of FRBSs and GFRBSs has still not been demonstrated in trip distribution modelling. Their efficiency, accuracy, applicability, and interpretability are still under investigation. In particular, applicability of GFRBSs to the estimation of intra-city passenger flows had not been investigated earlier. They offer high-quality predictions, but their computational challenges with an additional friction variable are not known.

With this background, this study attempts to set out an FRBS and a GFRBS for modelling intra-city passenger flows in Istanbul. Our primary interest is to contribute to the knowledge and literature on the feasibility of using such models for urban trip distribution modelling. Another objective is to compare and evaluate the accuracy, applicability, and generalizability of such models against well-known trip distribution models in a complex real-world case. For this purpose, a doubly-constrained gravity model and a multilayer feed-forward NNs based trip distribution model were established as the benchmarks, and model performances were evaluated empirically using the 2006 Istanbul Travel Survey data.

The thesis starts with an overall explanation of traditional travel demand analysis and four-step modelling process. Then a brief review on aggregate trip distribution models are given with the descriptions and mathematical expressions. The main focus is presented on the gravity type of trip distribution model and its doubly-constrained type as it has been selected as a benchmark model.

The third chapter consists of soft computing techniques and their applications in transport modelling and trip distribution analysis. First, basic components of NNs, FL and GAs are introduced, then a literature review on their application to spatial interaction and trip distribution modelling is summarized in the chapter.

A general explanation of the FRBSs and GFRBSs are given in the fourth chapter. It also introduces main properties of fuzzy set theory and GAs.

The fifth chapter includes the whole empirical analysis. Description of the study area and data sets are included in this chapter. Additionally, the fifth chapter includes calibration, training, learning and implementation issues of the applied trip distribution models. The performance measure and goodness-of-fit statistics used in the thesis are also explained here.

Chapter six includes results and empirical findings of the all applied trip distribution models. Finally, summary findings and model evaluation of the study along with further research possibilities are discussed and evaluated in the conclusion chapter.

CHAPTER 2

TRADITIONAL TRIP DISTRIBUTION MODELS

2.1. Travel Demand Analysis and The Four-Step Demand Modelling

Urban transportation planning process takes community needs and expectations into consideration and establishes a way of designing future transportation systems. One of the most important stage of transportation planning process is forecasting future travel demand in a desired level of accuracy. Starting from the middle of the twentieth century, progressive researches in this area has led travel demand modelling into a well-designed modelling methodology. A number of deterministic and stochastic models have been developed to understand travel behaviour better and to achieve more accurate forecasts.

The traditional urban travel demand modelling consists of a sequential procedure often referred as the 'four-step' modelling process: trip generation, trip distribution, mode choice and trip assignment. In order to identify possible transportation system needs and required changes, these sub-models forecast future travel demand using existing transportation system and base year travel demand information.

Meyer and Miller (2001, p.270) describe the basic assumption behind the four-step model as *"…in a sequential decision process, people decide to make a trip (generation), decide where to go (distribution), decide what mode to take (modal split), and decide what route to use (assignment)"* and make brief definition of each steps as:

- Trip generation is the prediction of the number of trips produced by and attracted to each zone, that is, the number of trips ends "generated" within the urban area.
- Trip distribution is the prediction of origin-destination (O-D) flows, that is, the linking of the trip ends predicted by the trip generation model together to form trip interchanges or flows.
- Modal split models predict the percentages of travel flow that will use each of the available modes (auto, transit, walk, etc.) between each origin-destination pair.
- Trip assignment places the O-D flows for each mode on specific routes of travel through the respective modal networks (Meyer and Miller, 2001, p.270).

The origins of four-step modelling procedure go back to the 1950s to the comprehensive transportation plans of large North American Cities. As being known the pioneering urban transportation study, *Chicago Area Transportation Study* (1959) contains a diagram showing the travel forecasting phase of the planning process consisting of three steps: i) estimate trip generation, ii) estimate trip distribution, iii) estimate future travel demand. The concepts of modal split and traffic assignment were introduced in the second part of the Chicago study (Boyce, 2002).

After the initial studies in North America, the transportation planning process became institutionalized with the *Federal Aid Highway Act* (1962) and standardized by the codification of all technical aspects of planning process in a series of technical manual (Taaffe et al., 1996). First formed in these studies, the four-step modelling procedure is in use over forty years with its main framework. A general form of four-step travel demand modelling procedure can be shown as in Figure 2.1.

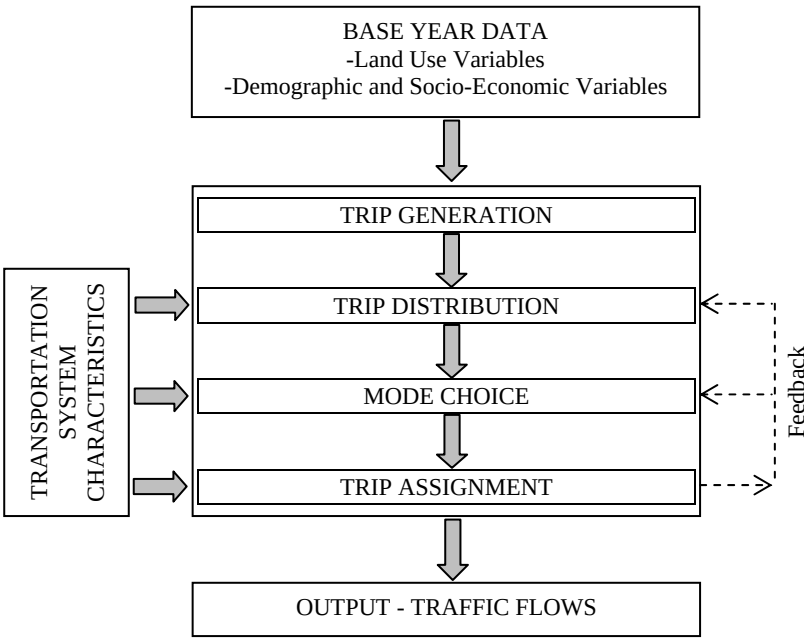

Figure 2.1. Four-Step Travel Demand Modelling Process

Martin et al. (1961) prepared an early review of urban travel forecasting methods including all features of data collection, plan formulation, travel forecasting, and also four-step modelling procedure. During the period of 1965-1980, many academic researchers began to investigate improved model formulations for the individual steps of the sequential procedure. For instance, Wilson (1967; 1970)

proposed the entropy-maximizing formulation of the trip distribution model. Furthermore, he linked the mode choice and residential location models. McFadden (1974) derived the logit mode choice model based on random utility theory and Williams (1977) produced his seminal treatment of nested logit models (Boyce, 2002).

Researchers have continued their studies in a slowing rate especially to increase predictive capabilities of each individual steps of sequential procedure with aggregate or disaggregate approaches after 1980s. Extensive reviews on some important models on travel demand forecasting can be seen in the works of Ortuzar and Willumsen (2001), Meyer and Miller (2001), Taaffe et al. (1996) and Oppenheim (1995).

2.2. Fundamentals of Trip Distribution Analysis

Trip distribution analysis has always been one of the most active sector of travel demand modelling process. Simply for a given trip purpose, any trip distribution model estimates the trip interactions between given origins and destinations. Considering a zone system in any city or a region, the starting point of modelling is to build an production-attraction and a friction matrix related to the zone system. Such matrixes are shown in Table 2.1. Between m origins and n destinations, there is a set of flows as in trip matrix, and a matched set of spatial separations as in friction matrix.

Table 2.1. Trip and Friction Matrixes with Notations

Trip Matrix		**Attraction Zone**			
		1	2	...	n
Production Zone	1	T_{11}	T_{12}	...	T_{1n}
	2	T_{21}	T_{22}	...	T_{2n}
	...	...	...	...	...
	m	T_{m1}	T_{m2}	...	T_{mn}

Friction Matrix		**Attraction Zone**			
		1	2	...	n
Production Zone	1	C_{11}	C_{12}	...	C_{1n}
	2	C_{21}	C_{22}	...	C_{2n}
	...	...	...	...	...
	m	C_{m1}	C_{m2}	...	C_{mn}

The row sum of trip matrix represents the total number of trips originated from zone *i*, and P_i stands for the production total of zone *i* (Eq. 2.1) . The column sum represents the total number of trips destinated from zone *j*, and A_j stands for the

attraction total of zone j (Eq. 2.2). Finally, sum of all interactions represents total number of trips T, which also equals to the production and attraction totals (Eq. 2.3).

$$P_i = \sum_j T_{ij} \tag{2.1}$$

$$A_j = \sum_i T_{ij} \tag{2.2}$$

$$\sum_{ij} T_{ij} = \sum_i P_i = \sum_j A_j \tag{2.3}$$

All compatible cells (C_{ij}s) in the friction matrix represents some measure of spatial impedance. Generally three types of measures dominate the literature: physical distance, travel cost and travel time. Considering the above matrices as base year data, observed distributions are to be estimated using some appropriate functions and parameters. If a desired level of accuracy is satisfied with the estimations, the same function and/or parameters can be used to predict projection year trip table. The output of trip generation phase becomes input for those predictions. A general form of a trip distribution model can be written as follows:

$$T_{ij} = f\left(P_i A_j C_{ij}\right) \tag{2.4}$$

where T_{ij} stands for the estimated trips between zones i and j; P_i and A_j represent some function of trip production and attraction potentials of zone i and zone j; and C_{ij} stands for the general travel cost between zone i and zone j. In some cases additional socioeconomic characteristics could be considered as zone based inputs.

However, most of the modelling efforts suggest use of trip production and attraction totals solely. These two variables comprise and represent many other factors and are convenient with the trip generation outputs. It is also preferable to use either a production-attraction (PA) or an origin-destination (OD) matrix depending on purpose of the model. The trip distribution of the traditional four-step modelling process deals generally with a PA matrix, where an OD matrix is required for directional traffic assignment.

2.3. Trip Distribution Models

2.3.1. An Overview

From the early 1950s, modellers have used several different formulations to deal with distribution of trips between given origins and destinations. Initial approaches expanded known distributions with growth factors to forecast future patterns. The gravity models in the mid-1950s and many other models aggregate or disaggregate have followed the initial techniques, such as the intervening opportunities models, random utility models, activity based models and several mixed and combined models.

All above mentioned models have a theoretical base mainly on physics, statistics, economics or behavioural sciences. Today, the main classification of theoretical trip distribution models can be formed as it is shown in Table 2.2. As seen in the table, trip distribution models can be divided into two broad categories : aggregate and disaggregate.

Table 2.2. Classification of Theoretical Trip Distribution Models
(Source: Cascetta et al., 2007, p.601)

Aggregate Approach	**Disaggregate Approach**
Growth Factor Methods Gravity/Spatial Interaction Models Intervening Opportunities Models	Random Utility Models
Mixed Models	
Gravity-opportunities Models	Random Utility Models with Intervening Opportunities

Aggregate models analyze total number of trips between each traffic analysis zones and can be further classified into growth-factor methods, gravity or spatial interaction models and intervening opportunities models. On the other hand, disaggregate models such as logit and activity-based models deal with individuals' behaviours and destination choices.

Present work deals with aggregate models of trip distribution, so the following sections give essentials of well-known aggregate models. Much of the focus is given on gravity type of spatial interaction models due to its proved effectiveness and widespread usage. More on theoretical models and extensive reviews on either aggregate or disaggregate models can be seen in the works of Ortuzar and Willumsen (2001), Easa (1993), Black (2003), Kanafani (1983), Oppenheim (1995), and Cascetta et al. (2007).

2.3.2. The Growth-Factor Methods

Growth-factor methods are initial modelling techniques especially used in the early transportation plans mentioned earlier. The first applications in the 1950-1960s took origin-destination flows of large household surveys and expand known distributions with some growth factors.

The Uniform model, Fratar model , Detroit model and Furness model were the significant methods in this respect. Today, they are especially used to update a trip matrix for short term forecasts. According to the available information, Ortuzar and Willumsen (2001) distinguishes growth-factor methods into three: Uniform, Singly-Constrained and Doubly-Constrained Growth-Factor Techniques.

2.3.2.1. Uniform Growth-Factor Technique

If only information on overall future trip rates or growth factors are available, an old or present trip matrix can simply be expanded with the uniform growth-factor technique. First, a single-factor is computed for the whole study area (eq. 2.5), and then each cells in the trip matrix are expanded using the computed growth rate (eq. 2.6) as:

$$F = T/T^0 \tag{2.5}$$

$$T_{ij} = FT_{ij}^0 \tag{2.6}$$

where, F is the ratio of total number of future trips, T, over observed total number of trips T^0, and T_{ij} is the future and T_{ij}^0 is the observed trips from zone i to zone j.

2.3.2.2. Singly-Constrained Growth-Factor Technique

If there is likely information on total number of trips originating or attracted to each zones, it would be possible to apply origin-specific or destination-specific growth factors. The computation steps take the same form of previously established equations 2.5 and 2.6. The only difference is that the total number of trips are replaced with the origin or destination totals of zones. With this replacement any trip interaction can be computed with the growth rate of corresponding rows or columns in the trip matrix.

2.3.2.3. Doubly-Constrained Growth-Factor Technique

In most cases of trip distribution analysis, both originating and attracted trip totals are known collectively as an output of trip generation step. In such cases, there should be a row factor F_i and a column factor F_j when expanding the interaction T_{ij} between zone *i* and zone j. However, there is still a need to use additional correction factors to satisfy trip total constraints iteratively. Furness (1965) propose, in this manner is one of the best known iterative method with 'balancing factors' *Ai* and *Bj* as:

$$T_{ij} = T_{ij}^0 F_i F_j A_i B_j \tag{2.7}$$

or incorporating growth factors F_i and F_j into new variables a_i and b_j:

$$T_{ij} = T_{ij}^0 a_i b_j \tag{2.8}$$

where, a_i and b_j are balancing factors, and T_{ij} and T_{ij}^0 are the future and observed number of trips between zones *i* and *j*.

In order to compute the equation, one can set all $b_j s$ as 1 and calculate $a_i s$ that satisfy origin constraints. Then using the latest $a_i s$ the column factors $b_j s$ can be calculated to satisfy trip attraction constrains. The process is continued iteratively until the changes are sufficiently small. This method is also called as bi-proportional algorithm, and is a special case of entropy maximization which excludes spatial separation of zones (Ortuzar and Willumsen, 2001).

2.3.3. The Gravity Models

2.3.3.1. Historical Background and Early Forms of Gravity Models

With its well-known theoretical base and various application procedures gravity type of spatial interaction models have been by far the most commonly used aggregate trip distribution models. Simply they assume, analogously to the Newton's Law of Universal Gravitation (1686), that the interaction between any two zones is directly proportional to their magnitudes and inversely proportional to distance between them.

In social sciences, the gravity concept go back to the 19th century to the works of Carey (1858) and Ravenstein (1885;1889). Since first application to identify potential effects and notions of market area for retail trade (*law of Reilly's retail gravitation*, 1929), the gravity models have been extensively employed by geographers, planners and transportation modellers. Considering the trip distribution problem, one of the first study was made by Casey in 1955. He estimated shopping trips in a region and used an early form of gravity model as follows:

$$T_{ij} = \alpha P_i P_j / d_{ij}^2 \tag{2.9}$$

where, T_{ij} is the number of trips between towns *i* and *j*, P_i and P_j are the populations of towns, d_{ij} is the distance between *i* and *j*, and α is a constant.

This basic form has improved over the years and new theoretical insights together with new variables have introduced to the initial gravity model. A general form of classical gravity model can be written as :

$$T_{ij} = \alpha P_i A_j f(C_{ij}) \tag{2.10}$$

where, T_{ij} is the total number of trips between zones *i* and *j*, P_i and A_j are the total number of trips produced in and attracted to zones *i* and *j* , $f(C_{ij})$ is the friction function that exist between zones (generally a decreasing function, power or exponential), and α is an adjustment factor.

Past several decades have brought fundamental contributions to the early gravity models and create a large area of research called "Spatial Interaction Modeling". Today the name spatial interaction models is used interchangeable with the gravity models in any application of flow distributions.

The initial improvements in spatial interaction modelling by Hansen (1959), Huff (1962; 1963) and Lowry (1964), have followed by many others for the subsequent years. Wilson's (1967;1970) 'Family of Spatial Interactions', Alonso's (1973;1978) 'General Theory of Movement' and Fotheringham's (1983) ' Theory of Competing Destinations' were the important contributions in this respect. As well as these improvements, applying spatial interaction models to real cases has been further enhanced especially with the development of new calibration techniques. There is a considerable amount of literature on gravity and spatial interaction models, and it is possible to find excellent reviews in Fotheringham and O'Kelly (1989), Batten and Boyce (1986), Sen and Smith (1995) and Roy (2004).

2.3.2.2. A Family of Spatial Interaction Models

Spatial interaction models drifted further away from its original gravity formulation with the important works of Wilson (1967, 1970) on entropy maximization. The maximum entropy approach created the basis for the development and implementation of numerous operational models including trip distribution analysis. Wilson has distinguished several cases introducing 'A Family of Spatial Interaction Models'. According to his justification, the interactions can be unconstrained, as in early gravity models, production constrained, attraction constrained or doubly constrained.

The main assumption is that any interaction (T_{ij}) between two zones is proportional to total interaction flows (P_i) leaving zone *i*, total interaction flows (A_j) terminating at zone *j*, and some decreasing function of travel cost $f(C_{ij})$ between zone *i* and zone *j* (Wilson, 1974) .

$$T_{ij} \propto P_i \text{ and } T_{ij} \propto A_j \text{ and } T_{ij} \propto f(C_{ij}) \tag{2.11}$$

At this point of view, a constant K can be introduced, which substitutes for the proportionality of interactions.

$$T_{ij} = KP_iA_jf(C_{ij}) \quad (2.12)$$

Then, if either total outflow from i or total inflow to j is known, it can be possible to derive the proportionality constant K as :

$$\sum_j T_{ij} = P_i \quad (2.13)$$

$$K = 1/\sum_i P_i\, f(C_{ij}) \quad (2.14)$$

where P_is are known, and

$$\sum_i T_{ij} = A_j \quad (2.15)$$

$$K = 1/\sum_j A_j\, f(C_{ij}) \quad (2.16)$$

where A_js are known.

Considering these derivations, four cases can be distinguished as:

i) Unconstrained case: Neither the set of row totals nor the set of column totals is known.

ii) Production Constrained case: The set of row totals is known.

iii) Attraction-Constrained case: The set of column totals is known.

iv) Production-Attraction-Constrained (Doubly-Constrained) case: Both sets of interaction totals are known (Wilson, 1974).

2.3.2.3. The Doubly-Constrained Gravity Model

In transportation studies, the number of trips generated and attracted at each zones of origins and destinations is usually known. Corresponding to the case of maximum information, Doubly Constrained Gravity/Spatial Interaction Model (DCGM) has found a wide applicability in trip distribution problems.

The high accuracy of DCGMs' estimations is also shown empirically in Fortheringham and O'kelly (1989). Accordingly, the traditional form of DCGM

introduced with the following expressions, is selected as a benchmark model for the performance measures of the present study. In classical manner, the expression of the doubly-constrained gravity distribution model can be stated as follows:

$$T_{ij} = a_i b_j P_i A_j f(C_{ij}) \tag{2.17}$$

$$a_i = {}^{1}/_{\sum_j b_j A_j f(C_{ij})} \tag{2.18}$$

$$b_j = {}^{1}/_{\sum_i a_i P_i f(C_{ij})} \tag{2.19}$$

where, T_{ij} is the number of trips from zone *i* to zone *j*, P_i is the total number of trips produced in zone *i*, A_j is the total number of trips attracted to zone *j*, $f(C_{ij})$ is the friction factor related to some measure of spatial separation between zone *i* and zone *j*, and finally a_i and b_j are the balancing factors that ensures origin ($\sum_j T_{ij} = P_i$) and destination ($\sum_i T_{ij} = A_j$) constraints are satisfied.

The spatial separation of zones is usually included in the model as a cost of physical distance or measured or assigned travel time. Ones the friction parameter(s) have been calibrated for the base year trip matrix, the future pattern of trips can be simulated easily. The well known friction functions are as follows:

$$f(C_{ij}) = e^{-\beta(C_{ij})} \qquad \text{exponential cost function} \tag{2.20}$$

$$f(C_{ij}) = {C_{ij}}^{-\beta} \qquad \text{power cost function} \tag{2.21}$$

$$f(C_{ij}) = ae^{-\alpha(C_{ij})}C^{-\beta} \qquad \text{combined or gamma cost function} \tag{2.22}$$

The selected calibration techniques and implementation procedure of DCGM are discussed in another chapter of the study. Further reviews on calibration as well as theoretical aspects of spatial interaction and trip distribution modelling can be found in the texts by Fortheringham and O'kelly (1989), Ortuzar and Willumsen (2001), Roy (2004) and Sen and Smith (1995).

2.3.4. The Intervening Opportunities Model

One of the important approach to trip distribution modelling, other than above mentioned gravity type of spatial interaction models, is the 'intervening opportunities model. It was originally developed by Stouffer (1940) and refined by Schneider (1959) in Chicago Area Transportation Study.

According to the intervening opportunities model, trip making is not explicitly related to travel distance but to accessible opportunities. It basically assumes that the trip endings are directly proportional to the number of opportunities at the destination and inversely proportional to the number of intervening opportunities.

Most widely used form of the intervening opportunities model can be written as follows (Vuchic, 2005):

$$T_{ij} = P_i \frac{\left(e^{-LA_j} - e^{-LA_{j+1}}\right)}{\left(1 - e^{-LA}\right)} \tag{2.23}$$

where, T_{ij} is the trip interchanges between zone *i* and zone *j*, L is the probability of travelling to particular destination, V is the total number of opportunities, $V_j \; and \; V_{j+1}$ are the number of opportunities passed up to the zones *j* and *j+1*, and P_i is the total number of trips leaving zone *i*. Although the intervening opportunities model is a kind of spatial interaction model, it differs from the traditional gravity model with its statistical nature and its different measure of attractiveness and impedance terms.

2.4. Advantages and Disadvantages of Traditional Trip Distribution Models

Growth-factor techniques are easy to understand and apply. There is no need for a calibration procedure. However, predicting all interactions with only a single factor and without any transportation system information is not sufficient in many cases. They are especially suitable for short term forecasts in small urban areas where a present trip matrix is available and a friction matrix is not. The base year trip matrix is also supposed to has no sampling error as the growth factor techniques is much more sensitive to actual trip interchanges.

The traditional gravity type of spatial interaction models can be used in many applications with various weights, functional forms, transportation costs and further disaggregation by route choice, trip type, transport mode, and so forth. This flexible structure with well-established calibration and validation procedures constitutes the main advantage of gravity models. Additionally, its underlying theory is simple and policy-responsive. Finally, calibration procedures of gravity models are well-known and available with many computer packages.

However, the gravity models have also some disadvantages. First, the gravity models are based on existing travel behaviour pattern and transportation system characteristics. The friction factor and other socio-economic factors or parameters are very unlikely to remain stable. Second, gravity models use only physical separation as a friction factor and this leads to the exclusion of other behavioural factors and opportunities. Third, they assume all the information is related with base year trip matrix and the trip end constraints. This would be also problematic when any sampling error is included with the base trip matrix and the outputs of the trip generation phase are not sufficiently accurate. Finally, gravity models do not use any explicit variable of individuals or households behaviour which is valid with all aggregate models.

Both the gravity models and the intervening opportunities model have been used in many urban transportation studies over the years. One of the main difference of these models is that the gravity models are deterministic and intervening opportunities are probabilistic. As Taafe et al. (1996) states that the results of the empirical use of both models has shown that they are equally effective in describing and predicting trip distributions.

However, although it begins from different principles and have some useful insights, the intervening opportunities model is not often used in practice. According to the Ortuzar and Willumsen (2001) the reasons of this situation would be: i) its theoretical basis is less well known and possibly more difficult by practitioners; ii) it does not include any practically measured trip cost attribute; iii) the lack of suitable software; iv) and finally its theoretical and practical advantages over the gravity models are not overwhelming. As Wilson (1974) states that the intervening opportunities model can be seen as a specialized gravity model while many of the features which are applied to gravity model could be applied to the intervening opportunities model.

CHAPTER 3

MODELLING TRIP DISTRIBUTION WITH SOFT COMPUTING TECHNIQUES: A REVIEW

3.1. Soft Computing Applications in Traffic and Transport Systems

Over the past decades, a variety of techniques of computational intelligence and artificial life have influenced regional science research to understand more fully the natural complexity of many spatial and regional systems. Achieving and evaluating huge amount of digital spatial data with the help of GIS and fast soft computing techniques have led spatial analysts to recall the traditional explanatory spatial models eliminating the past drawbacks (Roy and Thill, 2004).

The transportation systems are naturally complex systems involving a very large number of components and different parties, each having different and often conflicting objectives. In order to cope with its complexity, there has been increasing interest among both transportation researchers and practitioners in exploring the feasibility of applying Artificial Intelligence (AI) based techniques to real transportation problems (Sadek, 2007) .

According to Sadek (2007) AI refers to methods that mimic biologically intelligent behaviour in order to solve problems that is difficult to solve by classical mathematics. At present time, AI methods can be divided into two broad categories: i) symbolic AI, which focuses on the development of knowledge-based systems; and ii) computational intelligence, which includes such methods as artificial neural networks, fuzzy systems, and evolutionary computing as shown in Figure 3.1. Bezdek (1994; cited in Konar, 2005, p.5) describes a computationally intelligent system as follows.

> A system is computationally intelligent when it: deals with only numerical (low data), has pattern recognition components, does not use knowledge in the A I sense, and additionally when it (begins to) exhibit i) computational adaptivity, ii) computational fault tolerance, iii) speed approaching human-like turnaround and iv) error rates that approximate human performance.

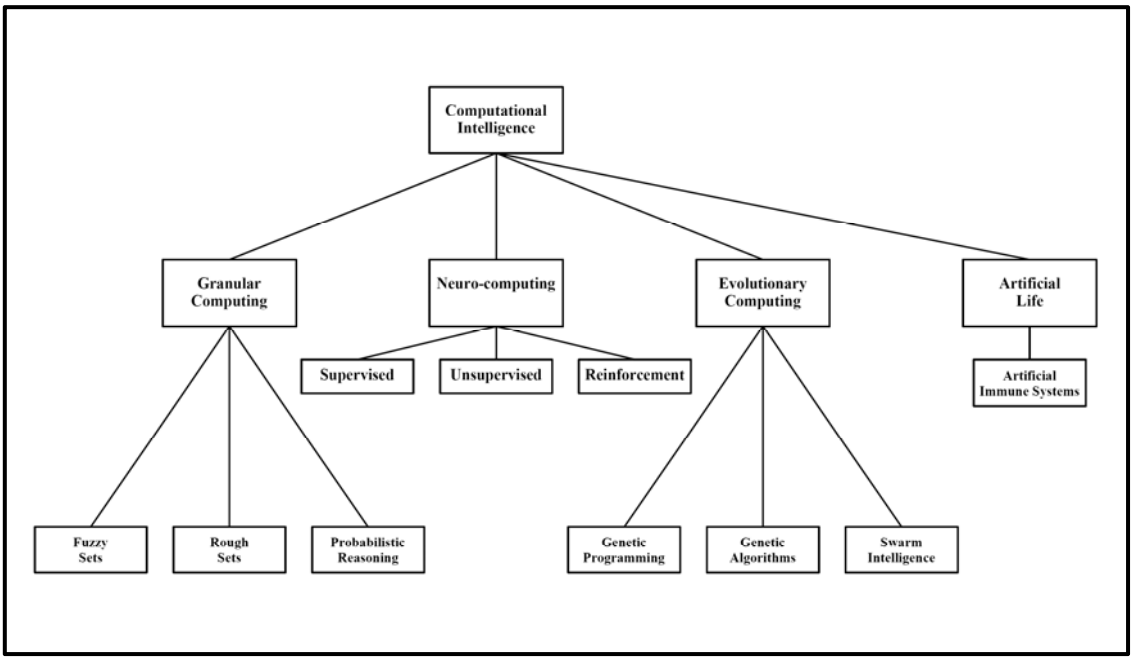

Figure 3.1. Computational Intelligence Based Techniques: A Family Tree
(Source: Konar, 2005, p.vi)

Especially with the 1990's, many researchers started to study the synergistic behaviour of computational intelligence based techniques. Zadeh called this synergism as *'Soft Computing'* and described their synergistic rather than competitive behaviour as in following two quotations:

> Soft computing is not a homogenous body of concepts and techniques. Rather it is a collection of methodologies, which in one form or another reflect the guiding principle of soft computing: exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost. Viewed in a slightly different perspective, soft computing is a consortium of methodologies which, either singly or in combination, serve to provide effective tools for the development of intelligent systems (Pedrycz, 1996;cited in Konar, 2005,p. 8)

> ...a recent trend to view fuzzy logic (FL), neurocomputing (NC), genetic computing (GC) and probabilistic computing (PC) as an association of computing methodologies falling under the rubric of so-called soft computing. The essence of soft computing is that its constituent methodologies are for the most part complementary and synergistic rather than competitive. A concomitant of the concept of soft computing is that in many situations it is advantageous to employ FL, NC, GC, and PC in combination rather than isolation (Pedryc and Gomide, 1998;cited in Konar, 2005, p.8).

In parallel with the other engineering sciences, Neural Networks (NNs), Fuzzy Logic (FL) and Genetic Algorithms (GAs) have been the most featured soft computing techniques used in transportation research. A number of studies have shown applicability of those techniques to transportation problems such as in traffic control and design, demand analysis and logistics. Here an overview of these studies is given

and fundamentals of these widely used soft computing techniques is introduced in the subsequent sections of the study.

A recent study (Avineri, 2005) gives a brief overview of the use of soft computing methodologies for modelling and analyzing traffic and transport systems. According to this study, during the last three decades approximately one thousand researches have established in the field of traffic and transport systems dealing with soft computing techniques. The field of traffic control and management with a rate of 37% has become the most studied field of transportation researches. The issues included in transport planning and management, and transport policy and administration studies have followed the traffic control and management studies with a total rate of 29%.

Table 3.1 shows the detailed results of Avineri's (2005) literature review according to the issues in traffic and transportation systems. Additionally, Figure 3.2 shows the citation results according to the yearly development of soft computing techniques.

According to the Avineri (2005), travel demand modelling with many special issues other than four step modelling is included in the section of modelling of travel choice behaviour with 83 (8%) citation. First applications of soft computing techniques to travel demand modelling are based on fuzzy rules and classical tools of fuzzy control. Teodorovic and Kikuchi (1990), Lotan and Koutsopoulos (1993) and Henn (2000) has studied route-choice problem with fuzzy set theory. Then several researchers compare the Neural Networks in different travel demand modeling stages, such as Black (1995), Ji-Rong (2000), Mozolin et al., (2000), and Kim (2001etc. Genetic algorithms were recently used in route-choice modelling by Nakayama (2000), and in trip distribution by Kalic and Teodorovic (2003).

The reader is referred to works by Teodorovic (1994; 1999), Teodorovic and Vukadinovic (1999), Dougherty (1995) and TRB (2007) for an in-depth coverage of the traffic and transportation applications of NNs, FL and GAs.

Table 3.1. Classification for Soft Computing Research in Traffic and Transport Systems (Source: Avineri, 2005, p.18)

Research Topics	Number of Citations
Traffic Control and Management	375 (37%)
Transport Planning and Management, Transport Administration, Transport Policy Modeling of Travel Choice Behaviour Transport Projects Selection Other Issues of Transport Planning	83 (8%) 17 (2%) 196 (20%)
Logistics	40 (4%)
Design and Construction of Transport Facilities Including Geometric Design, Pavement Management, Construction, Materials Properties	112 (11%)
Other Applications of Traffic and Transport Systems, and Review Papers Including Planning and Operating Public Transport, Operating and Management of Parking Facilities, Maintenance of Traffic and Transport Systems, Airline Network Applications, Airport Planning and Others	181 (18%)
Total	**1004**

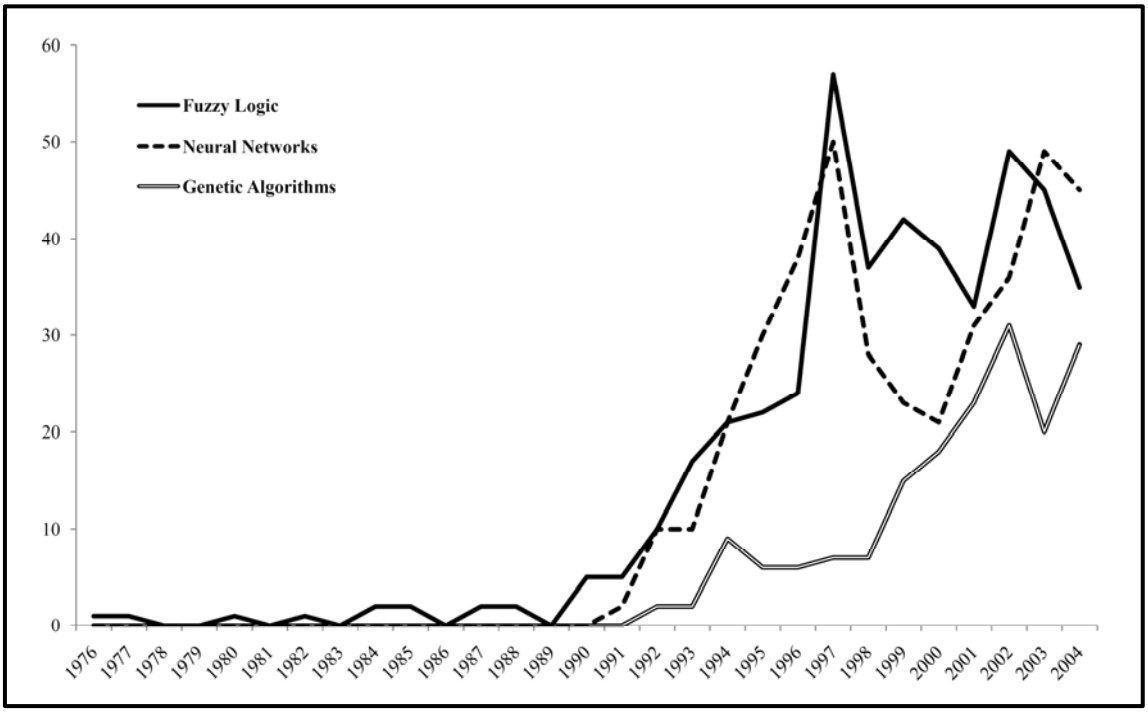

Figure 3.2. Number of Citations with NNs, FL and GAs by Years (Source: Avineri, 2005, p.20)

3.2. Modelling Trip Distribution with Neural Networks

Artificial Neural Networks, commonly referred to as Neural Networks (NNs), are computational models of the brain. Similar to the structure of brain, artificial neurons which are interconnected by edges constitutes a layered network, the network receives input, performs some internal process such as activations of the neurons, and produces output (Munakata, 2008). Multilayer NNs are at the same time universal approximators (Hornik et al., 1989; Hornik, 1991), by adjusting the connection weights of the neurons, NNs can be "trained" to approximate any nonlinear function to a required degree of accuracy (Sadek, 2007).

Especially in the last two decades, NNs have been applied in many scientific disciplines as well as transport geography and modelling. The application potentials of NN models across the traditional models have been shown in many studies. As the NNs have a highly flexible and equation free structure, a number of function approximation, system identification and control, nonlinear modelling, pattern recognition and optimization problems have solved or resolved with NNs.

The reader is referred to some previous publications by Doughtery (1995), Mussone (1999), Avineri (2005) and Ishak and Franco (2007) for a review on some of the NN applications in transport geography and traffic engineering, and to valuable works by Munakata (2008) and Haykin (1999) for a detailed theoretical framework of NNs.

Several recent studies have also proposed the use of NNs to model spatial interactions and trip distribution. Openshaw (1993) presented the potential use of NNs in spatial interaction modelling and Fischer and Gopal (1994) showed the applicability and predictive accuracy of NNs in modelling distribution of interregional telecommunication flows. Many others have followed these pioneering works in trip distribution modelling: Black (1995) and Celik (2004) modelled commodity flows and Mozolin et al. (2000), Tillema et al. (2006) and Tapkin (2009) modeled intercity passenger flows with NNs.

Nearly all the scholars compared the NNs predictive performance with some of conventional gravity type of spatial interaction models. In many cases, NNs outperformed the conventional models and it is concluded that NNs may perform well enough to estimate spatial interaction flows in general. The only differentiating

conclusion was presented by Mozolin et al. (2000) and Celik (2004). They trained the networks for a base-year interaction matrix and tried to forecast a known matrix at projection year. They concluded that the NNs may perform better than conventional models for the base year matrix, but they fail to outperform conventional models for forecasting purposes.

Although the debates are continuing, the NN based spatial interaction models have been in use for more than fifteen years. So, it would be appropriate for us to establish a NNs based trip distribution model and compare the overall performance measures with the proposed GFRBS design. The selected network topology and its training and implementation issues are discussed in subsequent sections of the present study.

3.3. Modelling Trip Distribution with Fuzzy Logic and Genetic Algorithms

The Fuzzy Set Theory (1965) and the Fuzzy Logic (1973) were first introduced by Lotfi Zadeh as a mathematical tool for dealing with uncertainty, imprecision, subjectivity and linguistic terms. Since then, a number of researches as well as a number practical engineering applications have been established using the concept of fuzziness. The fuzzy logic or FRBSs has proved to be a good tool for a wide range of application areas such as in system/process control, pattern recognition, classification, non-linear input-output mapping, approximate reasoning, machine learning and decision making.

As Kalic and Teodorovic states (2003, p.214), fuzzy logic applications in solving different kinds of problem can be divided into two categories.

- Firstly, use of fuzzy logic is a suitable for treating subjectivity, ambiguity, uncertainty and imprecision, when we do not have sufficiently precise input data, or the data including subjective feeling of the expert and are most often described in linguistic terms. Over the last three decades a number of models which treat subjectivity, uncertainty and imprecision have been developed using fuzzy logic.

- Secondly, fuzzy logic can be used with the problems in which uncertainty, subjectivity or imprecision are not present. Recently, significant theoretical results have been achieved in the field of fuzzy systems. Fuzzy logic has proved to be a good tool for tackling problems involving a mapping of inputs into outputs. Wang (1992), Wang and Mendel (1992) and Kosko (1994) showed in their works that the fuzzy logic systems can be treated as universal approximators.

In traffic and transportation studies, FRBSs have been applied in each of these categories especially for selecting transportation investment projects, and modelling trip generation, trip distribution, modal split, and route choice. As being universal approximators, they have also been used in traffic controls and related studies including aircraft control, ship loading/unloading control, intersection signal control, accident analysis/prevention, and level of service evaluation (see Teodorovic,1994; 1999).

Apart from these applications, learning fuzzy rules and tuning fuzzy membership functions are the two key components for an FRBS. Genetic Algorithms (GAs) have proven suitable for solving both combinatory optimisation and parameter optimization problems. Employing GAs to construct a fuzzy system with learning process from examples can greatly enhance the control performance of a fuzzy system (Chiou and Lan, 2005). This line of research has spurred broad use of fuzzy systems improved by a GA learning process: Genetic Fuzzy Systems (GFS) and in particular GFRBSs (see Cordon et al., 2004; Ishibuchi, 2007; Herrera, 2008).

Considering the spatial interaction and trip distribution modelling, it is possible to mention about some important studies that uses fuzzy inference systems and/or genetic algorithms. According to these studies listed below, FRBSs can be used to solve trip distribution problem efficiently and, together with the use of GAs, it is possible to achieve better model performances. However, the performance of FRBSs against a doubly-constrained gravity model and a NNs based trip distribution model has still not been investigated. Moreover, the GFRBS has still not been adapted for the prediction of intra-city passenger flows, which adds computational burden and complexity with an additional friction variable and additional fuzzy rules. The present study tries to make up these shortages with an empirical analysis.

- Kalic and Teodorovic (1996) estimated air passenger flows between some major industrial cities and tourist resorts using known productions and attractions as inputs. In comparison to non-fuzzy methods, the FRBS that they used gave better results.

- Kalic and Teodorovic (2003) carried out their studies with the use of GAs to optimize initial fuzzy rule-based system. They have achieved better results with a GFRBS design.

- Openshaw (1997;1998) proposed the use of fuzzy set systems in modelling spatial interactions and showed some empirical results for a origin-constrained gravity model using Sugeno-type of fuzzy inference.

- Diplock and Openshaw (1996) investigated the use of genetic algorithms in an attempt to obtain globally optimal parameter estimates for a mix of simple and complex spatial interaction models.

- Shafahi et al. (2008) proposed a simple FRBS to predict the discretionary trips in Tehran showing its capability in intra-city passenger flows. They used travel time as third input and gained better results against the unconstrained type of gravity model.

CHAPTER 4

FUNDEMENTALS OF FUZZY AND GENETIC FUZZY SYSTEMS

4.1. Fuzzy Logic and Fuzzy Rule-Based Systems

The concept of *fuzziness* (1965) and *fuzzy logic* (1973) was first introduced by Zadeh to deal with uncertainty, imprecision and partial truth. Simply, fuzzy logic provides a way to draw definite conclusions from vague information and it enables using linguistic terms and human like reasoning in modelling complex real-life systems.

The term fuzzy logic is generally used in two sense: i) in a narrow sense, it can be seen as a branch of fuzzy set theory dealing with logical systems where classical logic suffers, ii) in a broad sense, it can be specified as synonymously with fuzzy set theory, fuzzy control systems and fuzzy modelling. This study uses the term fuzzy logic as its broad and comprehensive meaning.

As mentioned earlier, fuzzy logic is useful in two general context: i) in situations involving uncertainty, imprecision and partial truth, and ii) in situations where mapping any inputs into desired outputs even if there is no uncertainty and imprecision exist. Present study introduces a fuzzy rule-based system designed in the latter context. The following two sections give a brief overview of fuzzy sets and introduces a general Fuzzy Rule-Based System (FRBS) also used in this study.

4.1.1. Fuzzy Sets and Membership Functions

Fuzz logic and fuzzy systems are mainly based on fuzzy set theory. Its mathematical foundations can be seen as a generalization of classical set theory. In classical set theory, boundaries of sets are rigid and elements are either members or not members of predefined crisp sets. If an object belongs to a set, its membership function value is 1, otherwise it is 0.

Classical set theory is not sufficient to describe vague concepts especially in real life cases. Seeing inadequacies of this binary structure, Zadeh (1965) introduced a gradual membership concept to ordinary sets with overlapping boundaries. In fuzzy sets, boundaries are not price and can overlap, additionally many degrees of membership are possible between a closed interval [0,1].

All information contained in a fuzzy set is described by its membership function, most fundamental parts of fuzzy sets. Membership function of a set $\mu_A(x)$, maps each element to its degree between 0 and 1. They can be formed in any discrete or continuous functions. Some of the most widely used continuous membership functions are triangular, trapezoidal, s-shaped, sigmoidal and gaussian functions.

A membership function is mainly constituted from three parts: the *core*, crisp subset of the universe that represents complete membership; the *support*, non-zero membership range of the membership function; and the *boundaries*, region of the universe that includes greater than zero, but smaller than complete membership degree (Figure 4.1).

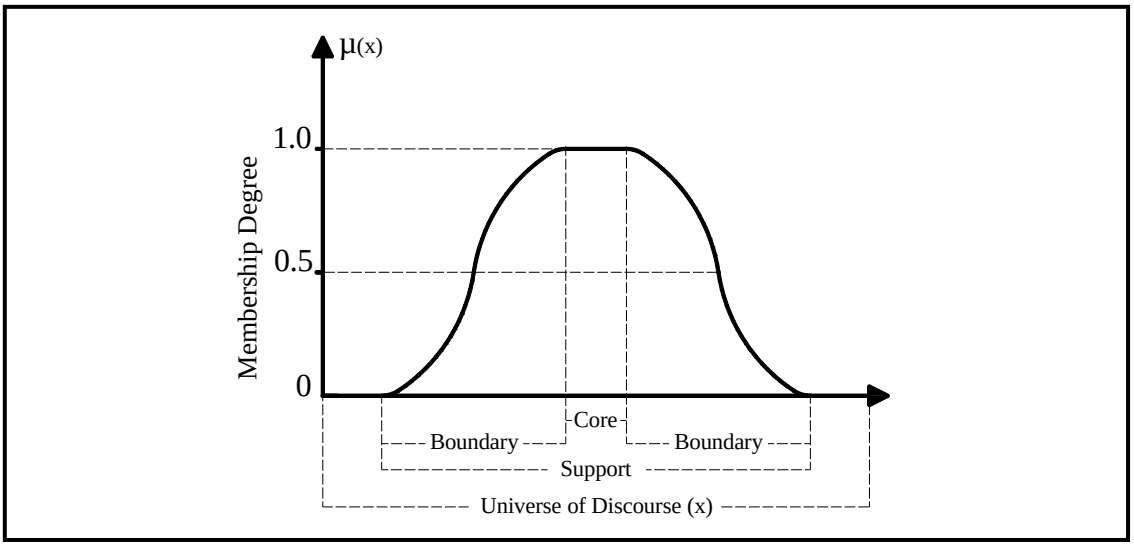

Figure 4.1. Fuzzy Membership Function Components

Let us consider a universe of discourse , say 'distance to city centre', and define three subsets: low, medium and high distances. If some appropriate intervals are given to predefined membership functions, the crisp and fuzzy sets can be shown graphically as in Figure 4.2.

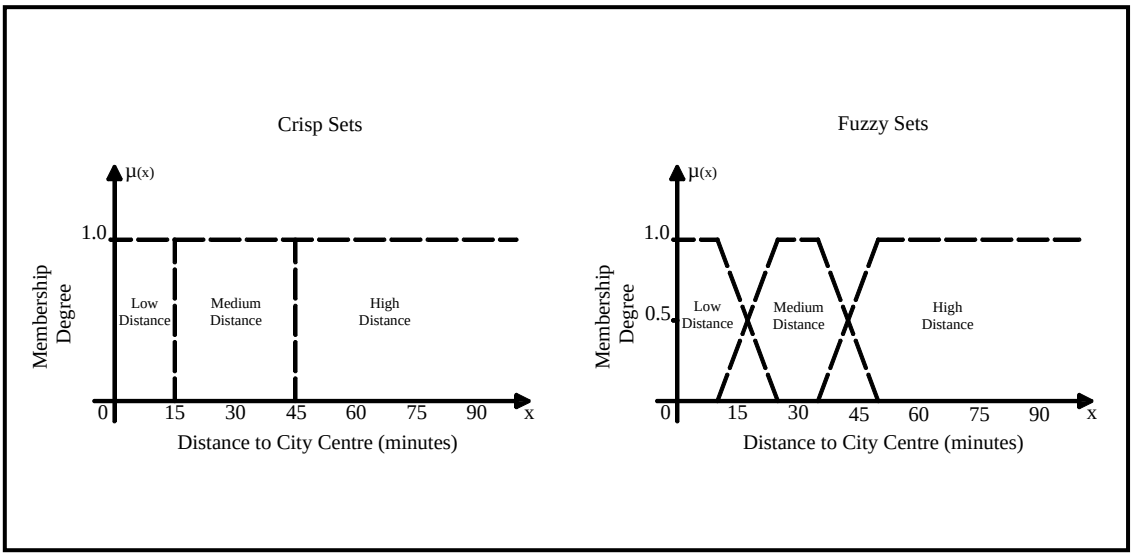

Figure 4.2. Crisp and Fuzzy Sets: An Example

Additionally, for an element x that belongs to subset A , say 'medium distance to city centre', the membership degree, $\mu_A(x)$, of crisp and fuzzy subsets can be calculated as in equations 4.1 and 4.2 respectively. As seen in the figure and the mathematical form of fuzzy membership function, any value of x can get different membership degrees other than exact 0 and 1. Moreover, the element x can also be a member of a nearby subset if it is in a *fuzzy* or *gray* area.

$$\mu_A(x) = \begin{cases} 1 & \text{if} \quad 15 \leq x \leq 45 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$\mu_A(x) = \begin{cases} \frac{x-10}{25-10} & \text{if} \quad 10 \leq x \leq 25 \\ 1 & \text{if} \quad 25 \leq x \leq 35 \\ \frac{x-50}{35-50} & \text{if} \quad 35 \leq x \leq 50 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Classical sets can be considered as a special case of fuzzy sets that is restricted to certain values. So, nearly all classical set operations and properties are supported by fuzzy sets. Three of the operators have remarkable importance, union, intersection and complement, which are illustrated in Figure 4.3 and summarized in Table 4.1.

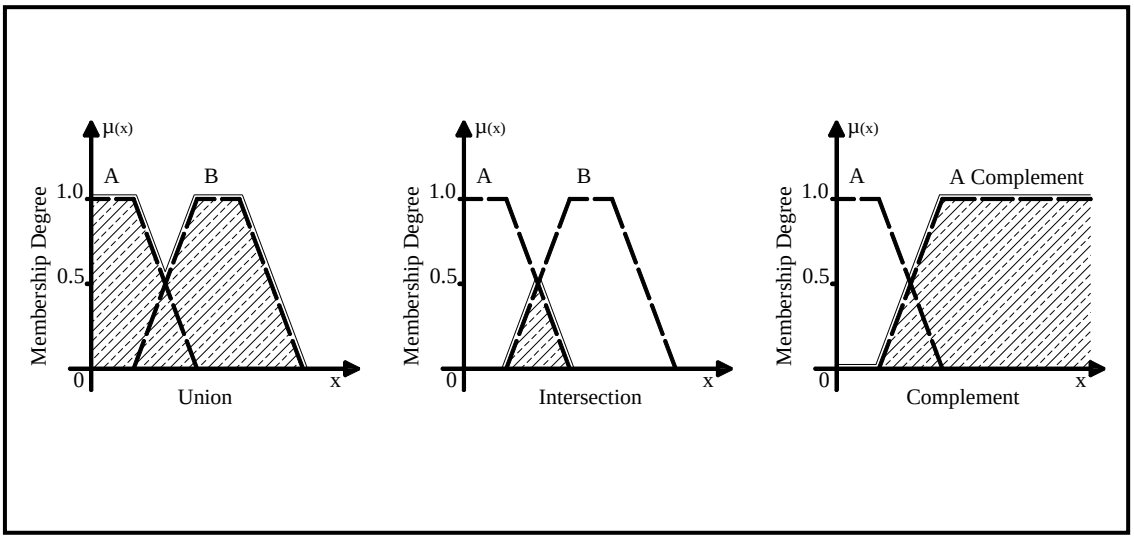

Figure 4.3. Fuzzy Set Operations: Union, Intersection and Complement

Suppose that *A* and *B* are two fuzzy sets on the universe *X*, then the *union* of sets *A* and *B* is denoted by $\mu_{A \cup B}(x)$. The fuzzy union operator is also called as logical *'or'* , representing all the elements that belong to either set A or set B.

Similarly, the *intersection* of fuzzy sets *A* and *B* is denoted by $\mu_{A \cap B}(x)$, and represents the region where all the elements are member of both of set A and set B. The fuzzy intersection operator is also called as logical *'and'* operator.

The *complement* of fuzzy set *A* is a new fuzzy set denoted by $\bar{A}$, which represents elements that are not belong to set A. Complementation operator fuzzy sets is also known as logical *'not'* operator.

Other properties of classical sets can also be applied to fuzzy sets. Classical set properties such as, commutativity, associativity, distributivity and identity are familiar with fuzzy set properties.

Table 4.1. Fuzzy Set Operations: Union, Intersection and Complement

Fuzzy Set Operator	Logical Operator	Mathematical Expression
Union	OR	$\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$
Intersection	AND	$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$
Complement	NOT	$\mu_{\bar{A}}(x) = max\{1 - \mu_A(x)\}$

4.1.2. Fuzzy Rule-Based Systems (FRBSs)

Fuzzy Rule-Based Systems (FRBS) are one of the most important application areas of fuzzy set theory. As an extension of classical rule-based systems, a FRBS uses fuzzy sets and fuzzy logic to represent and connect knowledge which is usually linguistic. Because of its multi-disciplinary nature, FRBS are also known as fuzzy inference systems, fuzzy control systems, fuzzy expert systems, fuzzy reasoning, fuzzy modelling and finally fuzzy systems which is more broad.

FRBS are most useful in modelling complex systems that can be observed by humans. The most common way to represent human knowledge is to form it into natural language expressions: *IF premise, THEN conclusion*. This expression is commonly referred as an IF-THEN rule-based system. They enable use of linguistic variables as antecedents and consequents, and logical connectives as well. Using basic properties and operations of fuzzy sets, any compound rule structure can be decomposed and reduced to a number of simple canonical rules as shown in Table 4.2 Canonical rules may include either assignment statements, conditional statements or unconditional statements. Conditional and unconditional statements place restrictions on the consequent, and linguistic connections such as "and", "or", "not" connect them each other (Ross, 2004).

Table 4.2. The Canonical Form Of A Fuzzy Rule-Based System

Rule 1:	IF Condition C^1, THEN restriction R^1
Rule 2:	IF Condition C^2, THEN restriction R^2
.....	.....
Rule r:	IF Condition C^r, THEN restriction R^r

The most commonly used FRBS can be distinguished into two main configurations: i) *Mamdani-type FRBS*, proposed by Mamdani (1974) and Mamdani and Assilian (1975), and ii) *Sugeno-type FRBS*, introduced by Takagi and Sugeno (1985) and Sugeno and Kang (1988).

They are all similar in their antecedents and rule base structure and their ability to process human like reasoning with linguistic variables. However, they become different in producing system outputs. The output of a Mamdani-type FRBS is a fuzzy

set as in its inputs, whereas the output of a Sugeno-type FRBS is generally either a linear function of its inputs or some constants. An approximate fuzzy rule can be formed as following expressions in each case:

Mamdani-type FRBS: ***IF*** X_1 *is* A_1 *and* X_2 *is* A_2 *and* ..... X_i *is* A_i, ***THEN*** Y *is* B

Sugeno-type FRBS: ***IF*** X_1 *is* A_1 *and* X_2 *is* A_2 *and* ..... X_i *is* A_i, ***THEN*** $Y = a_0 + a_1X_1 + a_2X_2 + ...+a_iX_i$

Mamdani-type FRBS also used in this study, is the most common FRBS in practice and in the literature. It generally deals with mapping crisp inputs into crisp outputs and enables the use of linguistic variables and expert knowledge. This knowledge can be easily combined with automatically generated rules from data sets that describe the relation between system input and output as in our case.

Either Mamdani-type or Sugeno-type, a FRBS generally consist of four main components: *fuzzification*, *knowledge base*, *inference* and *defuzzification*. A general form of a FRBS and its components is shown in Figure 4.4 and is described as following.

Fuzzification, is the process of mapping inputs into linguistic fuzzy sets and computing their membership degrees for rule antecedents. In most cases, inputs are crisp numerical values which are then transformed into fuzzy values.

Knowledge base, is the most fundamental part of a FRBS and constituted from a data base and a rule base. The data base keeps linguistic variables or in other words fuzzy membership functions for both of the inputs and outputs. The rule base, on the other side, includes a collection of fuzzy If-Then rules. All other components of a FRBS uses the information preserved in knowledge base.

Inference system, is responsible for combining fuzzy sets with corresponding logical operators, then giving an fuzzy output. The type of combining is changeable depending on selected implication method such as *max-min* or *max-product implication*. Additionally, the type of outputs may differ according to the selected inference system: A *Mamdani-Type inference system* produces fuzzy outputs to defuzzify, on the other hand, a *Sugeno-Type inference system* produces a vector of real values.

Defuzzification, is the process of reduction which maps aggregated fuzzy results into a crisp output. There are also a number of defuzzification methods. The most widely used of them are the *Centre of gravity* and *weighted average methods*.

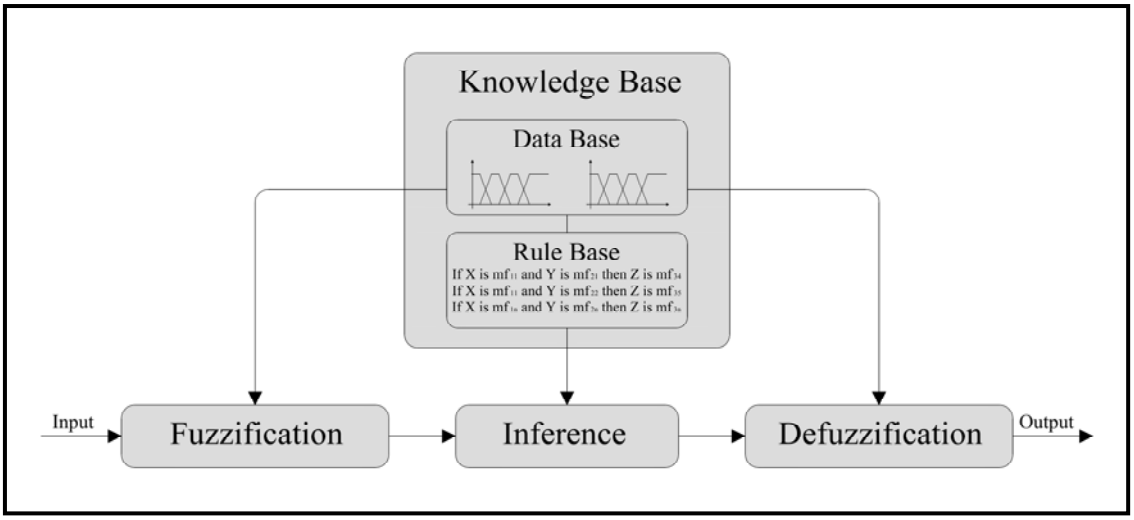

Figure 4.4. General Form of A Fuzzy Rule-Based System

Graphical interpretation of a FRBS can considerably help us to understand nature of fuzzy modelling. Implication of a Mamdani-type Fuzzy Rule-Based System can be distinguished into four steps as described below and illustrated in Figure 4.5 :

Step 1 : **Initialization:** Define input and output variables. Normalize data into some appropriate range if it is required. Constitute the knowledge base: set fuzzy membership functions on variables and determine fuzzy rules.

Step 2 : **Fuzzification:** Compute fuzzy membership values of actual inputs using corresponding membership functions, usually as crisp into fuzzy values.

Step 3 : **Inference:** Combine fuzzy sets with logical operators using appropriate implication algorithm. Aggregate all outcome to generate fuzzy output.

Step 4 : **Defuzzification:** Convert aggregated fuzzy output into crisp output using a defuzzification method.

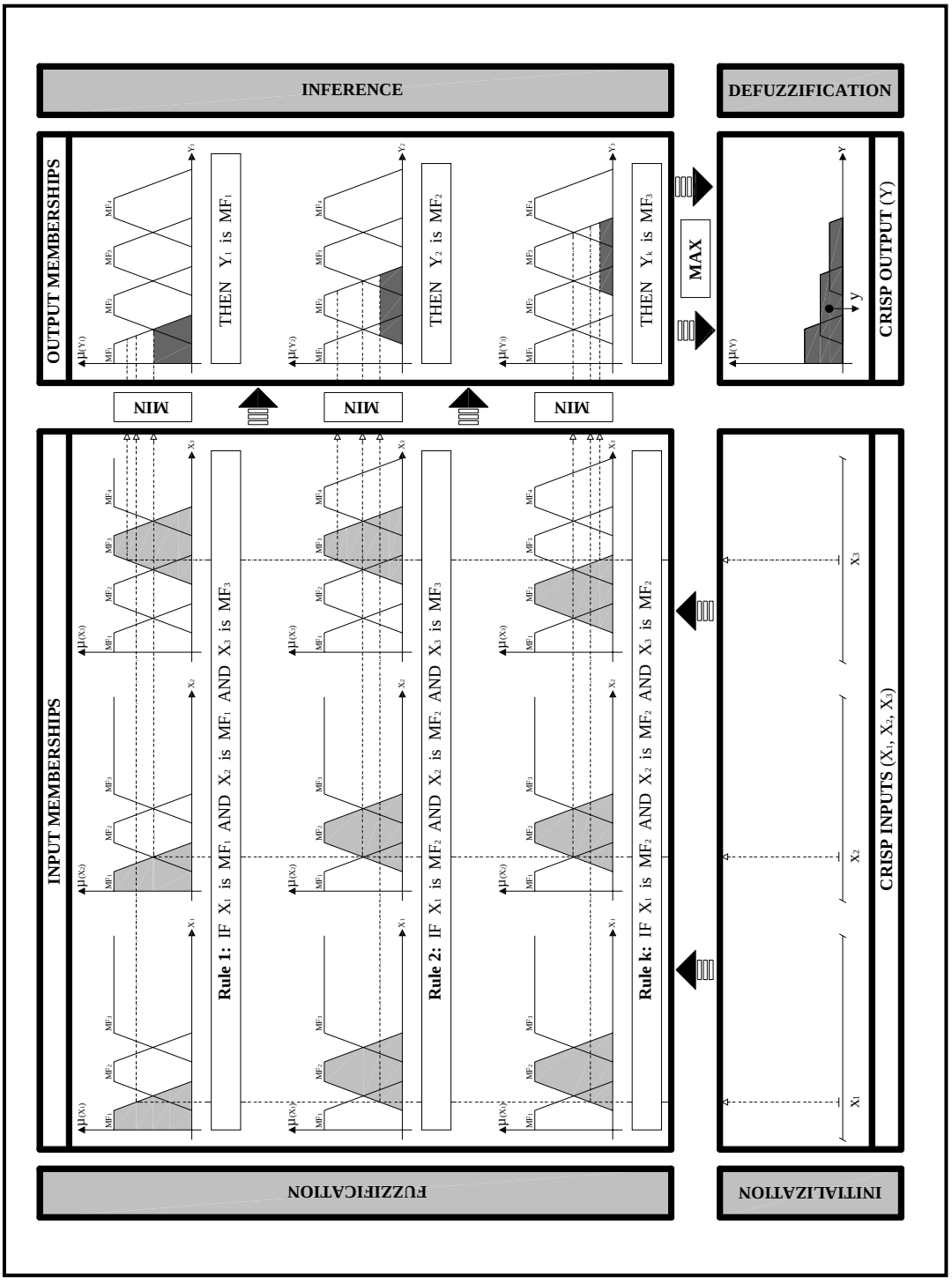

Figure 4.5. Graphical Interpretation of A General Mamdani-Type FRBS

Development of fuzzy membership functions and generating a fuzzy rule base (rule induction) are two key components of a FRBS design. There is a number of way to assign membership values or functions to fuzzy variables. The most widely used techniques are based on some numeric or logical operations or they are intuitive. These techniques can be distinguished into six: *i) intuition, ii) inference, iii) rank ordering, iv) neural networks, v) genetic algorithms, and vi) inductive reasoning*.

Apart from these, there are several automated methods for fuzzy systems which provide additional procedures to develop membership functions as well as rule base learning procedures. *Batch least squares, recursive least squares, gradient method, learning from example, modified learning from example and clustering method* are some of the important algorithms for fuzzy system development (Ross, 2004).

Among the automated fuzzy models, the *learning from example* is one of the most widely used technique for fuzzy rule induction. This technique is originally developed by Wang and Mendel (1992) as a supervised data mining technique to generate fuzzy rules from numerical data.

It is also known as *Wang-Mendel Method* or *One-Pass Method* which is a simple FRBS design method that generates a set of IF–THEN rules by performing a one-pass operation on the given input–output data, and then combines the rules in a common rule base, to construct a final FRBS. It can be described with the following five steps as indicated in Mendel and Mouzouris (1997, p. 888-889):

The One-Pass Method:

Given a set of input–output pairs,

$$\left(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}; y^1\right), \left(x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}; y^2\right) \dots \quad (4.3)$$

where $x_1, x_2, \dots, x_n$ are inputs and is the output, we proceed as follows to construct a FLS:

1) Let $[x_1^-, x_1^+], [x_2^-, x_2^+], \dots, [x_n^-, x_n^+]; [y^-, y^+]$ be the domain intervals of the input and output variables, respectively, where domain interval implies the interval a variable is most likely to lie in. We divide each domain interval into 2N+1 regions, where N can be different for each variable. Then, we assign membership functions to the regions, labelled as SN (Small N), ..., S1(Small 1), CE (Center), B1 (Big 1), ..., BN (Big N).

2) We evaluate the membership of each input–output point in regions where it may occur, and assign the given $x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}$, or $y^{(i)}$ to the region with maximum membership.

3) In order to resolve conflicting rules, i.e., rules with the same antecedent membership functions and different consequent membership functions, we assign a degree to each rule as follows: $\mu_{x_k}(x_k)$ let denote the kth membership of the input variable in the region X_k with maximum membership, and $\mu_Y(y)$ the membership of the output variable in the region Y with

maximum membership, where X_k and Y are labels from their corresponding sets SN, ..., S1,CE,B1,...BN. Then, the degree for the lth rule, R^l , is defined as

$$D(R^l) = \prod_{k=1}^{n} \mu_{x_k}(x_k)\mu_Y(y) \quad (4.4)$$

In the event of conflicting rules, the rule with the highest degree (eq.4.4) is kept in the rule base, and all other conflicting rules are discarded.

4) We generate a combined rule base comprised both of numerically generated fuzzy rules (as described above) and linguistic information provided by experts.

5) After the combined rule base is generated, we employ a defuzzification method (such as Center-of-Area, Center-of-Sums, Height defuzzifier), to obtain the crisp output of the FLS.

4.2. Genetic Fuzzy Systems

Fuzzy rule-based systems have an advantage of storing knowledge that is learned from the data itself or set up by an expert. However, they lack a self-learning feature. If knowledge of the system is fixed and well-defined it is easy to design an effective FRBS. On the other hand, an increase in the size and complexity of the knowledge base complicates the process of designing an optimum FRBS. One of the recent important approaches to removing this learning deficiency of fuzzy systems is to enhance them using GAs.

The use of GAs in enhancing or optimizing fuzzy systems has started a new field of research that is called Genetic Fuzzy Systems (GFSs). The pioneering works of GFSs can be dated back to the early 1990s (Karr,1991; Thrift, 1991; Pham and Karaboga, 1991; Valenzuela-Rendon,1991). Since then, evolutionary learning of fuzzy systems has been extended and several types of GFS have been developed. Genetic Fuzzy Rule-Based Systems (GFRBS), genetic fuzzy clustering systems, genetic fuzzy neural systems and genetic fuzzy decision trees has constituted the main types of GFS.

The most widely used types of GFSs are GFRBSs, which incorporate evolutionary techniques to achieve automatic generation or modification of each component of the FRBS knowledge base. The present study proposes a GFRBS to model intra-city passenger flows. A brief review of GAs and main components of GFRBS are introduced in the following two sections of the study. Further reviews and various applications of GFSs can be found in the texts by Cordon et al. (2001), Bodenhower and Herrera (1997) and Herrera and Verdegay (1996)

4.2.1. A Brief Description of Genetic Algorithms

Genetic Algorithms (GAs) are effective tools with acceptable solutions when exploring large search spaces in a reasonable time. First initiated by Holland (1975) and his colleagues, they are guided random search techniques which are primarily based on Darwin's principals of natural selection and the genetics branch of Biology. In a GA process, genetic codes of individuals within a population evolve into a solution with the overriding principal of survival of the fittest. New generations produce individuals having improved genetic codes with reproduction, crossover, and mutation operators.

Typically a GAs procedure can be identified with some main steps as: the creation of the initial population and the evaluation function; the determination of chromosome representation (generally binary strings) and selection function; and finally the set of parameters for genetic operators, reproduction, and termination criteria (Figure 4.6). Genetic algorithms work well in a wide variety of engineering problems. They have attracted considerable attention in a great number of disciplines as a methodology of search, optimisation and learning, especially after the work of Goldberg (1989). A brief explanation of GAs is introduced below, further reviews and applications can be found in the text by Haupt and Haupt (2004), Sivanandam and Deepa (2008), Gen and Cheng (2000) and Bodenhofer (1999).

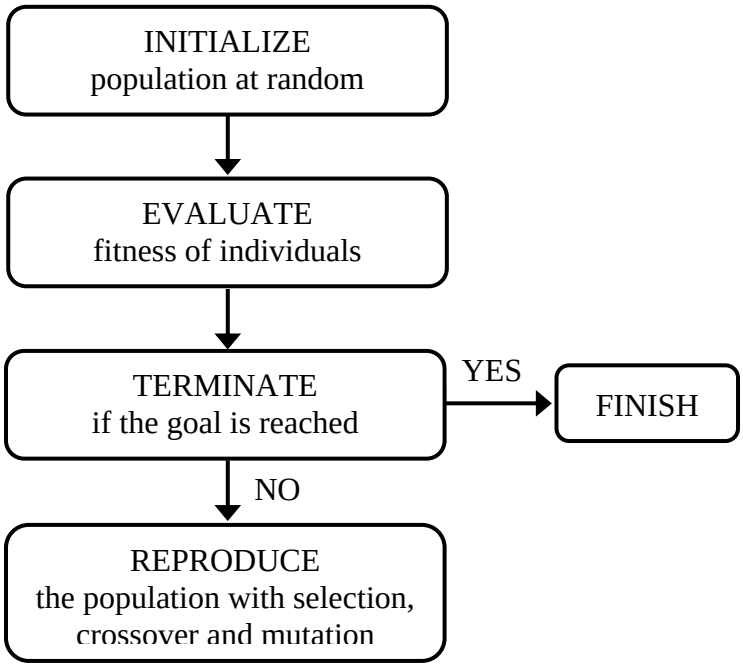

Figure 4.6. A Typical GAs Procedure

The main steps of simple GAs can be further described with the following operations (Sivanandam and Deepa, 2008, p. 30-31):

- Start: Genetic random population of n chromosomes (suitable solutions for the problem)
- Fitness: Evaluate the fitness f(x) of each chromosome x in the population
- New population: Create a new population by repeating following steps until the new population is complete
    - Selection: select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
    - Crossover: With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
    - Mutation: With a mutation probability, mutate new offspring at each locus (position in chromosome)
    - Accepting: Place new offspring in the new population.
- Replace: Use new generated population for a further sum of the algorithm.
- Test: If the end condition is satisfied, stop, and return the best solution in current population.
- Loop: Go to step 2 for fitness evaluation.

Additionally, GAs can be realized with the following procedural code where *P(t)* denotes a population of chromosomes at time *t* (Konar, 2005, p.324):

```
BEGIN
    t=0;
    initialize P(t);
    evaluate P(t);
    while (termination condition not satisfied) do
        begin
        t=t+1;
        select P(t) from P(t-1);
        alter P(t);
        evaluate P(t);
    end;
END
```

4.2.2. Genetic Fuzzy Rule-Based Systems (GFRBSs)

The subject of a GFRBS is to learn or modify the knowledge base of a FRBS. Fuzzy membership functions, scaling functions and rule base can be stored in the knowledge base of a fuzzy system. In designing GFRBS, either part of or the entire knowledge base can be subject to optimisation by GAs. Figure 4.7 indicates such an integration of GAs with FRBSs.

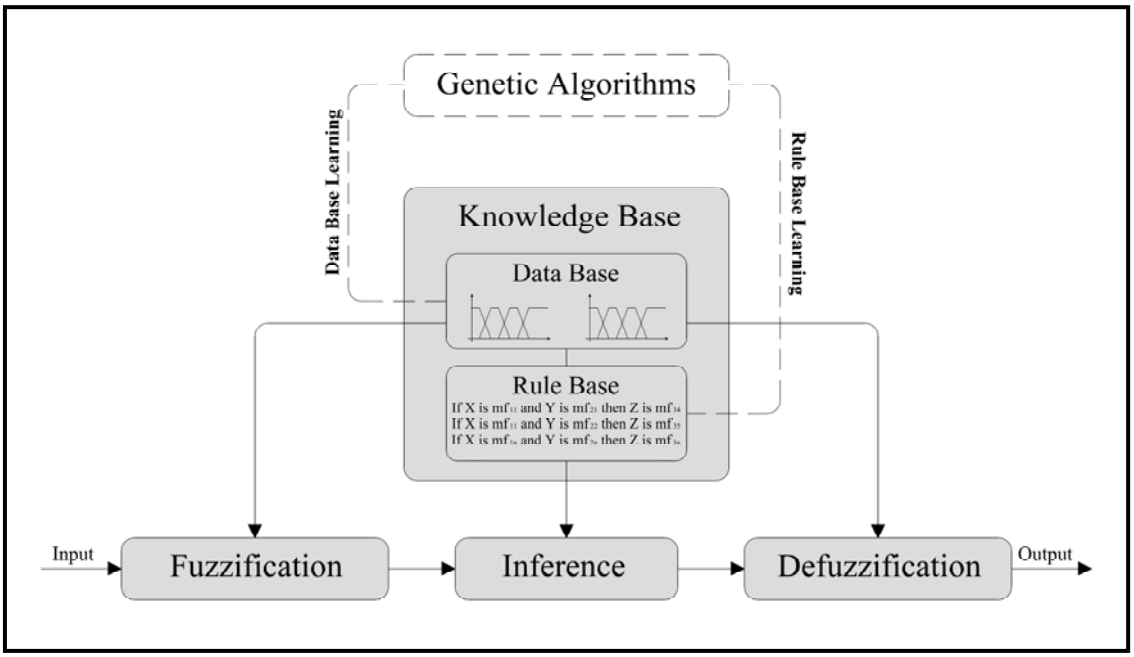

Figure 4.7. Main Components Of Genetic Fuzzy Rule-Based Systems

Two procedures are possible in GFRBS: genetic tuning and genetic learning. In a genetic tuning process, parameters of the data base including membership functions are adjusted using a predefined rule base. In genetic learning process, the performance of the FRBS is improved with knowledge base learning including the rule base. Considering the genetic tuning and learning processes GFRBS designs generally fall into one of four main categories: The present study deals with the second category: learning the rule base of a predefined FRBS with fixed membership functions.

1. Use of GAs to tune up membership functions under a given set of fuzzy rules.
2. Use of GAs to learn the rule base with fixed membership functions.
3. Use of GAs to learn both the database and the rule base simultaneously.
4. Use of GAs to learn the database and the rule base sequentially.

Regardless of the problem, genetic tuning or learning process will be based on evolution. Three issues are essential in the process (Bodenhower and Herrera; 1997):

The population of potential solutions: The population of a GFRBS has to be constituted from one of several components of the FRBS. The individuals in the population may represent partly or completely the parameters of FRBS or membership functions or the rule base.

The set of evolution operators: The genetic learning process has to be evolved into best solution through selection, crossover and mutation operators. Success of an evolutionary learning is based on applying appropriate genetic operators that is compatible with the chromosome representation of the FRBS component.

The performance index: A fitness/error function has to be established in order to measure the difference between the desired and the actual output of the FRBS.

Finally, GAs can be used with various representations in genetic learning of a rule base. These representations generally follow two different approaches: i) the 'chromosome = set of rules", and the 'chromosome = rule" (Herrera, 2008). The first approach, also known as the *Pittsburgh Approach* (Smith, 1980), is selected as the genetic learning strategy of the proposed design. The Pittsburgh approach successfully solves the cooperation versus competition problem by evolving a population of rule bases instead of single rules (Cordon et al.; 2001). However, it brings much greater computational burden which can be solved with improved genetic operators.

CHAPTER 5

EMPIRICAL ANALYSIS

5.1. Description of the Study Area and Data

İstanbul Metropolitan Area has been selected for the case study. It is a very complex and challenging city region to test a trip distribution model. Its transportation system consists a of high number of interaction links, nodes and bridge crossings. Moreover, the production-attraction and the friction matrices of the Istanbul metropolitan area were measured recently, in a large household survey.

Istanbul is located in Northwest Turkey connecting the Marmara and Black Seas and separating the two continents: Asia and Europe. It has a population of nearly 13.8 million, or 17.8% of Turkey's total population (TurkStat, 2010). An estimated 21 million daily trips occur in the Istanbul metropolitan area. 50% of these trips are by foot, 14% by private cars, and 36% by public transit modes. Additionally, 1.3 million daily trips are continent crossings, 1 million on bridges and the remaining 300.000 with ferries (Istanbul Metropolitan Municipality Transportation Planning Depart., 2008).

The data used in this study come from the Household Travel Survey conducted by the Transportation Department of the Istanbul Metropolitan Municipality in 2006. The survey was established in 451 Traffic Analysis Zones (TAZs) covering the entire metropolitan area of Istanbul and including 90.000 households (3% sampling rate). In the survey, which had an 80% unit response rate, approximately 264.000 people in 72.000 households were surveyed and a total of 356.000 trips were recorded between 451 origin-destination pairs.

The observed trips (including both pedestrian and motorized trips) between a possible 203.401 distinct interaction points were categorized by trip purposes. Approximately 127.000 of these trips were for home-based-work (HBW), 94.000 trips were for home-based-school (HBS), 115.000 trips were home-based-other (HBO), and 20.000 trips were non-home-based (NHB) trips (Istanbul Metropolitan Municipality Transportation Planning Department, 2008).

Use of home-based-work (HBW) trips was found sufficient for empirical analysis. The use of the production-attraction (P-A) form of the HBW trip matrix is preferred for the modelling procedure. Figures 5.1 and 5.2 show spatial distribution of zonal production and attraction totals of HBW trips in the Istanbul Metropolitan area. The trip matrix includes both within and between interactions of TAZs. Additionally, assigned travel times are chosen due to typical problems with travel time self-reported in the survey. In summary, the modelling data is constituted from two 451 by 451 matrices: a P-A trip matrix and a travel time matrix.

To divide a single set of data into two representative parts —the first part for the purpose of training and calibration, the second part for the testing of generalization purposes— is an orthodox methodology, especially in NNs and FL based modelling. Therefore, the data matrices are divided into two representative parts, but with an unusual technique.

The calibration of doubly-constrained gravity models requires that the data be in matrix format due to row and column constraints. In addition, taking any number of independent observations or incomplete small matrices from the whole matrix may cause biased parameter estimations. So, the whole matrix is divided into two equal rectangle matrices: a training matrix, and a testing matrix. The training matrix includes trips from all TAZs to odd numbered TAZs (a 451 by 226 matrix); the testing matrix includes trips from all TAZs to even numbered TAZs (a 451 by 225 matrix).

It can be criticized that dividing the data set can reduce the gravity model's performance as the benchmark. However, it has been experimented that the gravity model parameters remain stable even if it is calibrated with smaller amount of samples. The representativeness of samples was found more effective in successfully calibrating gravity models. The representativeness of two datasets is further tested with a two-sample paired t-test. No statistically significant difference was found between training, testing and whole datasets when the trip length distributions (TLDs) were taken into account. Descriptive statistics and observed TLDs of all data sets are shown in Table 5.1 and Figure 5.3.

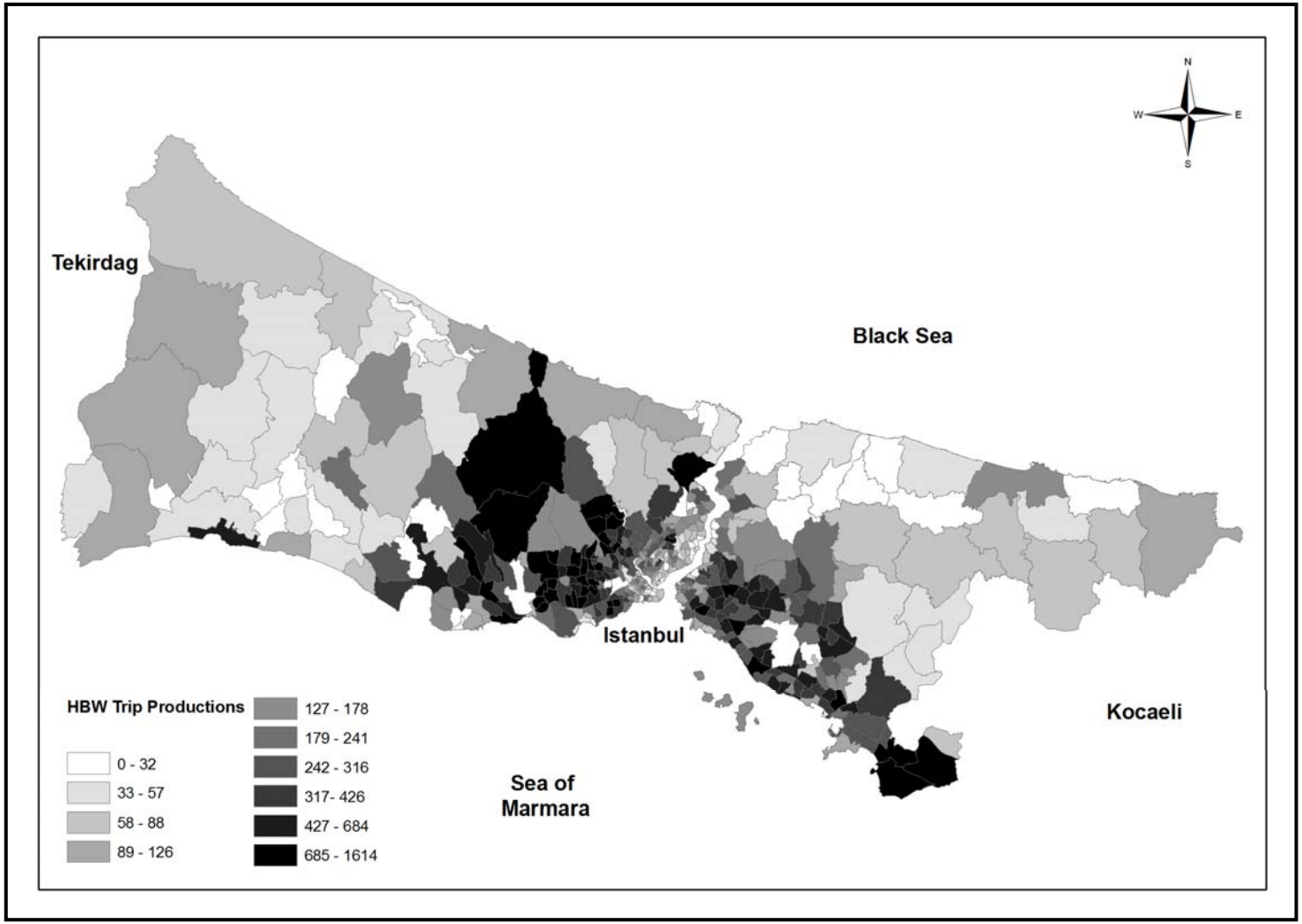

Figure 5.1. Traffic Analysis Zones and Home Based Work Trip Productions
(Source: Istanbul Metropolitan Municipality Transportation Planning Department)

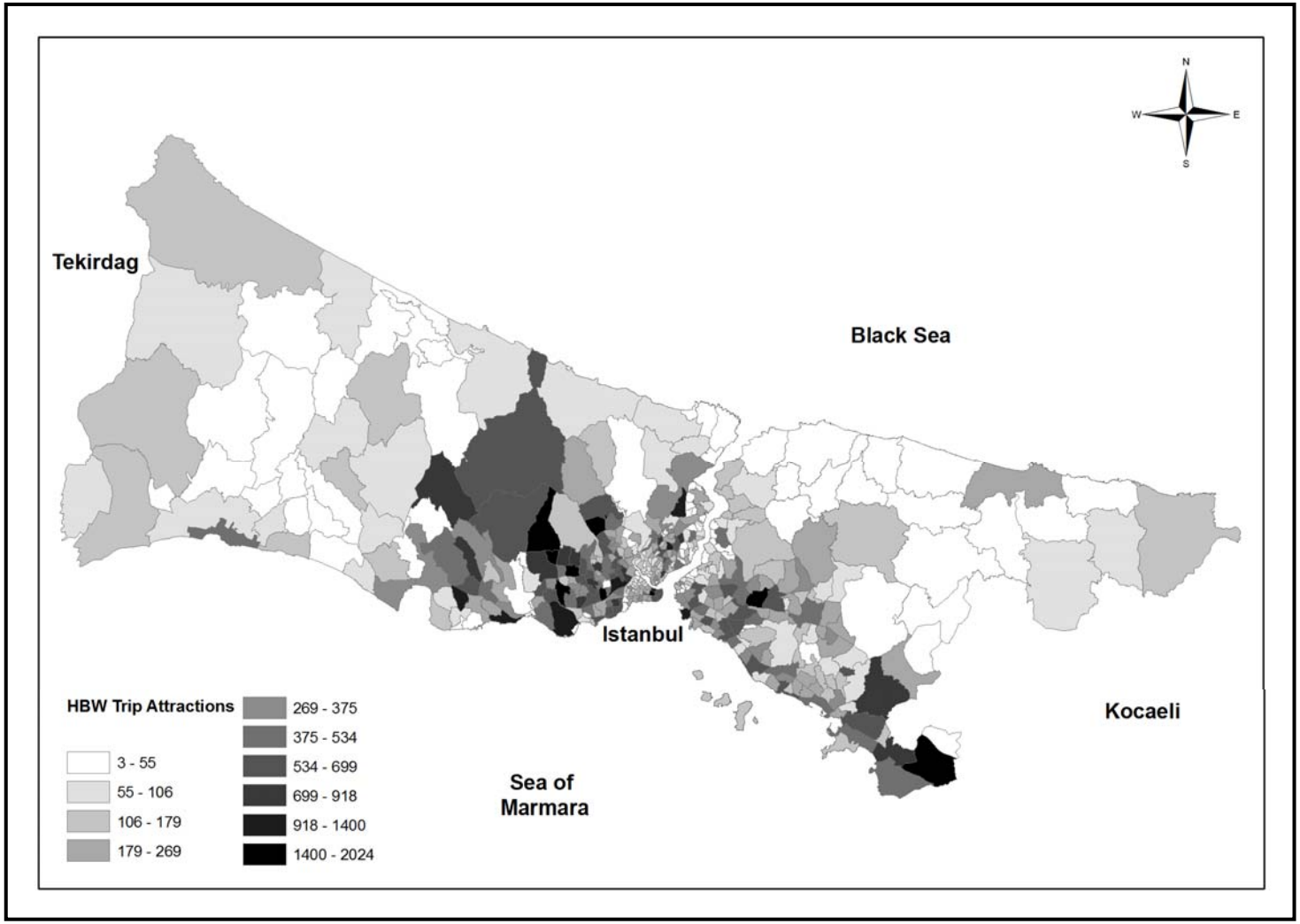

Figure 5.2. Traffic Analysis Zones and Home Based Work Trip Attractions
(Source: Istanbul Metropolitan Municipality Transportation Planning Department)

Table 5.1. Descriptive Statistics of Data Sets

Data Sets	Number of Elements	Mean	Std. Deviation	Minimum	Maximum
Full Data Set					
Production Vector	451	283	307	2	1614
Attraction Vector	451	283	329	3	2024
Travel Time Matrix	203401	54	37	0.17	299
Trip Matrix	203401	0.63	6.2	0	885
Training Data Set					
Production Vector	451	139	172	1	1171
Attraction Vector	226	278	339	4	1952
Travel Time Matrix	101926	54	38	0.17	299
Trip Matrix	101926	0.62	6.8	0	885
Testing Data Set					
Production Vector	451	144	156	1	1009
Attraction Vector	225	288	318	3	2024
Travel Time Matrix	101475	54	37	0.17	275
Trip Matrix	101475	0.64	5.6	0	454

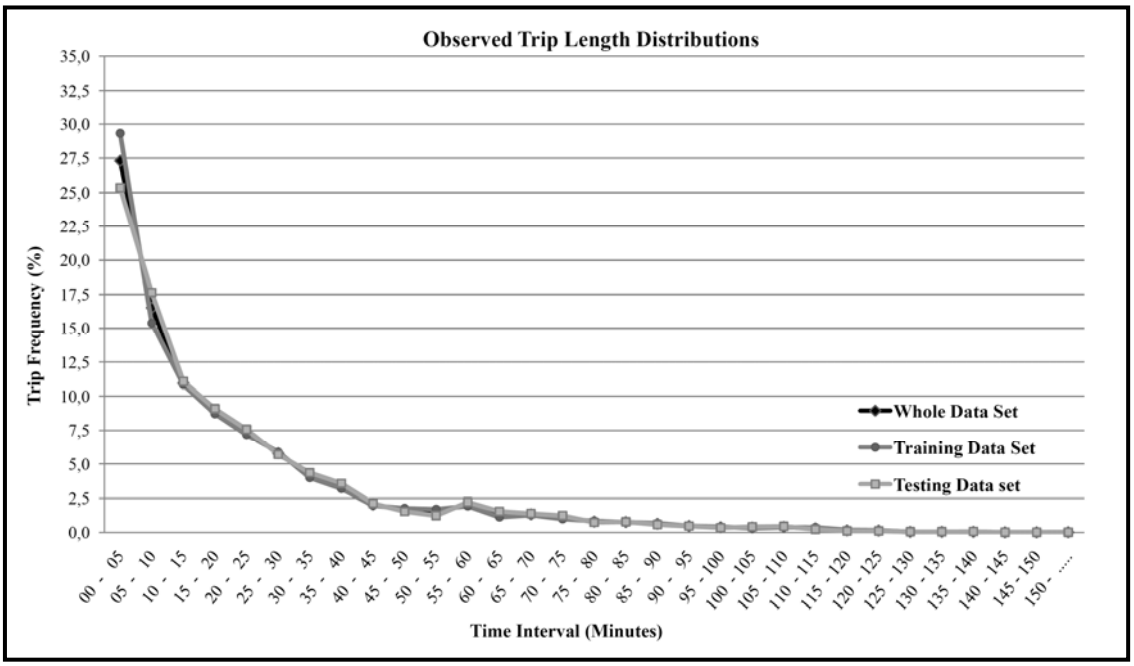

Figure 5.3. Observed Trip Length Distributions of Data Sets

5.2. Modelling Trip Distribution with a Fuzzy Rule-Based System (FRBS)

Fuzzy rule-based systems (FRBSs) are useful in two general contexts: i) in situations involving uncertainty, imprecision and partial truth, and ii) in situations where investigators are mapping any inputs into desired outputs even if there is no uncertainty and imprecision present. In this study we propose a Mamdani-type FRBS designed in the latter context to solve urban trip distribution problem.

The main logic behind the proposed FRBS design and its variable structure are the same as with the classical gravity model including three inputs and one output. Considering an origin-destination zone pair, if zonal *trip productions* and *trip attractions* are known along with the corresponding *friction factor*, the number of *interactions/trips* between this zone pair can be estimated using a FRBS as a universal approximator. Its main structure can be established starting from the simple verbal statements as follows:

- IF total trip production of the origin is LOW, AND total trip attraction of the destination is LOW, AND friction factor between corresponding origin and destination is HIGH, THEN the interactions/trips between origin and destination is LOW;

- IF total trip production of the origin is HIGH, AND total trip attraction of the destination is HIGH, AND friction factor between corresponding origin and destination is LOW, THEN the interactions/trips between origin and destination is HIGH.

In order to construct a FRBS, these verbal statements are to be decomposed into a set of overlapping fuzzy sets connected to If-Then rules with logical operators. There are several ways exist to establish fuzzy partitions and fuzzy rule base.

The present study deals with designing a simple and effective FRBS that is regardful to the accuracy-interpretability trade-off. Therefore, a heuristic design of fuzzy sets with few partitions is preferred for simplicity. Then, a widely-used *Wang-Mendel method* (1992), also known as the *one-pass method*, is implied as the fuzzy rule induction procedure. The following five steps describe the construction and training of the proposed FRBS design:

(1) Divide input-output spaces into overlapping regions:

The input-output pairs of trip distribution problem can be stated as,

$$(P_1, A_1, F_1; T_1), (P_2, A_2, F_2; T_2), \ldots, (P_i, A_i, F_i; T_i)$$

where P, A, and F (production, attraction, and friction) represent input and T (trips) represent output variables. Each variable has a domain interval which lies between minimum and maximum values. The domain intervals are divided into a prespecified number of subintervals. Number and lengths of these subintervals are determined with intuition and visual inspection. The production and the attraction variables are divided into 5, the friction variable is divided into 6, and the output variable, trips, is divided into 20 fuzzy sets. The fuzzy sets are labelled with numbered Membership Functions (MFs) representing low, moderate and high quantities roughly. For simplicity, the first and the last MFs are established as semi-trapezoidal and the others are set as triangular. Figure 5.4 indicates an illustration of originally scaled fuzzy sets used in the study.

(2) Generate fuzzy rule candidates from numerical data:

In this step, membership degrees [$\mu(x_i)$] of each input-output point are evaluated, and then MFs having maximum degrees are assigned as a rule candidate. Suppose that membership degrees evaluated for any pair of data are indicated as following:

$$\mu_{(P_i,A_i,F_i;T_i)} \rightarrow \begin{bmatrix} P_{\mathrm{i}}(0.7 \text{ in } \mathrm{MF}_1; 0.4 \text{ in } \mathrm{MF}_2), A_{\mathrm{i}}(0.8 \text{ in } \mathrm{MF}_1; 0.2 \text{ in } \mathrm{MF}_2), \\ F_{\mathrm{i}}(0.6 \text{ in } \mathrm{MF}_1; 0.3 \text{ in } \mathrm{MF}_2); T_{\mathrm{i}}(0.9 \text{ in } \mathrm{MF}_2; 0.2 \text{ in } \mathrm{MF}_3) \end{bmatrix}$$

Assigning the corresponding MFs with maximum degree to logical If-Then structure constitutes the i^{th} rule candidate as:

$$i^{th} \text{ rule candidate} \rightarrow IF\ P_i \text{ is } MF_1 \text{ and } A_i \text{ is } MF_1 \text{ and } F_i \text{ is } MF_1,$$
$$THEN\ T_i \text{ is } MF_2$$

(3) Select one desired rule among conflicting rules:

Three input structure, P, A, and F with 5, 5 and 6 fuzzy partitions respectively, enable 150 (5*5*6) different rule antecedents (IF part of a rule) with logical 'and' connections. However, the training data set consists of 101926 observed input-output pairs leading to a great number of conflicting rules: the rules having same

antecedents but different consequents (THEN part of a rule). The traditional approach of the original Wang-Mendel method evaluates strength of the each rule candidate with multiplying their membership degrees, and then selects a final rule having the maximum strength. Since, this approach does not satisfy with our case due to high number of the same rule candidates, this step is processed in a different way: first, the weighted average of conflicting rule consequents are computed using their observed frequencies, then the nearest consequent part to the computed weighted average is selected as the final rule. For instance, if three different rule consequents are observed with various frequencies as in the following example:

IF P_i is $MF_1$and A_i is MF_1 and F_i is MF_1, THEN T_i is $\boldsymbol{MF_2}$ → frequency 25
THEN T_i is $\boldsymbol{MF_3}$ → frequency 35
THEN T_i is $\boldsymbol{MF_5}$ → frequency 20

a final rule consequent (MF label number) can be selected computing their weighted average as in the following expression:

$$n_{wa} = \frac{(n_1 * f_1) + (n_2 * f_2) + \cdots + (n_k * f_k)}{f_1 + f_2 + \cdots + f_k} => \frac{(2 * 25) + (3 * 35) + (5 * 20)}{25 + 35 + 20} \cong 3$$

where, n_k and f_k represent observed label number of a MF and corresponding frequency, and n_{wa} represents the weighted average of the conflicting rule consequents. Then, the final statement of the rule in the above example should end with $\boldsymbol{MF_3}$, since the computed n_{wa} equals approximately 3.

(4) Combine selected fuzzy rules and generate fuzzy rule base:

This step is the final step of an automated fuzzy rule base learning procedure. Steps one through four are automated with a self-created program written in the MATLAB environment. In this step, all single rules, selected in the previous step, are combined establishing the initial fuzzy rule base.

(5) Check out fuzzy rule base and make a limited number of changes:

Approximately 95% of the rule base has been directly generated from numerical input-output pairs with the first four steps. Then, non-observed rules are extrapolated making an analogy to neighbourhood rules having similar

antecedents. Finally, the whole rule base has been checked out logically and a limited number of rules (10%) have been changed to improve the generalizability of the system. Table 5.2 indicates the latest appearance of the rule base.

Table 5.2. An Appearance From The Constructed Rule Base

Rule Number	**Antecedents**	**Consequents**
Rule 1	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_3
Rule 2	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_3
Rule 3	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_4
.....	.....	.....
Rule 23	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{18}
Rule 24	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{19}
Rule 25	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{20}
.....	.....	.....
Rule 148	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 149	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 150	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_2

After having established the MFs and the rule base, the next step is to select the implementation techniques. The implication procedure of the proposed FRBS for both training and testing purposes is described in the following steps:

Step 1—Initialization: Initialize data and normalize input-output spaces into some appropriate range when it is required. Constitute the knowledge base.
Step 2—Fuzzification: Compute fuzzy membership degrees of actual inputs using corresponding membership functions: crisp into fuzzy values.
Step 3—Inference: Combine fuzzy sets with logical operators with appropriate implication algorithm. Aggregate all outcomes to generate fuzzy output. In this step, both of the *Max-Min* and the *Max-Product* techniques are tried and the *Max-Product* implication is selected.
Step 4—Defuzzification: Convert aggregated fuzzy outputs into crisp outputs using a defuzzification method. In this step, the *Centroid Defuzzification* method has been selected and implied within various defuzzification techniques.

Graphical interpretation of this procedure is beneficial to understand nature of fuzzy trip distribution modelling. The following illustration in Figure 5.4 shows the original components and scales of the proposed FRBS design. It produces unconstrained trip interactions as output. Before using it for simulation purposes, we had to ensure that the results satisfy production and attraction constraints. Therefore, the results of the FRBS were adjusted with a row-column balancing process in each simulation of data sets. A numerical example of such balancing process, which is similar to the well-known *Furness Iterations* (1965), can be seen in Ease (1993).

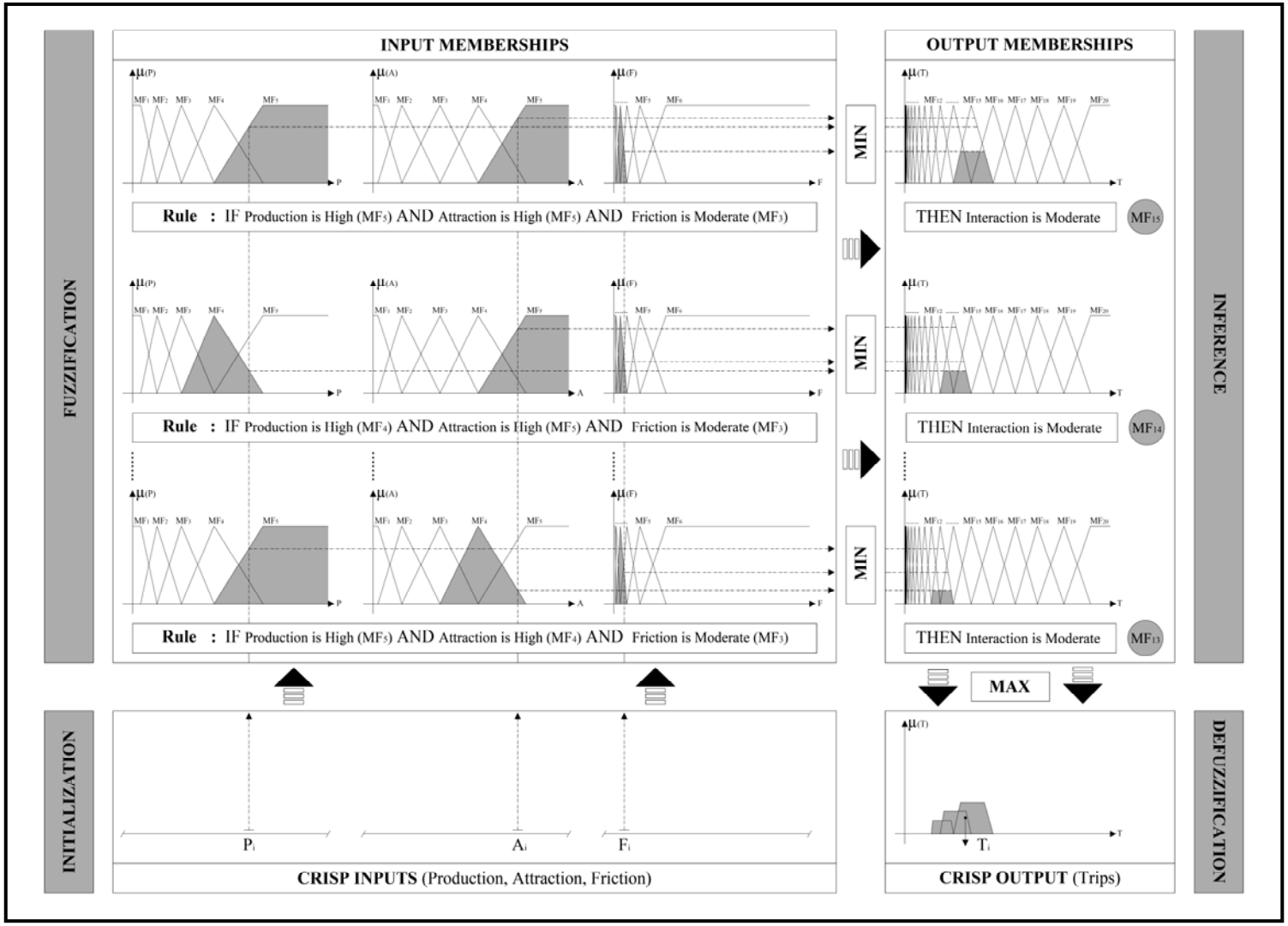

Figure 5.4. Graphical Illustration Of The Proposed FRBS Design

5.3. Modelling Trip Distribution with a Genetic Fuzzy Rule-Based System (GFRBS)

An FRBS to solve trip distribution problem was introduced in the previous section. Its rule base was constructed with a mixed procedure including both learning from examples and expertise. Here, the rule base of the proposed FRBS is learnt completely from examples with the use of GAs. All other components of the proposed FRBS remain unchanged . In other words, the proposed GA search for the best combination of rule consequents or output MF labels represented with gray circles in Figure 5.4.

Initially, a *Simple GA* (Goldberg, 1989) was developed with basic genetic operators and binary representation. However, due to the large combinatorial search space and huge amount of data, the convergence failed, and some additional modifications have been introduced to improve performance of the algorithm. Keeping the main flow chart and its binary representation, several probabilistic and adaptive features were introduced to the genetic operators. The flow chart in Figure 5.5 indicates main steps of the proposed GFRBS, fully automated and programmed with MATLAB. A brief description of the whole procedure is given afterwards.

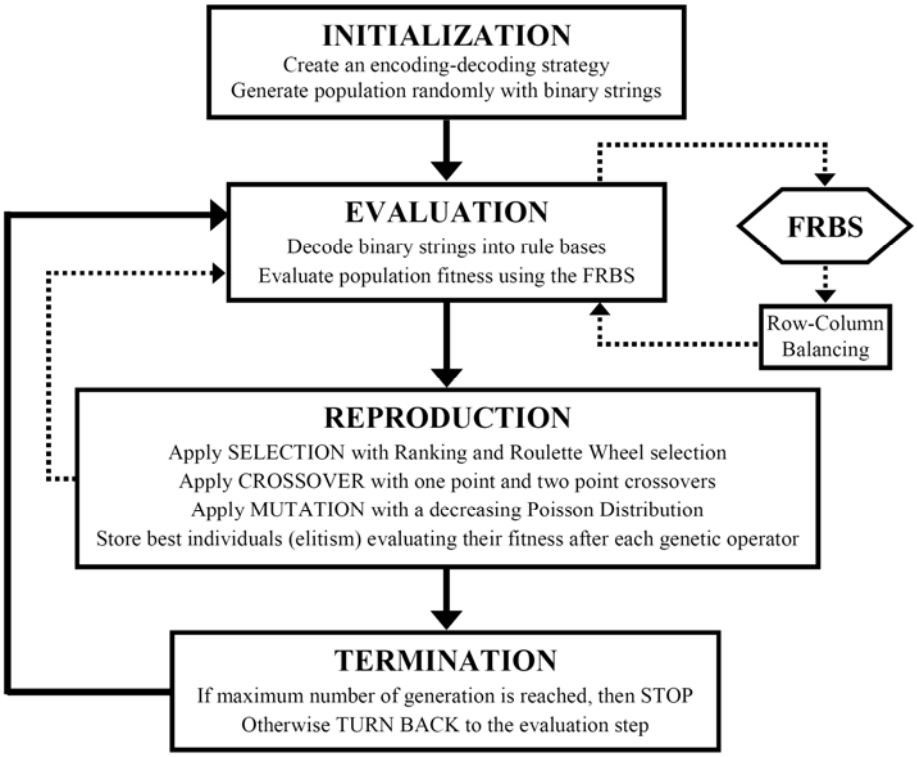

Figure 5.5. Flow Chart of the Proposed Genetic Algorithm

Initialization: The initial population of the GA consists of randomly generated 20 binary chromosomes encoding the whole rule set. The chromosomes have 450 binary digits where 3 digits were assigned for each of the 150 rule consequents. Meaning that a rule can end with one of 8 (2^3) alternatives. Actually, there are 20 output MFs in the proposed FRBS, however, a few of shifting consequents are meaningful for each rule antecedents. Eventually, maximum number of alternatives was restricted to 8 in order to reduce search space and save time.

The alternatives were obtained from available knowledge according to the conflicting rules of FRBS design. The most frequently observed or most probable 2, 4 or 8 rule consequents were identified for each rule antecedent and collected in a pool. With this rule pool, 45 rules can end with one of 2, 65 rules can end with one of 4, and 40 rules can end with one of 8 MF alternatives constituting $6.4*10^{88}$ ($2^{45}*4^{65}*8^{40}$) possible rule sets. Figure 5.6 shows the whole encoding-decoding strategy including a view from the rule base and the rule pool.

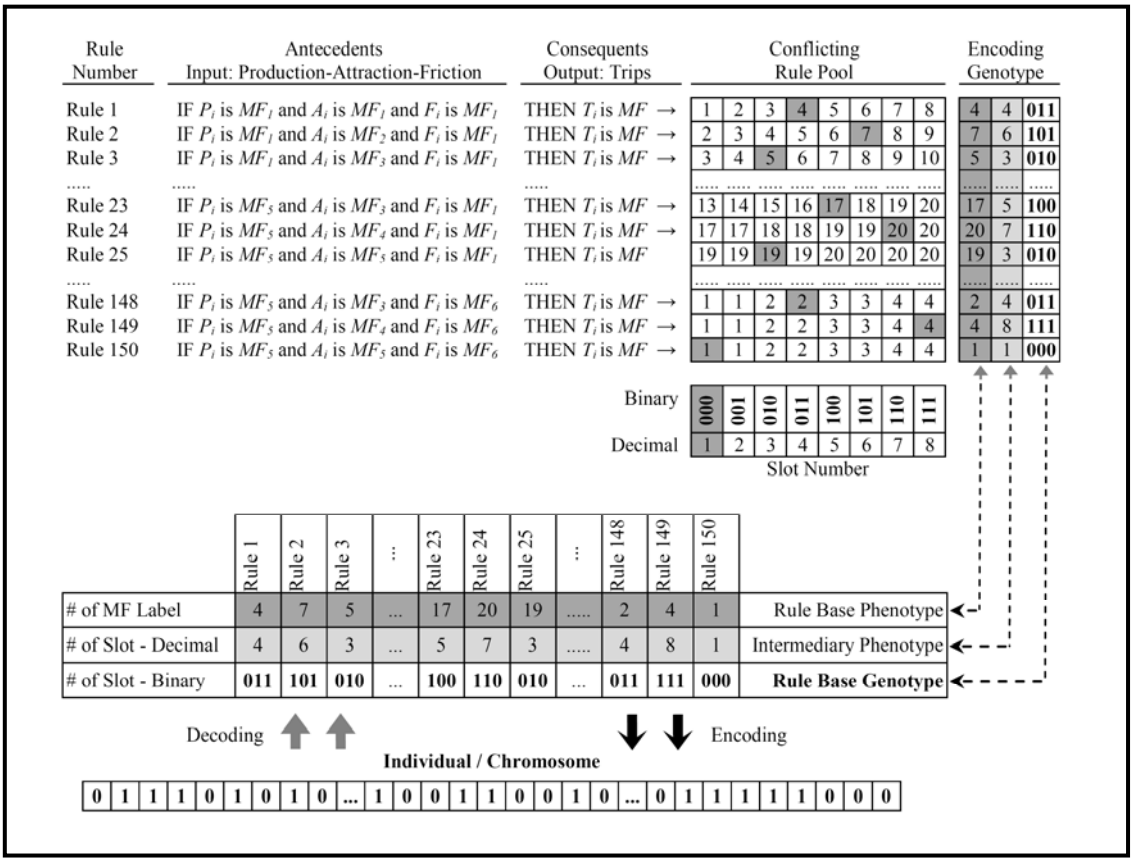

Figure 5.6. Graphical Representation Of Encoding-Decoding Strategy

Evaluation: In the evaluation step, the binary strings were decoded into the rule sets to run the FRBS. Then output values of the FRBS for each of the rule set were compared with the actual values. The *mean square error* (MSE) was used as the fitness (error) function:

$$MSE = \frac{\sum_{i=1}^{N}(T_i^0 - T_i)^2}{N} \tag{5.1}$$

where, N is the number of data pairs, T_i^0 is the number of observed trips, and T_i is the number of estimated trips. The FRBS estimated trips were also balanced to satisfy the constraints before making the comparisons.

Reproduction: Generally in GAs, successive generations of the population should base on transferring the best chromosomes to next generation (selection), and then improving them with gene exchanges (crossover) and gene alterations (mutation). The proposed GA was designed with this three step evolution process described as follows:

- *Selection:* In this step 'successful' chromosomes, the parents, were copied to a mating pool, then selected for crossover and mutation according to some measure of their fitness. There are number of ways to choose the parent population. A mixed procedure was implemented in order to improve the convergence performance: Firstly, a *ranking* was applied, in which chromosomes are ranked and assigned proportions only on their rank orders, not on their absolute fitness. Then each mating parents were selected with a biased *roulette*-wheel for recombination. The slots of the wheel was divided according to the ranking proportions determined with a power function (see Figure 5.7). Apart from these, when generating new populations with genetic selection, crossover and mutation operators an *elitist* strategy was developed. The few best chromosomes (10% of total population) of the former generations were directly copied to next generations through genetic operators. This strategy significantly improved the GA's performance preventing it from loss of good solutions.
- *Crossover:* In the crossover step, new chromosomes (offspring) were created by recombining two parent chromosomes with a certain probability (0.8). Classical *one point* or *two point* crossover were employed to the parent chromosomes with equal probability. In one point crossover, the algorithm chooses a point at random, called the crossover point, and exchanges the contents to the right of

this point; in two point crossover, the algorithm chooses two points, and exchanges the contents between these points. Figure 5.7 indicates an illustration of adopted crossover technique used in the GA.

- *Mutation:* In order to achieve faster results and prevent the algorithm from a premature convergence, following two components were introduced to the classical mutation operator. First, the assumption of a constant probability of mutation in each generation is abandoned in favour of an adaptive one. With high mutation probability at the startup, the population attacked to get out of from local optima; and with low probability at the end, the population resembled each other to find out small improvements. Second, the simple bit-flip mutation operator has been replaced with a probabilistic (*Poisson distribution*) mutation operator. This approach is an efficient and time-saving alternative of simple bit-flip mutation. In this approach, the average number of mutations (λ=72) in each generation is determined automatically, multiplying the population size (20) , chromosome length(450) and mutation probability (0.008). Then number of mutations in various generations is determined with a decreasing Poisson distribution at the start up. Finally, only in that number of randomly selected bits in the whole population is changed (inverted) through the generations. A graph that shows mutations through the generations can be seen in Figure 5.7.

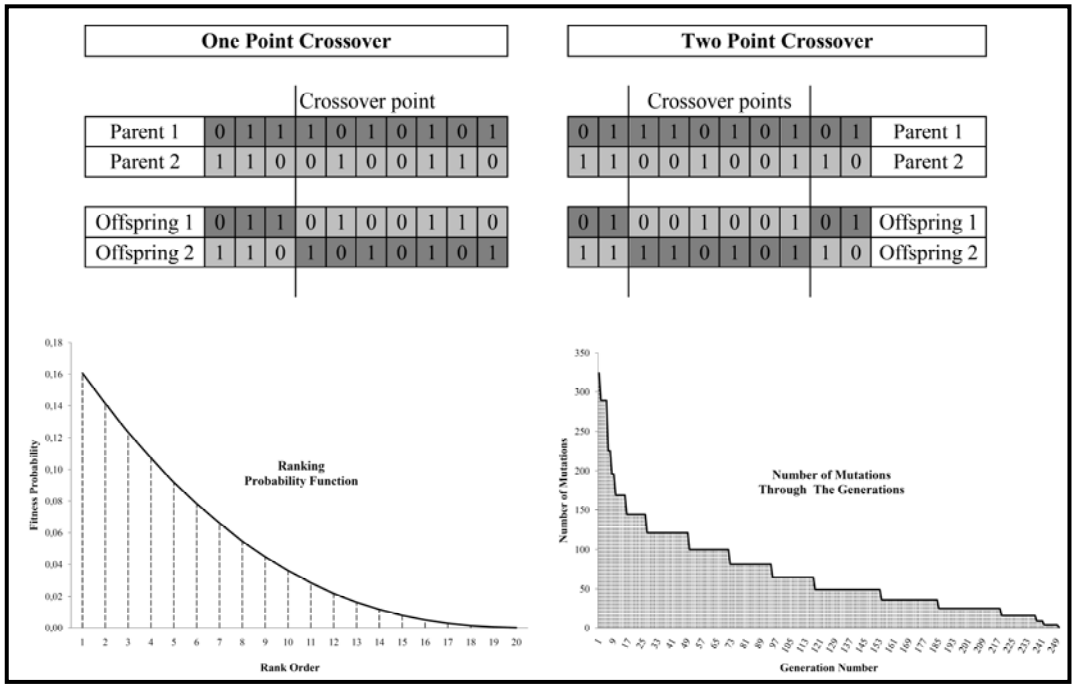

Figure 5.7. Illustrations Of Crossover, Ranking Probability Function And Number Of Mutations Through The Generations

Termination: Any early stopping of the GA was not seen necessary as the termination criteria. We rather limited the evolution of the population up to 250 maximum generations.

With the above procedure, the GFRBS design successfully converged to the best solution. Then it was used for simulation purposes with the optimized rule base. Figure 5.8 indicates its convergence with both of the progress of population average and best individual.

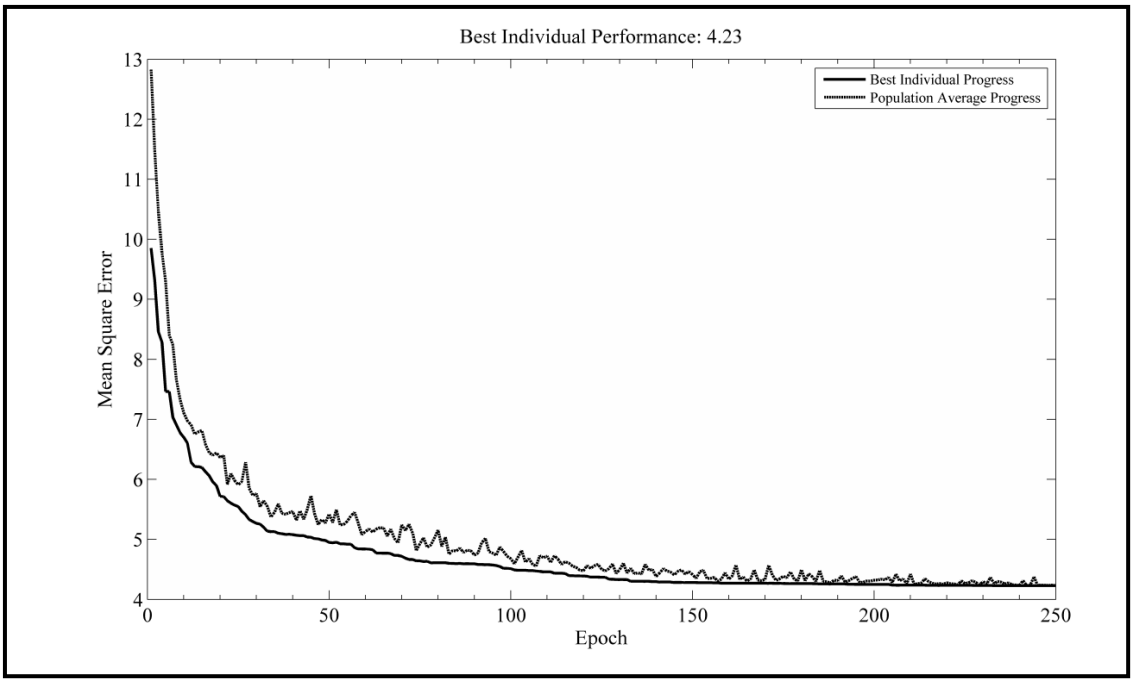

Figure 5..8. Convergence Of The GFRBS Design

5.4. Benchmark Models

The Doubly-Constrained Gravity Model (DCGM), introduced earlier, was selected as the first benchmark model. A Neural Network Based Trip Distribution Model (NNTDM) was established as the second benchmark. Even though debate continues, NNs-based distribution models have a history of successful use, and it is appropriate to establish a NNs-based model and compare its overall performance with the proposed FRBS and GFRBS designs. The next sections of this paper cover calibration, training, and implementation issues for these benchmark models.

5.4.1. Doubly-Constraint Gravity Model (DCGM)

Traditional DCGM calibration process generally involves the determination of the friction function parameter(s) introduced earlier. Many studies have suggested statistical or numerical computational procedures to calibrate these friction parameters (Hyman, 1969; Wilson, 1970; Evans, 1971; Williams, 1976; Sen and Soot, 1981; Gray and Sen, 1983, Dickey, 1983; Sen, 1986). Among these procedures, statistical least squares and maximum likelihood approaches, and TLD based numerical approaches are reasonably found efficient for our analysis.

The regression based techniques and maximum likelihood approaches have well-known desirable statistical properties, and they have consistently been proven the calibration abilities. The TLD based numerical approaches have also some advantages, especially when a great number of inter-zonal trips are missing as in our case. In these cases, a macro level measure of the interactions, such as trip length frequency distributions or mean travel cost, would enable a better understanding of the systems' behaviour. The selected three principally distinct calibration procedures are listed below and described with specific formulations next.

- *The Maximum Likelihood Estimation* which maximizes the likelihood function of a theoretical poison distribution of interactions and is described in Sen (1986) and Fortheringham and O'kelly (1989),
- *The Weighted Least Squares Estimation* which is based on the 'odds ratio technique' and logarithmic transformation proposed by Sen and Soot (1981) for rectangular interaction matrices,
- *The Trip Length Distribution Based Estimation* which is based on a line search algorithm that minimizes the root mean squared error (RMSE) between observed and estimated trip length frequency distributions.

<u>Maximum Likelihood (ML) Estimation:</u>

ML estimations are widely used in gravity type of model calibrations. The main steps involved in the estimation include: (i) identifying a theoretical distribution for the interactions, generally a Poisson distribution (Flowerdew and Aitkin, 1982; Sen 1986), (ii) maximizing the likelihood function of this distribution, and (iii) then deriving

equations that ensure the maximization of the likelihood function with its logarithmic transformations (Fortheringham and O'kelly, 1989). The derived ML equation for power cost function parameter, say beta (β), is:

$$\sum_i \sum_j T_{ij}^0 \ln (c_{ij}) = \sum_i \sum_j T_{ij} \ln (c_{ij}) \tag{5.2}$$

It is possible to derive the same equation using Wilson's (1970) entropy-maximizing or Hyman's (1969) Bayesian approaches. The equation can be solved iteratively starting with an initial estimate of beta value, β_0. Then, the use of DCGM equations, initiated in equations 2-17, 2-18 and 2-19, produce an estimated trip matrix. The value is then gradually decreased or increased through a convergence that satisfies the ML equation 5-2. Hyman's (1969) suggestion for the initial β value, and his second order formula for rest of the β values produce rapid convergence to final estimate as:

$$\beta_0 = 3/2\bar{C} \tag{5.3}$$

$$\beta_1 = \beta_0 \bar{C}_0 / \bar{C} \tag{5.4}$$

$$\beta_{k+1} = \frac{(\bar{C}-\bar{C}_{k-1})\beta_k - (\bar{C}-\bar{C}_k)\beta_{k-1}}{(\bar{C}_k-\bar{C}_{k-1})} \tag{5.5}$$

where $\bar{C}$ is the observed mean travel cost and the $\bar{C}_k$ is the estimated mean travel cost using DCGM equations with β_k.

Weighted Least Squares (WLS) Estimation:

The regression based calibration of spatial interaction models is required a linearized model form. The unconstrained models can easily be linearized with logarithmic transformations, then the parameter values can be computed with ordinarily least squares estimation. Unfortunately, it is a bit more complicated for constrained models. The 'odds ratio technique' for calibration of DCGMs provided by Sen and Soot (1981) and Gray and Sen (1983) is a valuable contribution at this point.

The technique separates the estimation of the friction parameters from the calculation of balancing factors, then takes ratios of interactions so that the $A_i O_i$ and

$B_j D_j$ terms in the model can be canceled out (Fortheringham and O'kelly, 1989). With a power cost function, the equation takes the form of:

$$\left(T_{ij}/T_{ii}\right).\left(T_{ji}/T_{jj}\right) = \left(c_{ij}/c_{ii}\right).\left(c_{ji}/c_{jj}\right)^{-\beta} \tag{5.6}$$

where, with a logarithmic transformation, its linearized form is,

$$lnT_{ij} + lnT_{ii} - lnT_{ij} - lnT_{ii} = -\beta(lnc_{ij} + lnc_{ii} - lnc_{ij} - lnc_{ii}) \tag{5.7}$$

Once the β parameter is estimated with ordinarily least squares or weighted least squares (WLS) with the weight being $\left(T_{ij}^{-1} + T_{ii}^{-1} + T_{ji}^{-1} + T_{jj}^{-1}\right)^{-0.5}$ (sen and Soot, 1981), the balancing factors can easily been computed from the equations 2-18 and 2-19. However, our data matrices are rectangular and cannot be computed with the above equations. Sen and Soot (1981) also proposed an alternative method of transformation for equation 5-8, which can also be used in calibrating our matrices.

$$\begin{aligned} lnT_{ij} - \left(\frac{1}{n}\right)\sum_j lnT_{ij} - \left(\frac{1}{m}\right)\sum_i lnT_{ij} + \left(\frac{1}{mn}\right)\sum_i\sum_j lnT_{ij} \\ = -\beta\left[lnc_{ij} - \left(\frac{1}{n}\right)\sum_j lnc_{ij} - \left(\frac{1}{m}\right)\sum_i lnc_{ij} + \left(\frac{1}{mn}\right)\sum_i\sum_j lnc_{ij}\right] \end{aligned} \tag{5.8}$$

In its application, a weighted least squares estimation technique with the weight being $T_{ij}^{0.5}$ is preferred to exclude the heteroscedastic error terms caused by the logarithmic transformations of equation 5-8.

Trip Length Distribution (TLD) Based Estimation:

Several search procedures are exist, especially for the one-parameter functions minimization. However, one of the simplest procedure is to run the model for a wide range of β values, and chose the best β value that optimizes a predetermined goodness-of-fit statistic (Wilson, 1974).

The proposed TLD based estimation uses a simple line search algorithm to find the best value of β. Firstly, trip matrices are estimated using β values in the search interval (0-4), then TLD of these matrices are computed, and finally, observed and estimated TLDs are compared with the root mean squared error (RMSE). The β value

with the lowest RMSE score determined as the impedance parameter. The RMSEs between observed and estimated TLDs are computed as in equation 5.22 which is described in goodness-of-fit statistics section.

During the calibration process, the use of combined gamma function is not concerned. Because Istanbul is a pedestrian oriented city, and the home-based-work trips data includes both motorized and pedestrian travels. This situation gives shape to the trip length frequency distribution (see Figure 5.3) where the frequencies decrease continuously starting from the first time intervals. So, the use of power and exponential cost functions other than combined gamma function would produce more appropriate distributions. Including these two cost functions, each of the three calibration procedures are applied to traditional DCGM initiated in equations 2.17, 2.18 and 2.19. All calibration procedures and algorithms are created in MATLAB programming environment, and the impedance parameters (say beta) of each procedures are computed using the training data set. The computer codes of all calibration procedures can be seen in Appendix A.

Afterwards, using calibrated impedance parameters, trip matrices of power and exponential cost functions are estimated and compared with various goodness-of-fit statistics (see the statistics in section 5.5). The overall results with training data set show that the use of power cost function produces considerably more accurate estimations for the Istanbul case. The results for both parameter estimates and related goodness-fit-statistics are shown numerically in Table 5.3. The changes in DCGM performance with respect to various parameter values are shown visually in Figures 5.9 and 5.10.

Table 5.3. DCGM Parameter Estimates and Related Goodness of Fit Statistics for Training Data Set

DCGM Calibration Procedure	**Beta Parameter**	**SRMSE**	**r square**	**Slope**	**ARV**	**Phi Statistic**	**MTCE**	**TLD RMSE**	**ARAE TLD F5**	**ARAE TLD L5**
Power Cost Function										
ML Est.	**-1.94**	4.17	0.85	0.99	0.17	0.94	-2.77	0.10	0.10	9.46
WLS Est.	**-2.05**	4.16	0.86	0.94	0.17	0.96	-1.65	0.18	0.14	8.92
TLD Based Est.	**-1.84**	4.29	0.85	1.06	0.18	0.93	-4.18	0.07	0.07	10.56
Exponential Cost Function										
ML Est.	**-0.12**	6.78	0.64	1.15	0.45	1.18	0.17	0.62	0.48	0.77
WLS Est.	**-0.21**	5.94	0.72	0.88	0.34	1.49	4.45	0.45	0.35	0.91
TLD Based Est.	**-0.31**	6.12	0.76	0.76	0.36	2.11	6.45	0.40	0.30	0.94

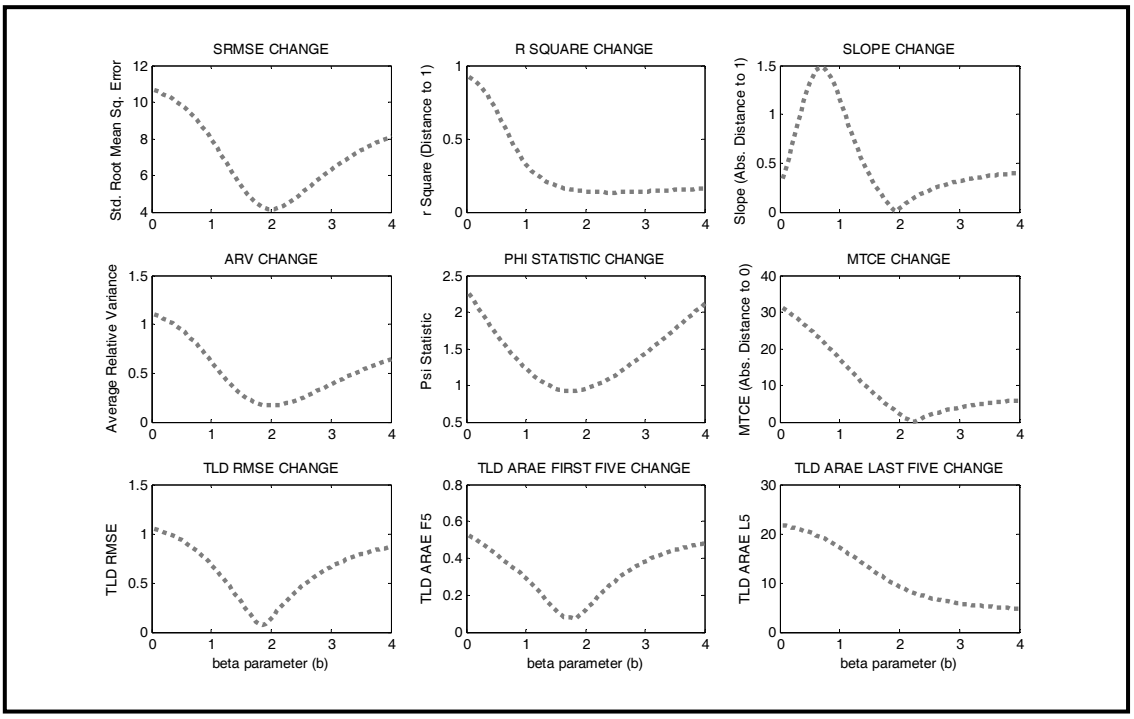

Figure 5.9. Changes in DCGM Performance Against Various Impedance Parameter Values: Measure for The Power Cost Function on Training Data Set

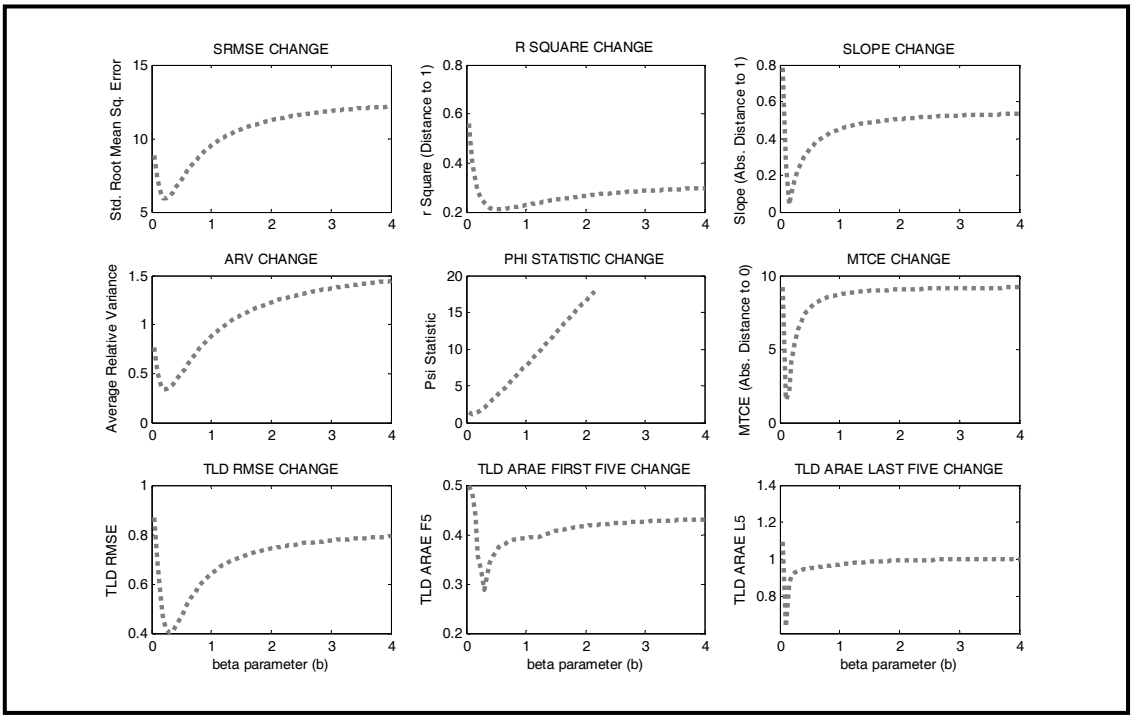

Figure 5.10. Changes in DCGM Performance Against Various Impedance Parameter Values: Measure for The Exponential Cost Function on Training Data Set

5.4.2. Neural Networks Based Trip Distribution Model (NNTDM)

The natural structure of NN based spatial interaction models involves three inputs (production, attraction and friction) and one output (interactions) as in traditional spatial interaction problem. Also called as "Neural Spatial Interaction Models" (see Fischer, 2001; 2009), they are more closely related to conventional spatial interaction models then they are to neurobiological models (Fischer, 2003). There are many types of NN models with various functionality and architecture. However, the subject of the past studies mentioned earlier and this paper also is a multilayer feed-forward network with (error) back-propagation training algorithm.

A multilayer feed-forward network generally consists of one input, one or more hidden and one output layers, where all the neurons in the layers transfer the information through only the neurons in the next layer. The weights, assigned to the links between neurons, are adjusted to minimize the mean square error between the networks' output values and actual target values in an iterative manner. This process is also a supervised learning process, generally adopted with a gradient descent method, the so-called (error) back-propagation training algorithm (see Rumelhart et.al.,1986).

A general illustration of a Neural Network based Trip Distribution Model (NNTDM), which is also used in this study, is presented at Figure 5.11. The number of neurons in the hidden layers, the number of hidden layers, type of transfer functions and learning algorithms as well as the number of inputs can be changed for the proper use. A brief explanation of the mathematical procedure for this three-layer feed-forward back-propagation network is given with mathematical expressions as follows. Please see in depth theoretical explanations in Munakata (2008) and Haykin (1999).

Each neurons in the proposed network (Figure 5.11) are composed of two units. The first unit evaluates the weighted sum of input signals, and the second unit transfers this weighted sum through the next layer. The most widely used transfer functions are sigmoidal functions, which also ensure the non-linear mapping of the network. With a logistic sigmoid activation functions in the hidden and output layers, the network outputs can be computed as following:

$$H_j = {}^{1}/_{1 + e^{-(\sum w_{ij} x_i)}} \tag{5.9}$$

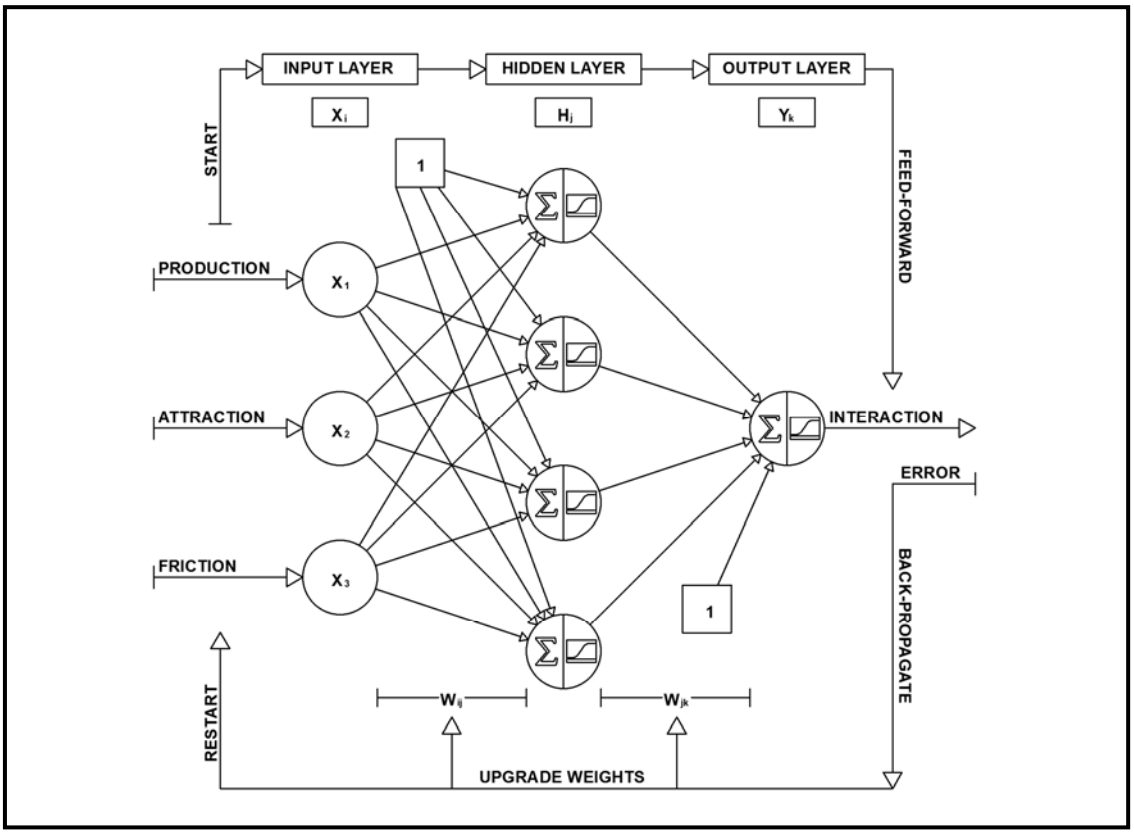

Figure 5.11. An Illustration of NN based Trip Distribution Model: A Three-Layer Feed-Forward Neural Network with Error Back-Propagation

where, X_i is the i^{th} element of the input vector, W_{ij} is the weight associated with the input and hidden layer neurons, H_j is the output signal of j^{th} neuron at hidden layer, and,

$$Y_k = {}^{1}/_{1 + e^{-(\sum W_{jk} H_j)}} \tag{5.10}$$

where, W_{jk} is the weight associated with hidden and output layer neurons, Y_k is the output signal of k^{th} neuron at output layer, which stands for "interactions" for this network. Note that there is an additional (imaginary - bias) neuron in each of the input and hidden layers, which accelerates the network's learning. These neurons are generally set as to 1, and are computed as other neurons in the related layers. If the optimal values of weights are already known, the network is ready to use for modeling purposes. If not, it is needed to train the network and adjust weights to minimize the error between observed and modeled outputs. Overall error, E, is usually computed as,

$$E = {}^{1}/_{2} \sum_{N} (T_k - Y_k)^2 \tag{5.11}$$

where, T_k stands for the target (desired) values, Y_k stands for the network produced output values introduced earlier and N stands for the number of examples in the pattern. The changes in the weights are inversely proportional to the derivative of the error with respect to the weights, and can be adjusted using a recursive algorithm starting from the output nodes as,

$$W'_{jk} = W_{jk} + \eta \delta_k H_j \tag{5.12}$$

$$W'_{ij} = W_{ij} + \eta \delta_j X_i \tag{5.13}$$

where,

$$\delta_k = Y_k(1 - Y_k)(T_k - Y_k) \tag{5.14}$$

$$\delta_j = H_j\left(1 - H_j\right) \sum \delta_k W_{jk} \tag{5.15}$$

where, W'_{jk} is the new weight associated with the hidden and output layers, W'_{ij} is the new weight associated with the input and hidden layers, η is a constant, also known as learning rate usually set between 0 and 1, and δ_k and δ_j are the error terms changeable with the selected type of transfer functions.

Defining the variables and the main architecture of the network is the first step of the NNs modelling. The other steps include, developing a strategy to avoid over-training (where the network learns incorrect information/noise, instead of the general pattern), and selecting appropriate training styles, activation functions, learning algorithms and parameter values.

In order to prevent the network from over-training and obtain the best generalization performance, the training data set further separated randomly into two: 80% for training the network, and 20% for cross-validation. As in common practice other network configuration and training issues are proceeded by trial and error selection. All the process is realized in Matlab environment using The Neural Network Toolbox (see Appendix A). Use of this toolbox and required explanations can be seen in Demuth et.al (2009). Table 5.4 indicates the experimented and selected cases for network training and Figure 5.12 shows the convergence of the network with the Levenberg-Marquardt learning algorithm.

The trained network is then used to simulate data sets and to produce unconstrained trip interactions. Finally, a balancing process was applied to predicted flows as in FRBS design in order to satisfy the origin-destination constraints.

Table 5.4. NNTDM Implementation Issues: Experimented and Selected Cases

Implementation Issues	Experimented Cases	Selected Cases
Normalization Technique	• Z-score Normalization • Min-Max Normalization	• Min-Max Normalization
Number of Hidden Layer Neurons	• 3-6-9-12-15 • 20-25-30-40	• 9
Activation Function of Hidden Layer	• Hyperbolic Tangent Function • Logistic Sigmoid Function	• Logistic Sigmoid
Activation Function of Output Layer	• Hyperbolic Tangent Function • Logistic Sigmoid Function • Linear Function	• Logistic Sigmoid
Training Style	• Batch Training • Incremental Training	• Batch Training
Learning Algorithm	• Gradient Descent with Learning Rate • Gradient Descent with Adaptive Learning Rate and Momentum Term • Levenberg-Marquardt	• Levenberg-Marquardt
Performance Measure	• Mean Squared / Absolute Error • r square	• Mean Squared Error • r square
Termination Criteria	• Maximum Epochs • Validation Performance	• Validation Performance

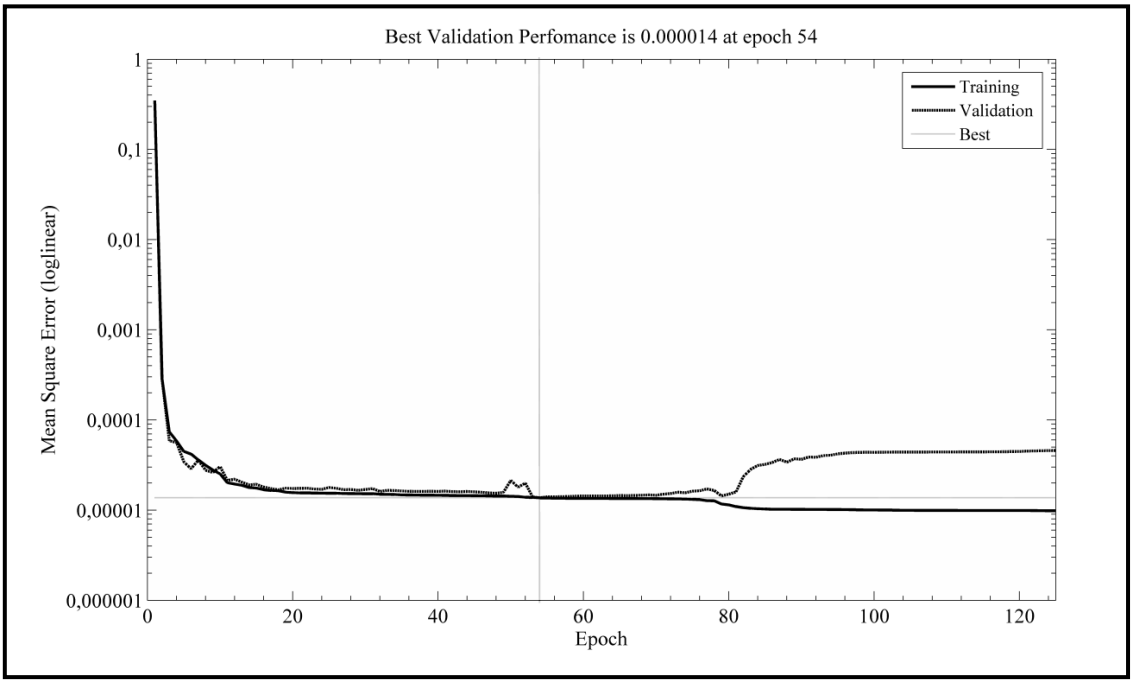

Figure 5.12. NNTDM Back-Propagation Training With Levenberg-Marquardt Learning

5.5. Performance Measures and Goodness-of-Fit Statistics

A spatial interaction model generally tries to identify any system characteristics from observed flows. As being the main prerequisite, a good estimate of observed flows also increases the generalizability of the model and intensifies the theoretical assumptions. Thus, over the years, various goodness-of-fit statistics have been proposed to measure the accuracy of models' estimations.

Commonly, a goodness-of-fit statistic is a quantitative description of some aspect of the difference between observed and estimated values. Especially in spatial interaction and trip distribution modelling, the used statistics can be classified into four main groups as: (i) general distance statistics such as standardized root mean squared error and index of dissimilarity, (ii) information based statistics such as information gain, phi statistic and psi statistic, iii) traditional statistics such as regression statistics (r square, slope and intercept) and Pearson Chi-square, and iv) log-likelihood statistics. (Fotheringham and Knudsen;1987). In addition to these, goodness-of-fit measures of trip distribution modelling can be classified into two levels: micro and macro. The first one is based on the entry by entry comparison of flows with one of the above mentioned statistics. The second one is based on the similarity comparison of certain macro characteristics such as the trip length distributions and mean travel costs of trip matrices. (Smith and Hutchinson;1981).

For the purpose of performance measure and model evaluation in this analysis, several different goodness-of-fit statistics are selected at both micro and macro levels of measure. Among the nine selected statistics having all different properties, five of them measures entry by entry micro level performance: (i) Standardized Root Mean Squared Error (SRMSE), (ii) Coefficient of Determination (r square), (iii) The Regression Slope (Slope), (iv) Average Relative Variance (ARV), (v) The Phi Statistic (Phi), and four of them measures some of macro level performance: (vi) Mean Travel Cost Error (MTCE), (vii) Trip Length Distribution Root Mean Squared Error (TLD RMSE), (viii) Trip Length Distribution Average Relative Absolute Error for the First Five Interval (TLD ARAE F5), (ix) Trip Length Distribution Average Relative Absolute Error for the Last Five Interval (TLD ARAE L5). Each of these statistics is described in turn at following sections. Further reviews can be found in the works by Smith and Hutchinson (1981), Knudsen and Fotheringham (1986) and Fotheringham and Knudsen (1987).

5.4.1. Micro Level Statistics

Description of Mathematical Notations:

T_{ij}^0 is the number of observed trips from zone *i* to zone *j*,

T_{ij} is the number of estimated trips from zone *i* to zone *j*,

$\bar{T}^0$ is the mean of the observed trips

$\bar{T}$ is the mean of the estimated trips

I is the number of origins

J is the number of destinations

$\hat{\sigma}^2$ is the variance of observed trips

$\sum T^0$ is the total number of observed trips

$\sum T$ is the total number of estimated trips

C_{ij} is the travel cost/time between origin *i* and destination *j*

F_i^0 is the observed trip length frequency percentage for the i^{th} *time interval*

F_i is the estimated trip length frequency percentage for the i^{th} *time interval*

n is the number of time intervals associated with trip length frequencies

β is the friction parameter for DCGM

$\bar{C}$ is the observed mean travel cost

$\bar{C}_k$ is the estimated mean travel cost using β_k

Standardized Root Mean Squared Error (SRMSE): SRMSE is one of the most accurate comparative measure of model performances (see Knudsen and Fotheringham, 1986). It is a measure of micro level dissimilarities and it has a lower limit of 0 and upper limit of 1.

$$SRMSE = \sqrt{\frac{\sum_{ij}(T_{ij}^0 - T_{ij})^2}{IxJ}} \Big/ \frac{\sum_{ij}(T_{ij}^0)}{IxJ} \tag{5.16}$$

However, SRMSE values can be greater than this upper limit if the computed average error is greater than the mean. In our case, the latter is valid, thus, a smaller value of SRMSE statistic indicates the more accurate set of estimations.

Coefficient of Determination (r square): Coefficient of determination is the most widely used measure of model performance with well known statistical properties. It is a measure of linear association between observed and estimated values. Its range is between 0 and 1, where the value of 1 indicates a perfect correspondence.

$$r^2 = 1 - \frac{\sum_{ij}(T_{ij}^0 - T_{ij})^2}{\sum_{ij}(T_{ij}^0 - \bar{T}^0)^2} \tag{5.17}$$

Several studies have also showed the error insensitivity of r square statistic in some specific cases (see Smith and Hutchinson, 1981; Knudsen and Fotheringham,1986; Fotheringham and Knudsen; 1987). Therefore, the interpretation of r square measures should be made cautiously.

The Regression Slope (Slope): Another comparison of model fits can be made with the use of linear regression parameters. Plotting the observed values, x_i, against the estimated values, y_i, and to fit a straight line between them with least squares estimate, $y_i = intercept + slope(x_i)$, give useful statistics for performance measure. The intercept should be zero and the slope should be 1 for a perfect fit and the deviation from these values shows worsening fit of the model (Wilson, 1974).

$$Slope = \frac{\sum_{ij}(T_{ij}^0 - \bar{T}^0)(T_{ij} - \bar{T})}{\sum_{ij}(T_{ij}^0 - \bar{T}^0)^2} \tag{5.8}$$

The intercept and the slope statistics are highly correlated. Therefore, the use of slope is found sufficient for performance measure. As emphasized in Fotheringham and Knudsen (1987): if the slope is less than 1, large values tend to be under predicted and small values over predicted, and if the slope is greater than 1, large values tend to be over predicted and small values under predicted.

Average Relative Variance (ARV): ARV is a measure of normalized mean squared error that is used widely in NNs literature (see Fischer and Gopal,1994; Fischer et.al., 2003). It is similar and highly correlated with the SRMSE statistic.

$$ARV = \frac{1}{\hat{\sigma}^2}\,\frac{1}{IxJ}\sum_{ij}(T_{ij}^0 - T_{ij})^2 \tag{5.19}$$

It has a lower limit of 0 and upper limit of 1. However, as in SRMSE values, it can be greater than 1, when the computed average error is greater than the mean. In this situation, a smaller value of ARV statistic indicates more accurate fit.

The Phi Statistic (Phi): The Phi statistic is similar to information gain statistic. The larger the value of Phi statistic the poorer the model fit. Its usage in spatial interaction and trip distribution modelling is reviewed and proposed in the works of Smith and Hutchinson (1981), Knudsen and Fotheringham (1986).

$$Phi = \sum_{ij} \frac{T_{ij}^0}{\sum T^0} \left| \ln \left(\frac{T_{ij}^0}{T_{ij}} \right) \right| \tag{5.20}$$

One of the main handicap of the Phi statistic is its sensitivity to zero entries. If necessary this problem can be solved with substituting a small non-zero elements to these entries.

5.4.2. Macro level Statistics

Mean Travel Cost Error (MTCE): MTCE is a macro level of performance measure in trip distribution modelling. It is additionally in use as a standard calibration procedure for years (see Hyman,1969 for instance).

$$MTCE = \left(\frac{\sum_{ij}(T_{ij}^0 C_{ij})}{\sum T^0} \right) - \left(\frac{\sum_{ij}(T_{ij} C_{ij})}{\sum T} \right) \tag{5.21}$$

If the deviation (negative or positive) from observed mean travel cost is larger, so the error in estimated friction factors are larger.

Trip Length Distribution Root Mean Squared Error (TLD RMSE): TLD based measure is also a macro level of measure for trip distribution modelling., and in use for years both in calibration (see Dickey,1983; TRB,1998 for some different applications) and the model evaluation phases. Simply a TLDE RMSE measures the root mean squared

differences between observed and estimated trip length frequency distributions associated with usually time intervals.

$$TLD\ RMSE = \sqrt{\frac{\sum_{i=1}^{n}(F_i^0 - F_i)^2}{n}} \tag{5.22}$$

The larger value of TLDE RMSE indicates higher error between observed and estimated TLDs.

Trip Length Distribution Average Relative Absolute Error for the First Five Interval (TLD ARAE F5): Average Relative Absolute Error (ARAE) is a useful statistic when it is preferred to measure the proportion of errors rather than magnitudes. It can also be used in entry by entry measures (see Roy, 2004), however, in our case, it is a measure of proportional errors between the observed and estimated trip length frequencies for the first five time intervals.

$$TLD\ ARAE\ F5 = \frac{1}{n}\sum_{i=1}^{n}\left(\left|F_i^0 - F_i\right|/F_i^0\right) \tag{5.23}$$

Its usage could be effective where especially searching out the relative errors related with short trips. When having a perfect fit of the observed and the estimated first five trip length frequencies, the TLD ARAE F5 is zero. But considering its upper level, its value has no restriction.

Trip Length Distribution Average Relative Absolute Error for the Last Five Interval (TLD ARAE L5): TLD ARAE L5 is a similar performance measure with the previously described TLD ARAE F5. The only difference is, it searches the relative errors in estimating trips with long travel times.

$$TLD\ ARAE\ L5 = \frac{1}{n-(n-5)}\sum_{i=(n-5)}^{n}\left(\left|F_i^0 - F_i\right|/F_i^0\right) \tag{5.24}$$

It measures proportional errors between the observed and the estimated trip length frequencies for the last five time intervals and it has a negative correlation with TLD RMSE and TLD ARAE F5.

CHAPTER 6

RESULTS

This chapter presents goodness-of-fit statistics of the previously introduced trip distribution models: a Doubly-Constrained Gravity Model with Maximum Likelihood (DCGM ML) estimation, a Doubly-Constrained Gravity Model with Weighted Least Squares (DCGM WLS) estimation, a Doubly-Constrained Gravity Model with Trip Length Distribution (DCGM TLD) based estimation, a Neural Network based Trip Distribution Model (NNTDM), a Fuzzy Rule-Based System (FRBS) design and a Genetic Fuzzy Rule-Based System (FRBS) design.

All the models have simulated for each of the training, testing and whole data sets. Their performances in each case have measured with various goodness-of fit statistics. Five of the statistics measure entry by entry, micro level performance: i) Standardized Root Mean Squared Error (SRMSE), ii) Coefficient of Determination (r square), iii) The Regression Slope (Slope), iv) Average Relative Variance (ARV), v) The Phi Statistic (Phi), and four of them measure some of macro level performance: vi) Mean Travel Cost Error (MTCE), vii) Trip Length Distribution Root Mean Squared Error (TLD RMSE), viii) Trip Length Distribution Average Relative Absolute Error for the First Five Interval (TLD ARAE F5), ix) Trip Length Distribution Average Relative Absolute Error for the Last Five Interval (TLD ARAE L5). Finally, the performance of the models further measured with respect to the results of district-based aggregation and trip shares such as among intra-zonal vs. inter-zonal, intra-district vs. inter-district trips.

6.1. Training Results

The training results reported in Table 6.1. show the learning capacity of the implemented trip distribution models. As an overall evaluation, it can be stated that the GFRBS design demonstrates a certain superiority for almost all goodness-of-fit statistics. It has showed a considerably good performance especially with having 3.42 SRMSE, 0.90 r square and 0.07 TLD RMSE scores. The DCGMs and FRBS design follow the GFRBS design with respect to their modelling performances. Their statistics

are close to each other and they all have produced fair predictions having a 0.85 or more r square scores. Especially, the DCGM ML estimation is good at both micro and macro level statistics, whereas the DCGM WLS estimation is good at micro level, and the DCGM TLD based estimation is good at macro level statistics. The FRBS design achieved better results when the micro level statistics are taken into account. It has the second best scores of SRMSE and r square with 4.10 and 0.87, but the second worst scores with the MTCE and TLD RMSE with -4.29 and 0.11. Finally, the NNTDM has obtained unexpected results in the training phase especially with the lowest SRMSE and r square scores. The reason for that can be one of its implementation procedures that the NNTDM was subjected to a row-column balancing, and it was stopped to train before a possible over-training problem.

Apart from these, as it is indicated with the graphical analysis of the TLDs (Figures 6.1 - 6.6), all the models have captured the general trend of the observed TLD, except for the DCGM WLS estimation. Especially, GFRBS design and DCGM TLD based estimation have caught nearly a perfect fit to the observed TLD. Additionally, when the regression plots of the models (Figures 6.7 - 6.12) are analyzed, it can be said that: all the models have showed a little tendency to under predict larger flows and over predict smaller flows. It can be explained with general structure of the observed flow matrix that is mostly involved small amount of flows. The GFRBS design and the DCGM WLS estimation is still successful in this respect. They have produced considerably good predictions with a slope score that is close to unity.

Table 6.1. Model Results: Goodness-of-Fit Statistics for Training Data Set

Trip Distribution Models	**SRMSE**	**r square**	**Slope**	**ARV**	**Phi Statistic**	**MTCE**	**TLD RMSE**	**ARAE TLD F5**	**ARAE TLD L5**
DCGM ML Est.	4.17	0.86	0.86	0.17	0.94	-2.77	0.10	0.10	9.46
DCGM WLS Est.	4.16	0.86	**0.92**	0.17	0.96	-1.65	0.18	0.14	8.92
DCGM TLD Based Est.	4.29	0.85	0.80	0.18	0.93	-4.18	**0.07**	0.07	10.56
NNTDM	4.75	0.82	0.74	0.22	1.00	-4.14	0.11	0.10	**7.5**
FRBS Design	4.10	0.87	0.79	0.16	1.00	-4.29	0.11	0.10	11.03
GFRBS Design	**3.42**	**0.90**	**0.92**	**0.11**	**0.92**	**-1.07**	**0.07**	**0.06**	10.58

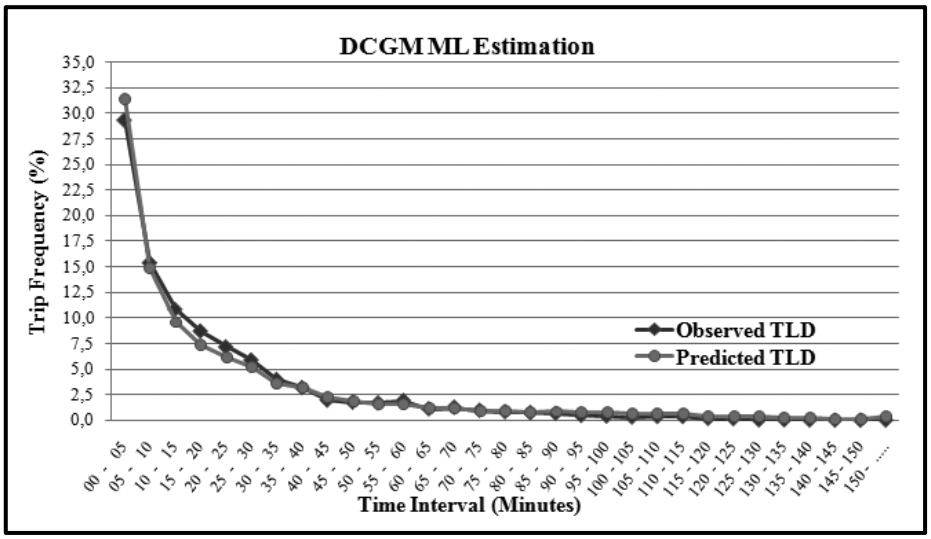

Figure 6.1. TLD Comparison - DCGM ML Estimation on Training Data Set

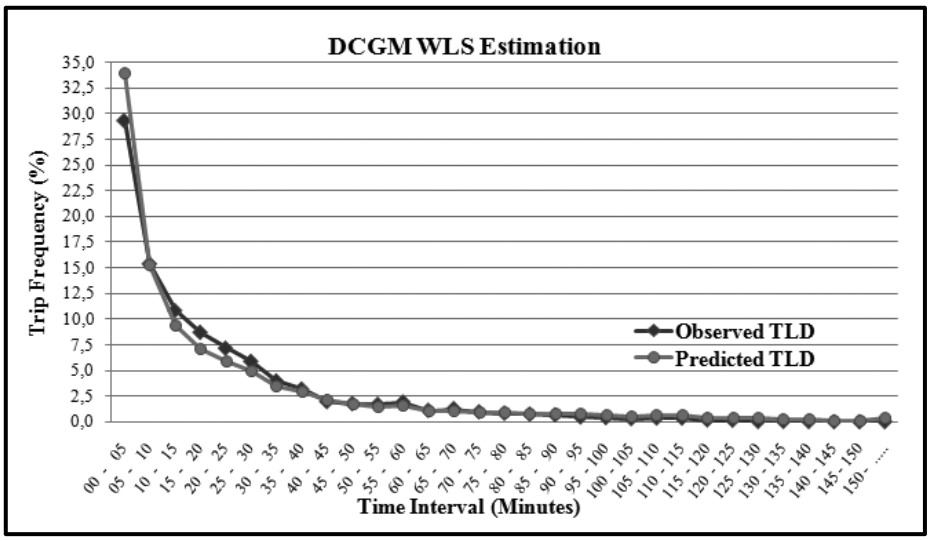

Figure 6.2. TLD Comparison - DCGM WLS Estimation on Training Data Set

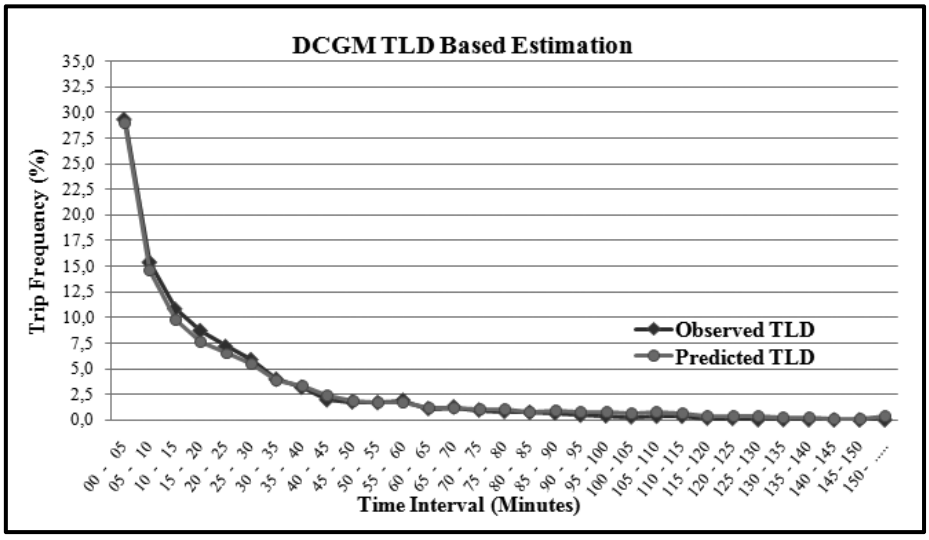

Figure 6.3. TLD Comparison - DCGM TLD Based Estimation on Training Data Set

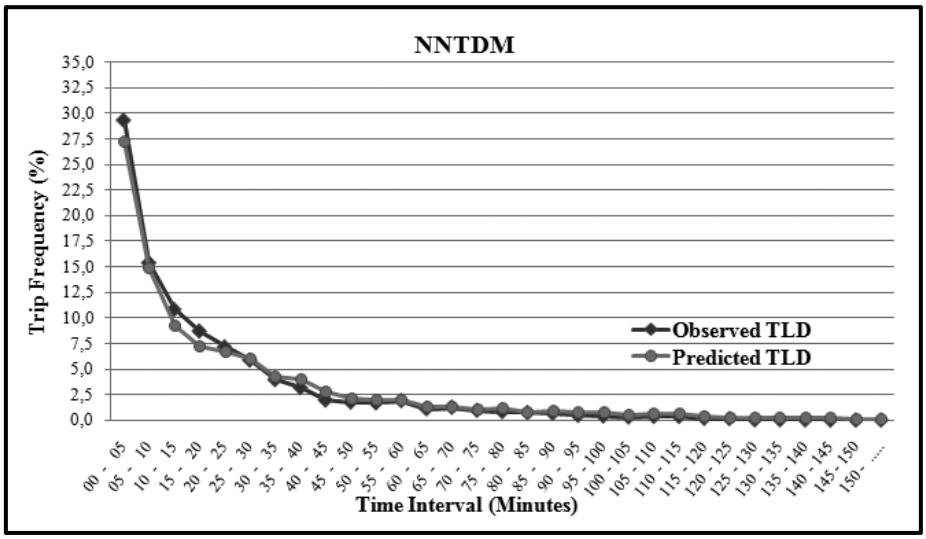

Figure 6.4. TLD Comparison - NNTDM on Training Data Set

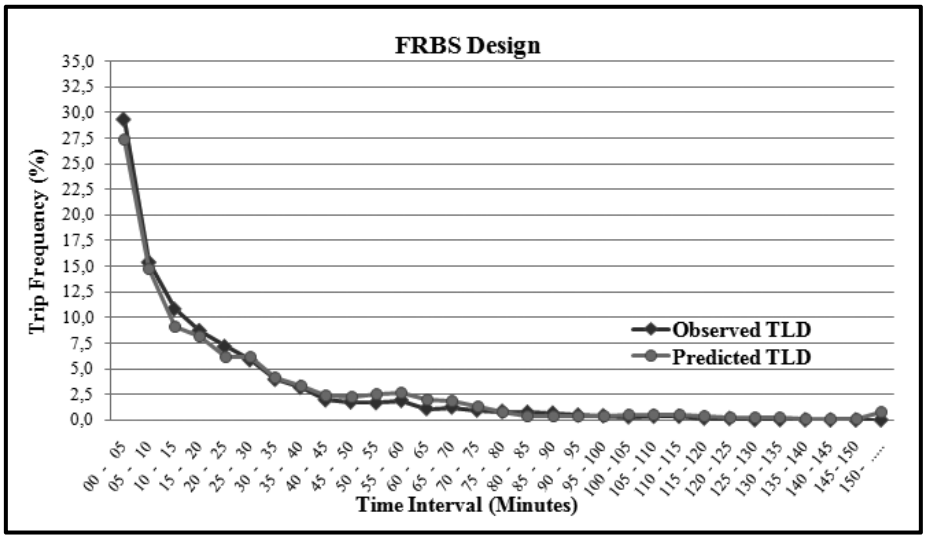

Figure 6.5. TLD Comparison - FRBS Design on Training Data Set

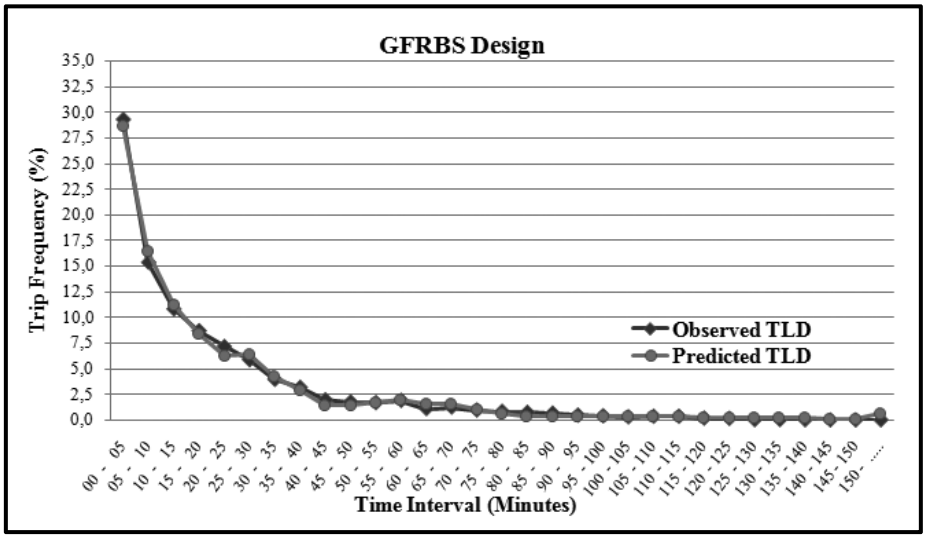

Figure 6.6. TLD Comparison - GFRBS Design on Training Data Set

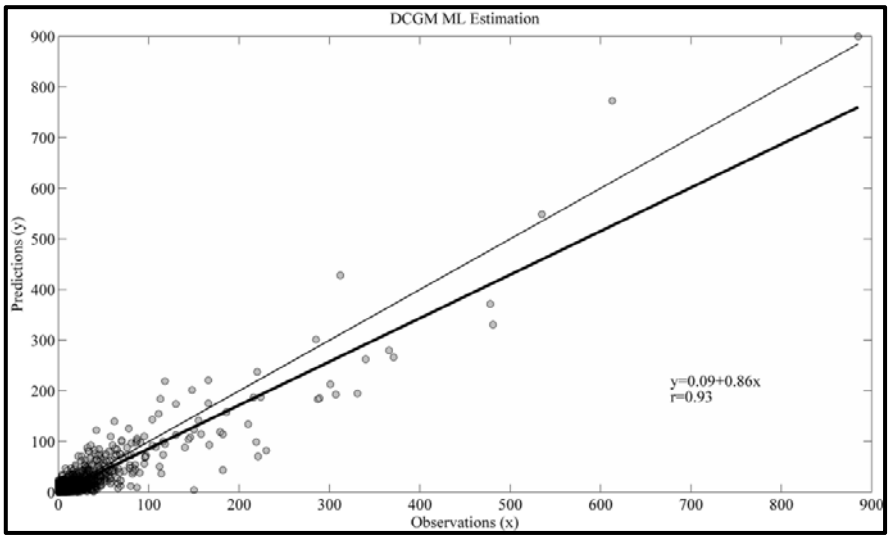

Figure 6.7. Regression Plots - DCGM ML Estimation on Training Data Set

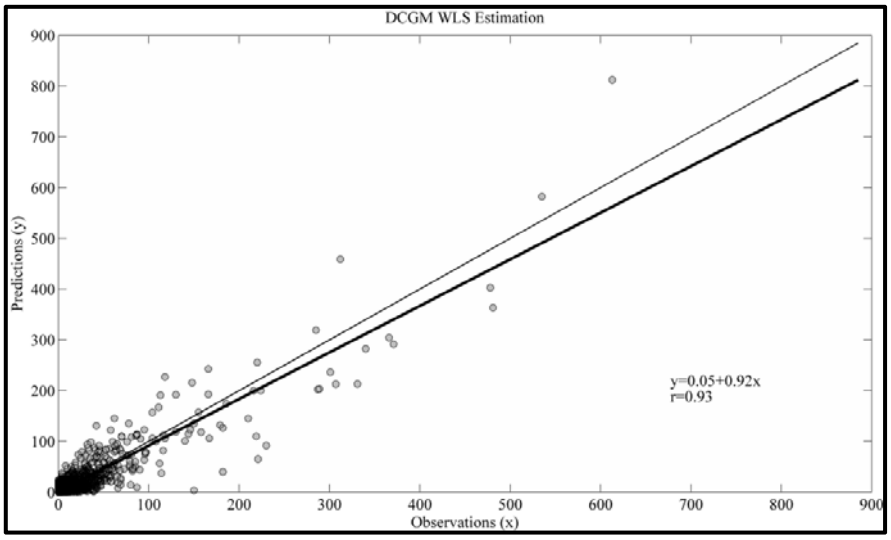

Figure 6.8. Regression Plots - DCGM WLS Estimation on Training Data Set

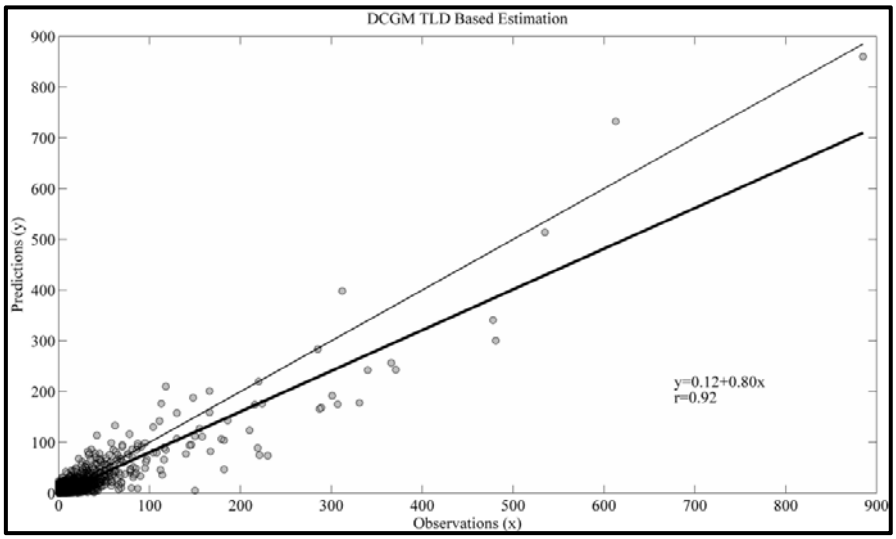

Figure 6.9. Regression Plots - DCGM TLD Based Estimation on Training Data Set

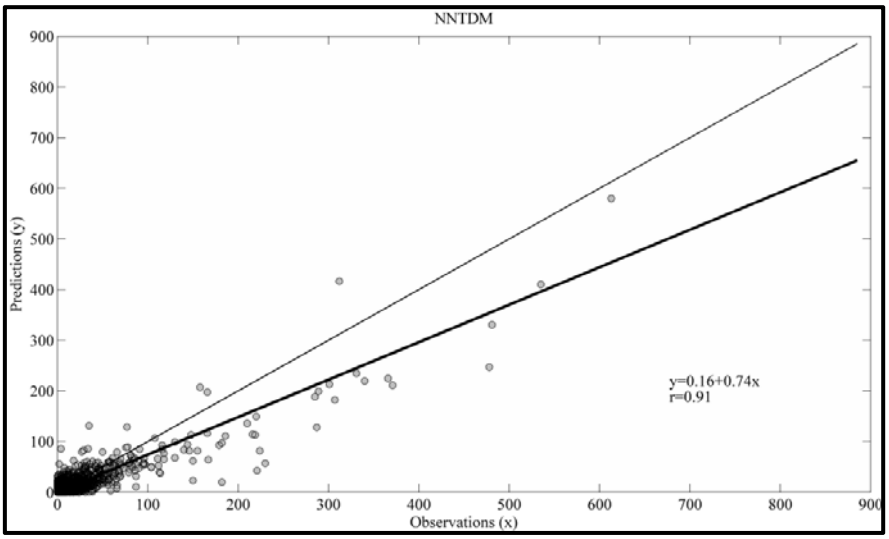

Figure 6.10. Regression Plots - NNTDM on Training Data Set

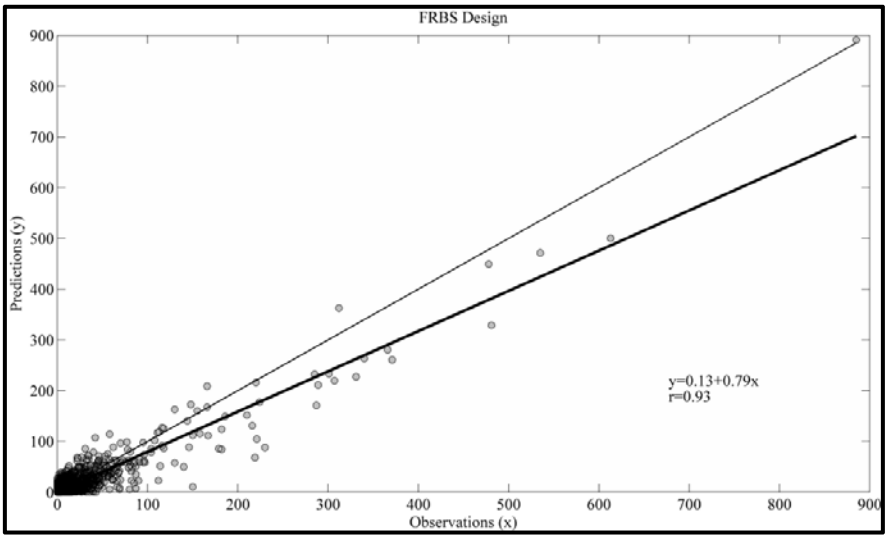

Figure 6.11. Regression Plots - FRBS Design on Training Data Set

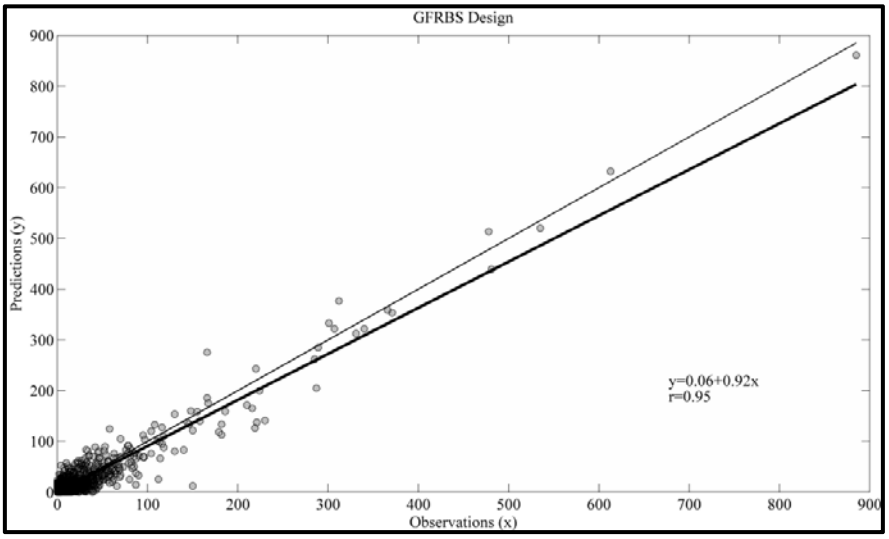

Figure 6.12. Regression Plots - GFRBS Design on Training Data Set

6.2. Testing Results

The testing results of the models are more important than the training results. They represent predictive ability of the models better, and give an idea about trained models generalizability performance. According to the testing scores (Table 6.2), the GFRBS design outperforms all other models in almost every statistics except for the regression slope and phi statistic. In comparison to the training results, the GFRBS design have recorded a small decrease in SRMSE (3.91) and r square (0.81) scores, whereas an increase in MTCE (-0.85) and TLD RMSE (0.05) scores. This is a desired and expected situation that the GFRBS design have successfully learnt macro behaviour of the analyzed system. A similar observation is valid for the FRBS design. On the contrary to training, FRBS design have showed a better performance than DCGMs in testing case. FRBS design has achieved the second best performance with respect to the three important statistics SRMSE (4.15), r square (0.78) and TLD RMSE (0.10) scores. Apart from these, it can be said that the DCGM ML and DCGM TLD based estimations have preserved their performance and produced fairly good predictions. However, the DCGM WLS estimation has got worse especially when its TLD RMSE score is taken into account. The NNTDM model catch up with the other models a little. Its SRMSE score in testing has decreased to 4.42 from 4.75 and its TLD RMSE score has remained nearly the same. This is meaningful that the over-training strategy has worked well with NNTDM. The visual analysis of the TLD comparisons and the regression plots of the testing case follow approximately the same pattern of the training estimations. The following figures from Figure 6.13 to Figure 6.24 presents visual comparison of results.

Table 6.2. Model Results: Goodness-of-Fit Statistics for Testing Data Set

Trip Distribution Models	SRMSE	r square	Slope	ARV	Phi Statistic	MTCE	TLD RMSE	ARAE TLD F5	ARAE TLD L5
DCGM ML Est.	4.25	0.78	0.86	0.19	0.94	-2.89	0.15	0.12	11.51
DCGM WLS Est.	4.34	0.78	**0.92**	0.20	0.97	-1.71	0.22	0.15	10.90
DCGM TLD Based Est.	4.27	0.77	0.79	0.19	**0.93**	-4.39	0.11	0.09	12.74
NNTDM	4.42	0.75	0.69	0.20	1.00	-4.19	0.12	0.11	**4.25**
FRBS Design	4.15	0.78	0.75	0.18	1.02	-4.25	0.10	0.08	18.02
GFRBS Design	**3.91**	**0.81**	0.86	**0.16**	0.97	**-0.85**	**0.05**	**0.02**	11.03

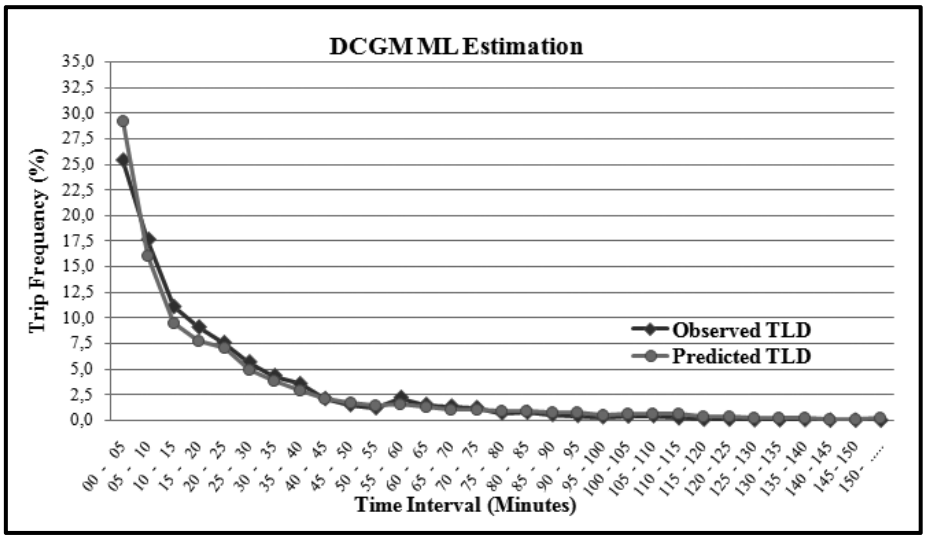

Figure 6.13. TLD Comparison - DCGM ML Estimation on Testing Data Set

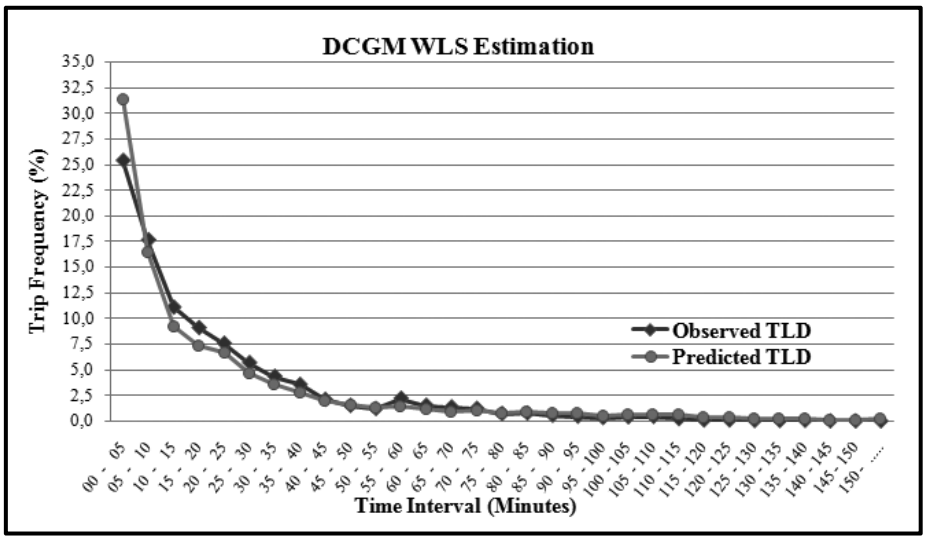

Figure 6.14. TLD Comparison - DCGM WLS Estimation on Testing Data Set

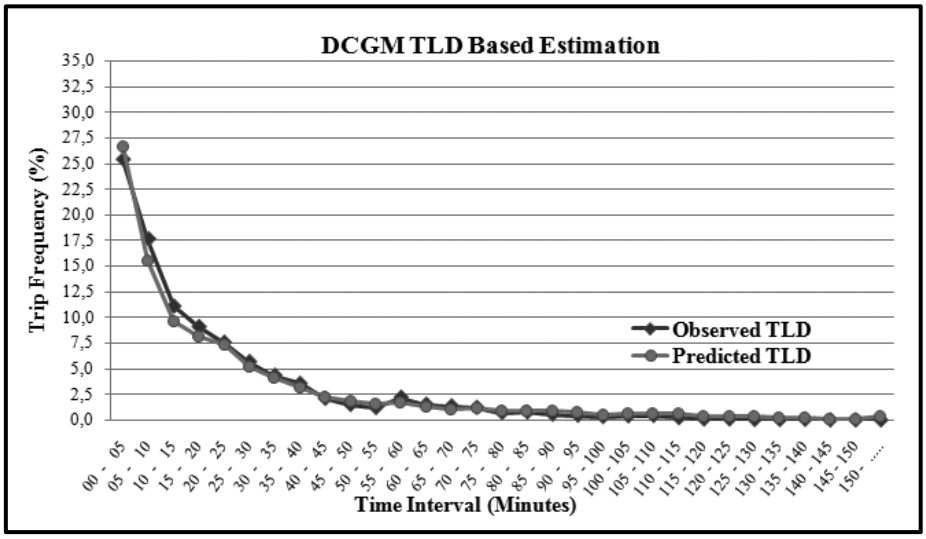

Figure 6.15. TLD Comparison - DCGM TLD Based Estimation on Testing Data Set

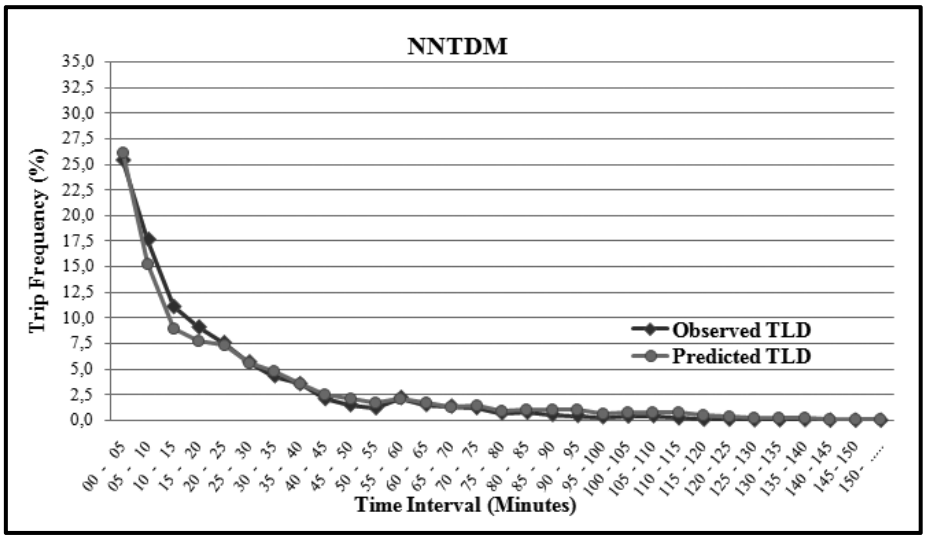

Figure 6.16. TLD Comparison - NNTDM on Testing Data Set

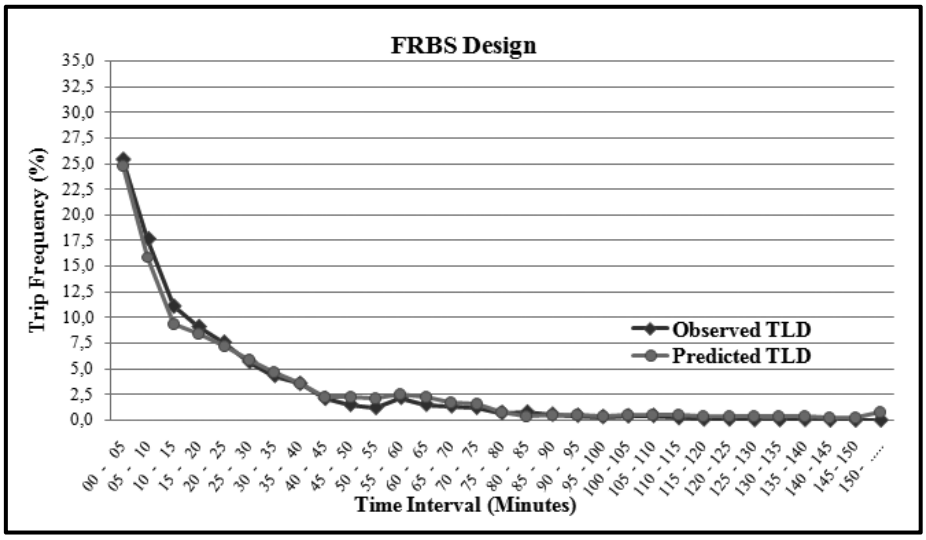

Figure 6.17. TLD Comparison - FRBS Design on Testing Data Set

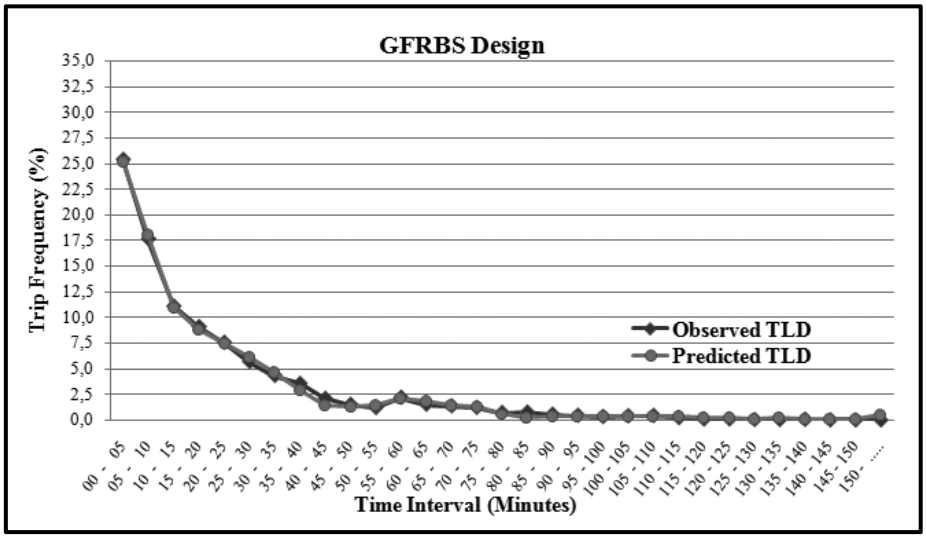

Figure 6.18. TLD Comparison - GFRBS Design on Testing Data Set

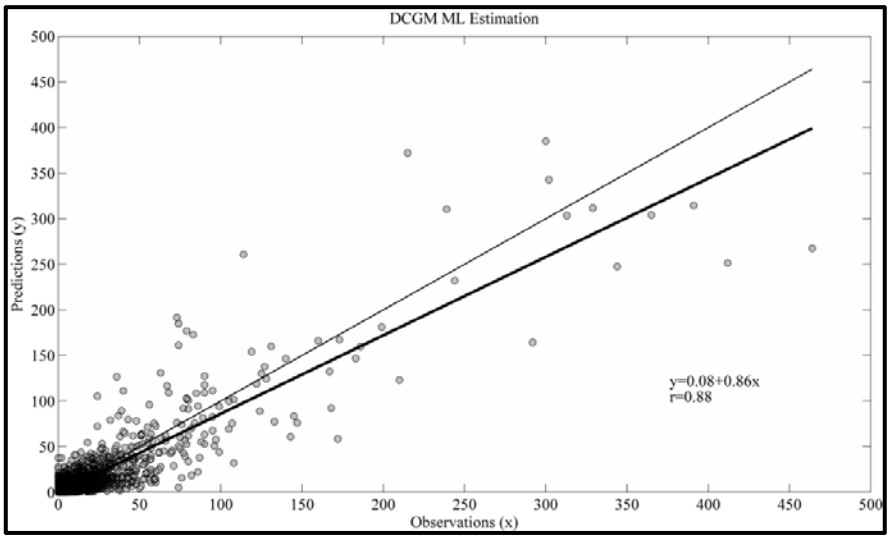

Figure 6.19. Regression Plots - DCGM ML Estimation on Testing Data Set

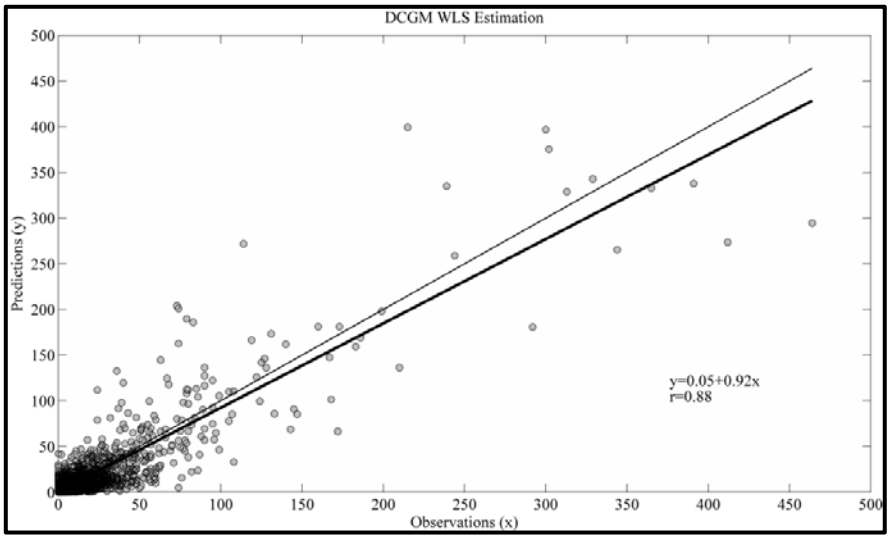

Figure 6.20. Regression Plots - DCGM WLS Estimation on Testing Data Set

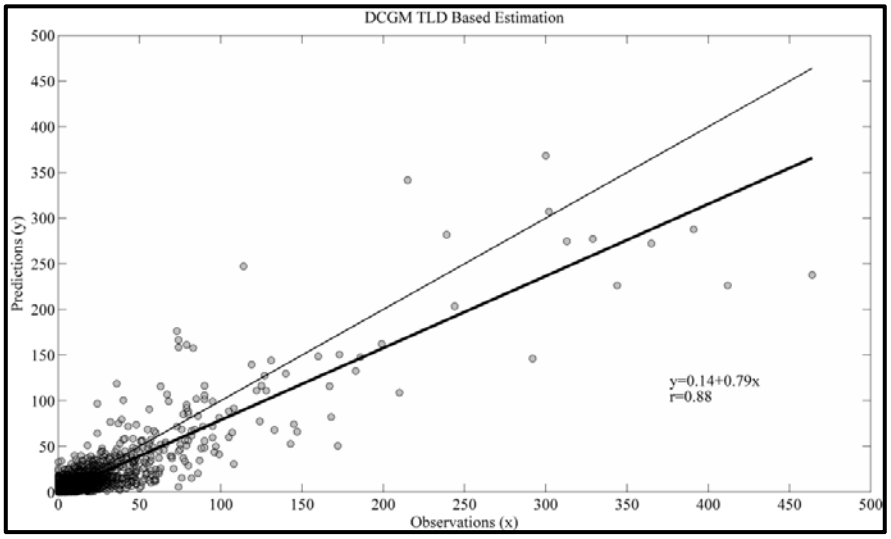

Figure 6.21. Regression Plots - DCGM TLD Based Estimation on Testing Data Set

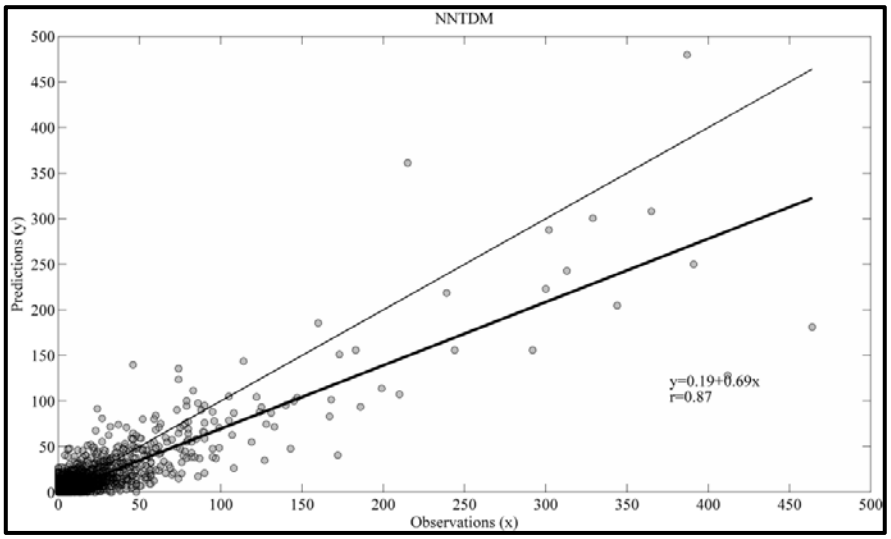

Figure 6.22. Regression Plots - NNTDM on Testing Data Set

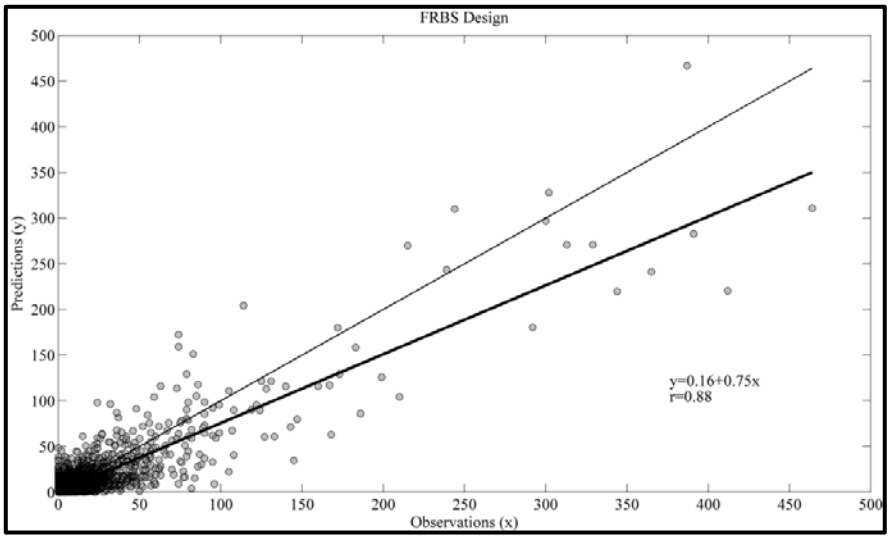

Figure 6.23. Regression Plots - FRBS Design on Testing Data Set

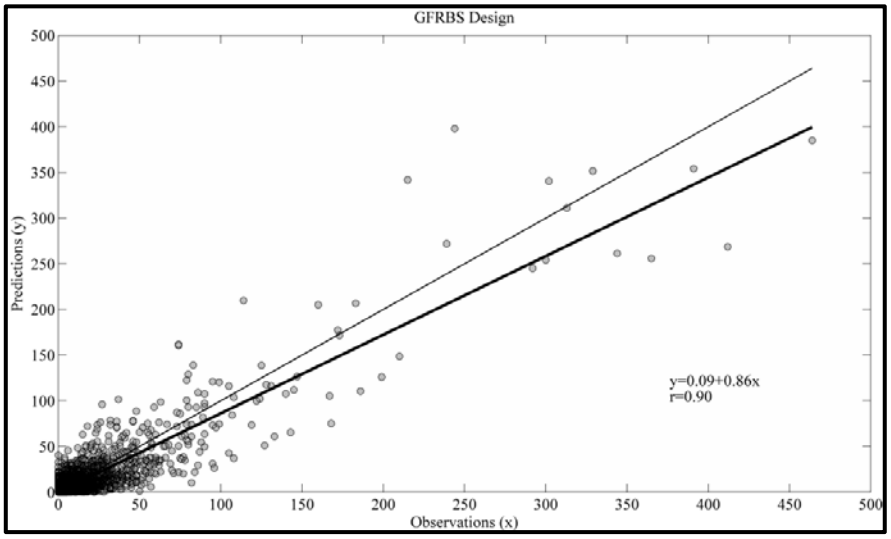

Figure 6.24. Regression Plots - GFRBS Design on Testing Data Set

6.3. Overall Results

The GFRBS design has achieved the best scores in almost all goodness-of-fit statistics, when the models are simulated with the whole data set. It has gained a 4.30 SRMSE, 0.82 r square and 0.08 TLD RMSE score that is consistent with the training and the testing cases. FRBS design has showed the second best performance in whole data simulation, especially with the 4.56 SRMSE, 0.79 r square and 0.09 TLD RMSE scores. DCGM ML and DCGM TLD based estimation has followed the FRBS design with a nearby performance and fairly good predictions. DCGM WLS estimation has showed a worsening performance in whole data simulation. On the contrary to the training and testing, its performance is poorer than the NNTDM with the worst TLD RMSE (0.26) and SRMSE (5.10) scores. Table 6.3 indicates the goodness-of fit statistics for the whole data set model simulations.

As it is indicated from the visual analysis of the TLDs (Figures 6.25 - 6.30), all the models have captured the general trend of the observed TLD, except for the DCGM WLS and DCGM ML estimations. Especially, the GFRBS and the FRBS designs have caught an outstanding fit to the observed TLD. Similar with the training case, all the models have showed a little tendency to under predict larger flows and over predict smaller flows as it can be seen in Figures 6.31-6.36. It can be explained with the structure of the observed trip matrix that is dominated with the small amount of flows. The DCGM WLS estimation and the GFRBS design are the most successful in this respect with 0.94 and 0.87 slope scores.

Table 6.3. Model Results: Goodness-of-Fit Statistics for the Whole Data Set

Trip Distribution Models	SRMSE	r square	Slope	ARV	Phi Statistic	MTCE	TLD RMSE	ARAE TLD F5	ARAE TLD L5
DCGM ML Est.	4.88	0.77	0.87	0.23	0.97	-2.68	0.17	0.13	10.40
DCGM WLS Est.	5.10	0.77	**0.94**	0.26	1.00	-1.44	0.26	0.18	9.80
DCGM TLD Based Est.	4.82	0.77	0.79	0.23	**0.96**	-4.33	0.11	0.10	11.76
NNTDM	5.02	0.75	0.71	0.25	1.07	-4.30	0.13	0.14	**5.91**
FRBS Design	4.56	0.79	0.75	0.21	1.02	-3.19	0.09	**0.07**	13.65
GFRBS Design	**4.30**	**0.82**	0.88	**0.19**	**0.96**	**-0.33**	**0.08**	**0.07**	10.59

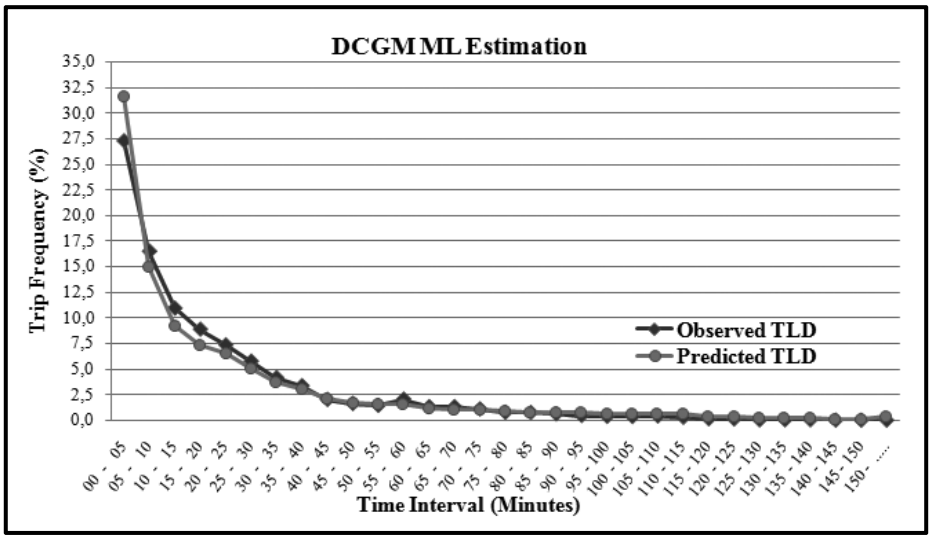

Figure 6.25. TLD Comparison - DCGM ML Estimation on Whole Data Set

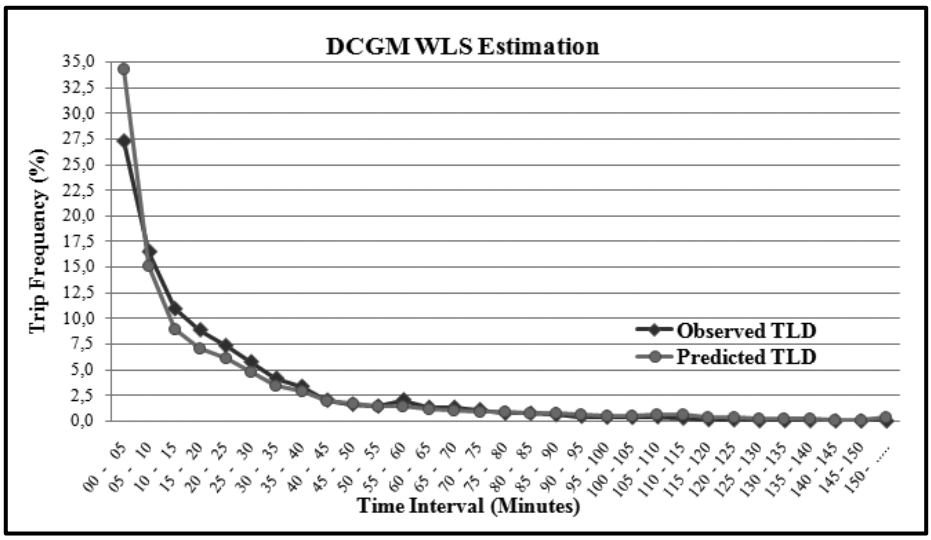

Figure 6.26. TLD Comparison - DCGM WLS Estimation on Whole Data Set

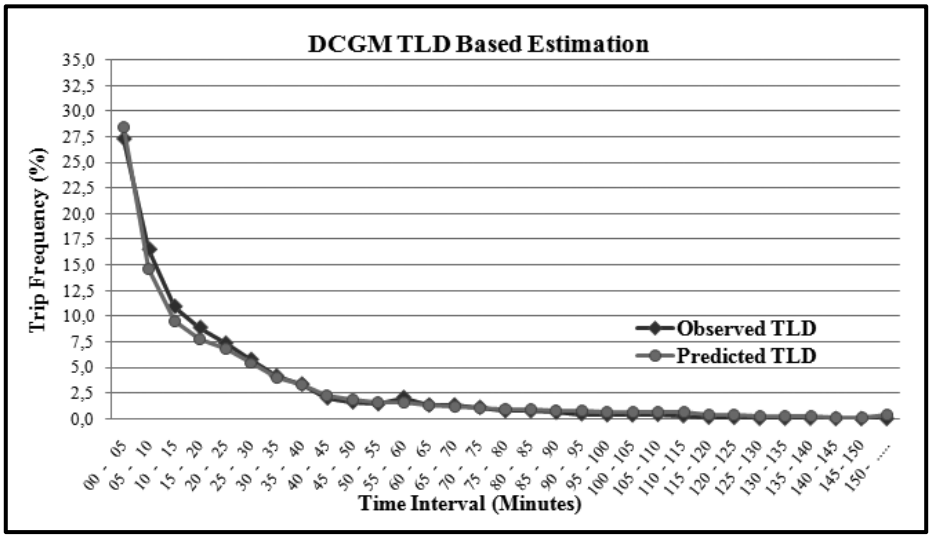

Figure 6.27. TLD Comparison - DCGM TLD Based Estimation on Whole Data Set

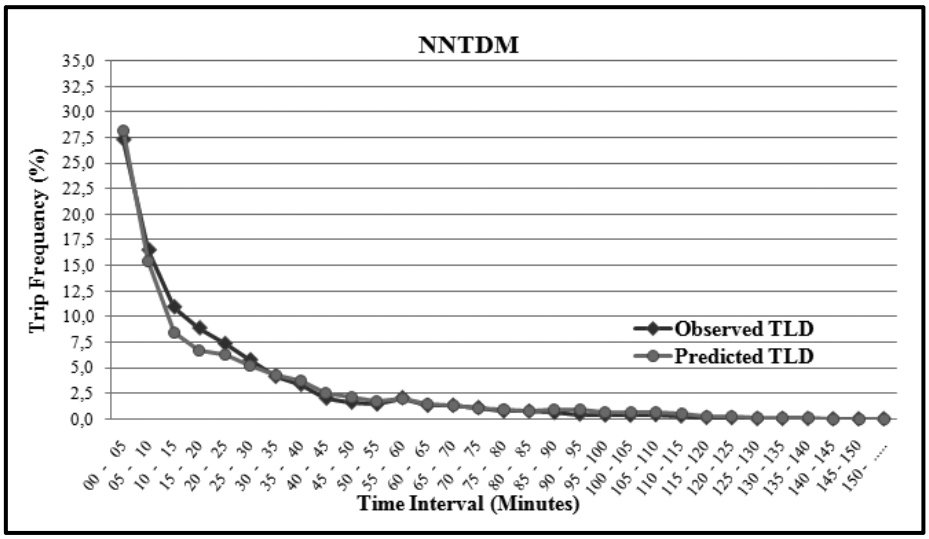

Figure 6.28. TLD Comparison - NNTDM on Whole Data Set

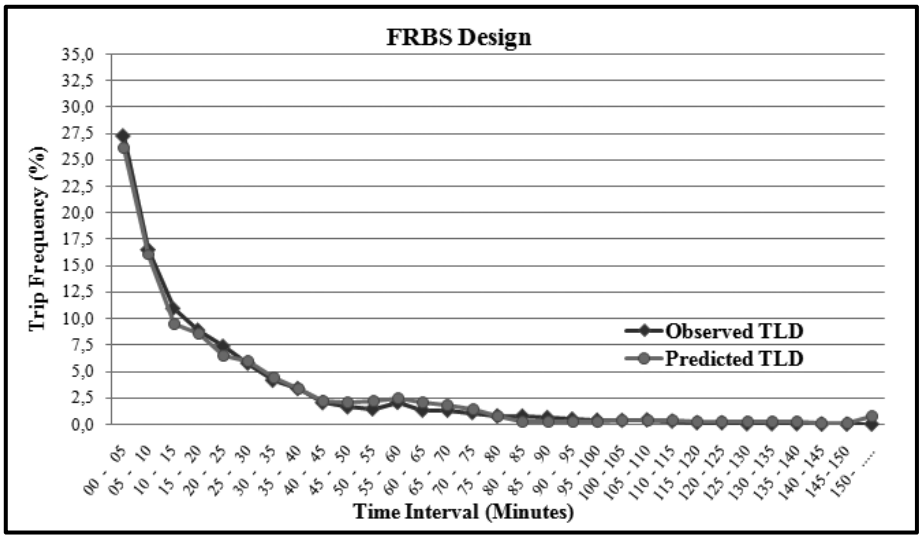

Figure 6.29. TLD Comparison - FRBS Design on Whole Data Set

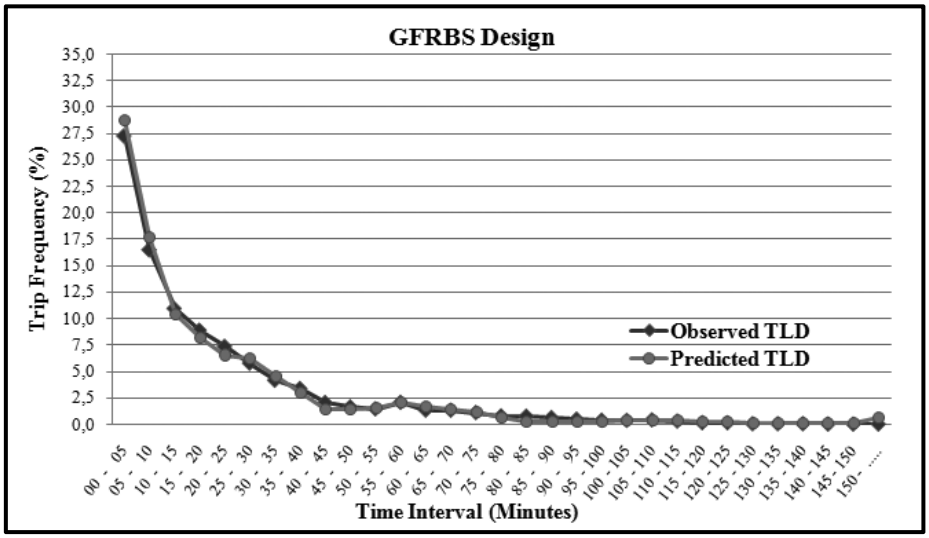

Figure 6.30. TLD Comparison - GFRBS Design on Whole Data Set

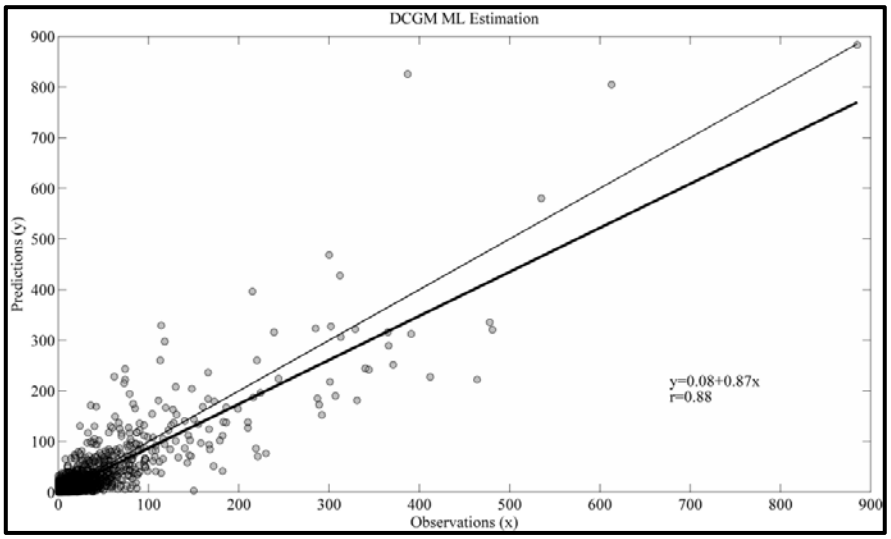

Figure 6.31. Regression Plots - DCGM ML Estimation on Whole Data Set

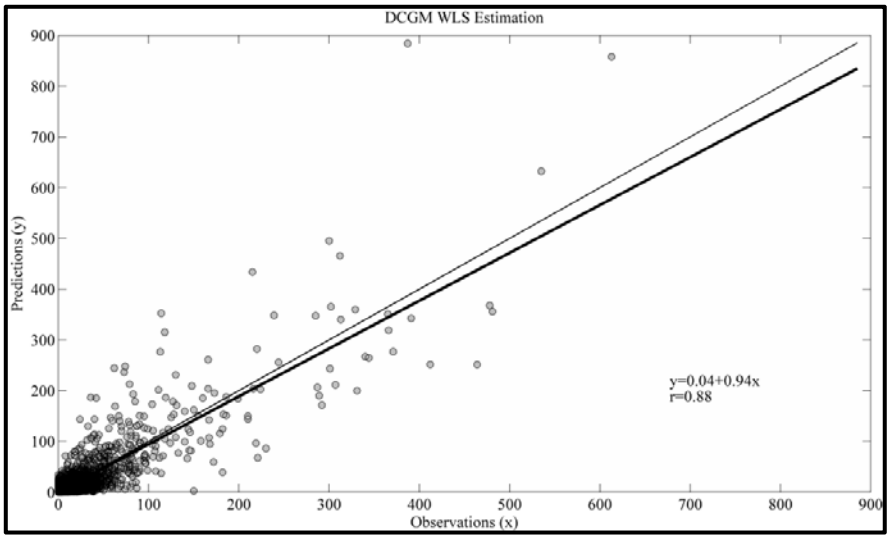

Figure 6.32. Regression Plots - DCGM ML Estimation on Whole Data Set

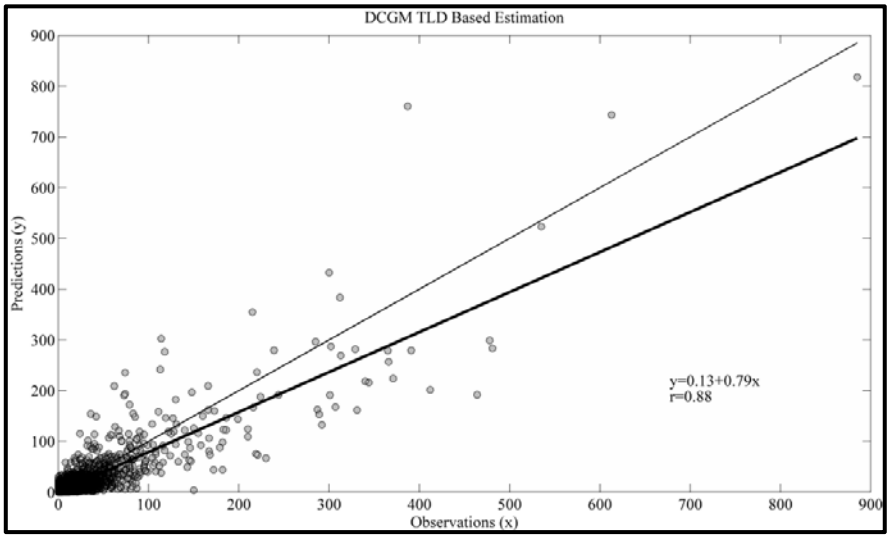

Figure 6.33. Regression Plots - DCGM TLD Based Estimation on Whole Data Set

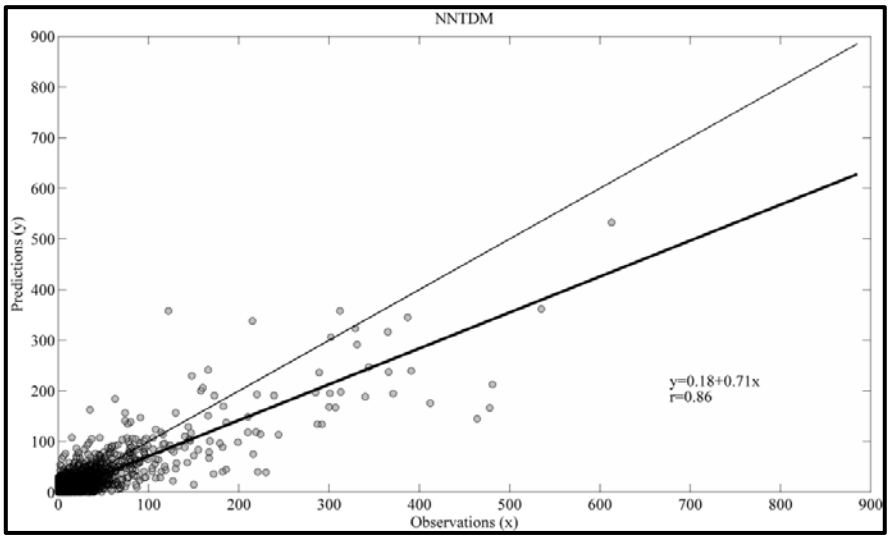

Figure 6.34. Regression Plots - NNTDM on Whole Data Set

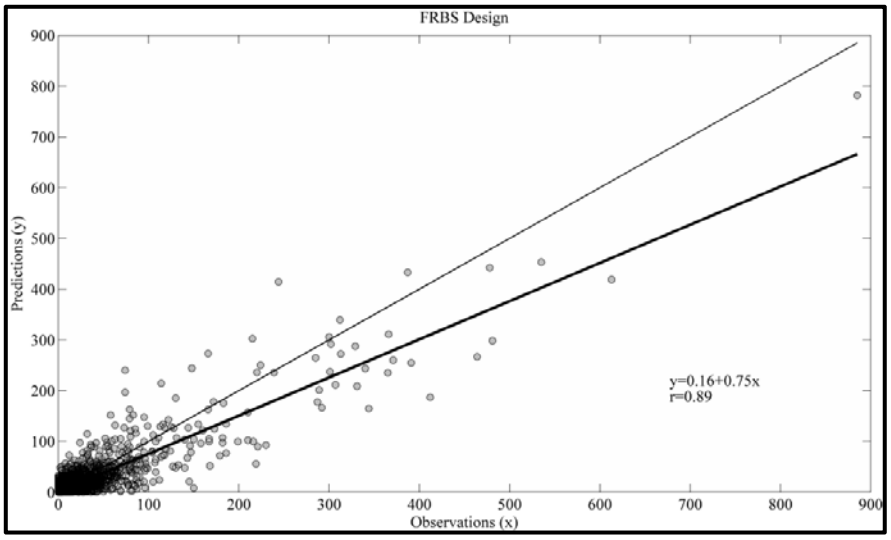

Figure 6.35. Regression Plots - FRBS Design on Whole Data Set

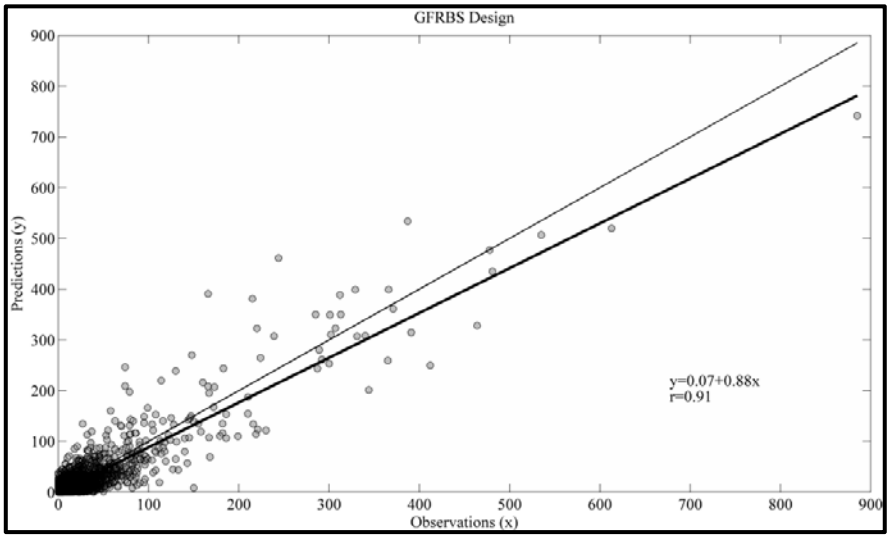

Figure 6.36. Regression Plots - GFRBS Design on Whole Data Set

As it can be observed in the majority of previous cases, the proposed GFRBS design have produced superior predictions. The FRBS design, the DCGM TLD based estimation and the DCGM ML estimation in turn have followed the GFRBS design. They have also achieved high level of accuracy and good level of generalizability after the GFRBS design. The DCGM WLS estimation and the NNTDM have not showed the expected performance. Especially, the performance of the DCGM WLS estimation has decreased within the testing and the whole data set simulations.

The performance of the models is further tested with a district-based aggregation of trip interchanges. All of the model results for traffic analysis zones were aggregated within the corresponding 31 districts in Istanbul Metropolitan Area. Then observed and predicted flows for the districts compared using three micro level goodness-of-statistics as shown in Table 6.4. The GFRBS design has outperformed the other models as in previous cases. In general all the models have achieved good results with having at least 0.92 r square score in district-based measure.

Table 6.4. Model Results: District-Based Goodness-of-Fit Statistics

Goodness-of-Fit Statistics	**DCGM ML Estimation**	**DCGM WLS Estimation**	**DCGM TLD Based Estimation**	**NNTDM**	**FRBS Design**	**GFRBS Design**
SRMSE	0.78	0.72	0.88	1.04	0.86	**0.63**
r square	0.95	0.95	0.94	0.92	0.95	**0.97**
Slope	1.08	**1.05**	1.15	1.18	1.14	**1.05**

Finally, the spatial distribution of modelled trip interchanges have further measured with a trip share comparison. Observed and modelled trip shares among intra-zonal vs. inter-zonal, intra-district vs. inter-district and bridge crossing vs. not bridge crossing trips has measured. The results of this measure is indicated in Table 6.5. Nearly all the models have successfully estimated trip shares approximately with 10-15% error in most of the cases. Surprisingly, the DCGM WLS estimation has outperformed the other models. We had mandatorily seeded the trip matrix with a very small number before calibrating DCGM WLS estimation. Spatial distribution of the modelled trip interchanges could be affected from this implementation leading to a better performance of the model with respect to trip shares.

Table 6.5. Observed and modelled trip shares: intra-zonal vs. inter-zonal, intra-district vs. inter-district and bridge crossing vs. not bridge crossing trips

Trips	**OBSERVED**	**DCGM ML Est.**	**DCGM WLS Est.**	**DCGM TLD Based Est.**	**NNTDM**	**FRBS Design**	**GFRBS Design**
Intra-Zonal	24.5%	24.3%	27.0%	21.4%	18.6%	18.5%	21.7%
Percentage Error	-----	**-1.1%**	+9.8%	-12.9%	-24.1%	-24.4%	-11.8%
Inter-Zonal	75.5%	75.7%	73.0%	78.6%	81.4%	81.5%	78.3%
Percentage Error	-----	**+0.3%**	-3.2%	+4.2%	+7.8%	+7.9%	+3.8%
Intra-District	47.2%	42.7%	45.7%	39.4%	38.8%	38.5%	42.7%
Percentage Error	-----	-9.44	**-3.20**	-16.55	-17.78	-18.47	-9.39
Inter-District	52.8%	57.3%	54.3%	60.6%	61.2%	61.5%	57.3%
Percentage Error	-----	8.43	**2.86**	14.78	15.88	16.49	8.39
Bridge Crossing	6.8%	8.2%	7.4%	9.2%	10.5%	7.7%	5.7%
Percentage Error	-----	+20.4%	**+8.6%**	+34.4%	+53.1%	+12.1%	-15.8%
Not Bridge Crossing	93.2%	91.8%	92.6%	90.8%	89.5%	92.3%	94.3%
Percentage Error	-----	-1.5%	**-0.6%**	-2.5%	-3.8%	-0.9%	+1.2%

CHAPTER 7

CONCLUSION

The general purpose of this study was to set out a fuzzy and a genetic fuzzy system to estimate intra-city passenger flows, thus, to contribute to the literature representing their potential use in trip distribution modelling. For this purpose, a simple Mamdani-type FRBS was developed to estimate trip interchanges in Istanbul Metropolitan Area. Its rule base and fuzzy partitions were constructed with a mixed procedure including both learning from examples and expertise. Aggregate variables of the traditional trip distribution problem (production, attraction and friction) were used as inputs. Afterwards, the rule base of the proposed FRBS was further improved with a novel GFRBS design. The rule base learning process in GFRBS design was shifted to a combinatorial optimization problem and solved with a probabilistic and adaptive genetic algorithm. Both of the two model outputs was enforced to satisfy production and attraction constraints giving access to use of them as a part of sequential travel demand modelling.

The performance of the proposed models was evaluated comparatively with respect to the benchmark models: a traditional doubly-constrained gravity model and a multilayer feed-forward neural network. Various goodness-of-fit statistics were used in the performance evaluation. According to the results achieved, a straightforward consequence is that the FRBSs and the GFRBSs can be used in predicting intra-city passenger flows with high level of accuracy.

The present study has also examined the proposed and the benchmark models in many respects and achieved a plenty of empirical results. Additionally, all the models have evaluated according to their simplicity, predictive ability, interpretability, flexibility, data dependency etc. in trip distribution modelling. The following statements below, briefly demonstrate findings of the empirical analysis, and Table 7.1 summarizes evaluation of the main characteristics of the models for a doubly-constrained case:

i. traditional doubly-constrained gravity models are still simple and efficient; they are steady and strong, yet they suffer from one-parameter generalization; they

should be tried first in almost every case, at least for a comparison; they offer better predictions with maximum likelihood or TLD based parameter estimation.

ii. neural networks may not show expected performance if they are forced to satisfy production-attraction constraints; they are unsteady, non-interpretable and case/data dependent; its equation-free structure and potential usage with additional inputs provide an outstanding advantage.

iii. simply-designed FRBSs, learning from numerical data and expertise, are both interpretable and efficient in forecasting trip interchanges even if the data is large and noisy; they do not require data and can be established with only basic human reasoning; additional inputs can be introduced to the model easily as in neural networks.

iv. GFRBSs offer high level of accuracy in trip distribution modelling, although it brings additional computation cost; they should be preferred especially when high accuracy is needed or when system's complexity increases and classical rule base learning approaches fail.

Table 7.1: An Evaluation of Trip Distribution Models for the Doubly-Constrained Case

CRITERIA	**DCGM ML Estimation**	**DCGM WLS Estimation**	**DCGM TLD Based Est.**	**NNTDM**	**FRBS Design**	**GFRBS Design**
Mathematical Simplicity	Moderate	Moderate	Strong	Moderate	Strong	Moderate
Application Simplicity	Strong	Strong	Strong	Moderate	Strong	Weak
Statistical Interpretability	Strong	Strong	Strong	Unknown	Unknown	Unknown
System Interpretability	Moderate	Moderate	Moderate	Weak	Strong	Strong
Prediction Ability	Moderate	Weak	Moderate	Weak	Moderate	Strong
Improvement with Additional Variables	Weak	Weak	Weak	Strong	Strong	Strong
Dependency to Data Existence	Strong	Strong	Strong	Weak	Strong	Weak
Hybridization Ability	Weak	Weak	Weak	Strong	Strong	Strong
Computational Costs	Strong	Strong	Strong	Moderate	Strong	Weak
Ready to Use Software Packages	Strong	Weak	Strong	Strong	Strong	Weak

It is appropriate to stress here again that the present study has demonstrated how soft computing techniques, neural networks, fuzzy and genetic fuzzy systems, can successfully be used in modelling any spatial interactions. Among the other soft computing techniques, use of fuzzy set theory and fuzzy logic in spatial sciences has very promising features. In several scientific disciplines such as urban planning, transportation modelling, urban geography and regional science, the field data is generally involved uncertainty, vagueness and incompleteness. FRBSs provide an outstanding opportunity to deal with this drawback. They enable using linguistic terms and human like reasoning in modelling complex real-life systems. Additionally, they have an equation-free structure; they are not black-box, in contrast interpretable ; they can be used without numerical data; they can incorporate expert knowledge into modelling procedure; they provide a flexible framework for hybridization with other soft computing techniques such as genetic algorithms and neural networks.

In conclusion, fuzzy and genetic fuzzy systems offer an alternative way to traditional gravity models and neural networks in modelling trip distributions. The present study has demonstrated their applicability to a challenging city region with a desired level of accuracy and interpretability. There are several positive reasons for continuing to develop trip distribution models in this fields.

This study differs from similar previous works (Kalic and Teodorovic, 1996; 2003 and Shafahi et al., 2008) in several respects: i) a genetic fuzzy system was proposed to model intra-city passenger flows for the first time; ii) original solutions to the fuzzy rule base learning problem were developed; iii) an extensive performance comparison was established for the first time, among the fuzzy, genetic fuzzy, doubly-constrained gravity and neural networks based trip distribution models.

Further researches should explore designing such fuzzy and genetic fuzzy systems with some new features such as : with additional variables, for instance a zonal land use variable or a geographical barrier can be introduced as additional inputs; with same variables but different configurations such as with a Sugeno-type FRBS; with innovative evolutionary algorithms and learning strategies; and most importantly with an approximate FRBS design that works properly in a low quality data environment or under uncertainty and imprecision. There is still a few number of studies that demonstrate such systems' ability on an environment where uncertainty and imprecision exist.

REFERENCES

Alonso, W. 1973, *National interregional demographic accounts: a prototype*, Institute of Urban and Regional Development, University of California, Berkeley, Rep. No. 17.

Alonso, W. 1978, "A theory of movements," *In Human Settlement Systems: International Perspectives on Structure, Change and Public Policy*, N. M. Hansen, ed., Cambridge, Massachusetts: Ballinger Publishing Company, pp. 197-211.

Avineri, E. 2005. Soft computing applications in traffic and transport systems: a review. *Advances in Soft Computing*, 1, 17-25

Batten, D. F. & Boyce, D. E. 1987, "Spatial interaction, transportation, and interregional commodity flow models," *In Handbook of Regional and Urban Economics - Volume 1*, ed. N. Peter, ed., Elsevier, pp. 357-406.

Bezdek, J. C. 1994, "What is computational intelligence?," *In Computational Intelligence Imitating Life*, J. M. Zurada, R. J. Marks, & C. J. Robinson, eds., NY: IEEE Press, pp. 1-12.

Black, W.R. 1995. Spatial interaction modeling using artificial neural networks. *Journal of Transport Geography*, 3, (3) 159-166

Black, W.R. 2003. *Transportation: a geographical analysis* New York - London, The Guilford Press.

Bodenhofer, U. & Herrera, F. 1997, *Ten Lectures on Genetic Fuzzy Systems*, Software Competence Center Hagenberg, Austria, SCCH-TR-0021.

Bodenhofer, U. 1999, *Genetic Algorithms: theory and applications*, Software Competence Center Hagenberg, Austria, SCCH-TR-0019.

Boyce, D. 2002. Is the sequential travel forecasting paradigm counterproductive? *Journal of Urban Planning and Development*, 128, (4) 169-183

Carey, H.C. 1858. *Principles of social sciences* Philadelphia, J. B. Lippincott.

Cascetta, E., Pagliara, F., & Papolla, A. 2007. Alternative approaches to trip distribution modelling: a retrospective review and suggestions for combining different approaches. *Papers in Regional Science* , 86, (4) 597-620

Casey, H.J. 1955. Applications to traffic engineering of the law of retail gravitation. *Traffic Quarterly*, 9, 23-25

Celik, M.H. 2004. Forecasting Interregional commodity flows using artificial neural networks: an evaluation. *Transportation Planning and Technology*, 27, (6) 449-467

Chiou, Y.C. & Lan, L.W. 2005. Genetic fuzzy logic controller: an iterative evolution algorithm with new encoding method. *Fuzzy Sets and Systems*, 152, (3) 617-635

Cordon, O., Herrera, F., Hoffmann, F., & Magdalena, L. 2001. *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases* Singapore, World Scientific.

Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., & Magdalena, L. 2004. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141, (1) 5-31

Demuth, H., Beale, M., & Hagan, M. 2009. *Neural Network Toolbox 6: users guide* Natick, MA, The MathWorks, Inc.

Dickey, J.W. 1983. *Metropolitan transportation planning*, 2nd edition ed. New York, McGraw-Hill.

Diplock, G. & Openshaw, S. 1996. Using Simple genetic algorithms to calibrate spatial interaction models. *Geographical Analysis*, 28, (3) 262-279

Dougherty, M. 1995. A Review Of Neural Networks Applied To Transport. *Transportation Research Part C: Emerging Technologies*, 3, (4) 247-260

Easa, S.M. 1993. Urban trip distribution in practice .1. conventional analysis. *Journal of Transportation Engineering-Asce*, 119, (6) 793-815

Evans, A.W. 1971. The calibration of trip distribution models with exponential or similar cost functions. *Transportation Research*, 5, (1) 15-38

Fischer, M.M. & Gopal, S. 1994. Artificial neural networks - a new approach to modeling interregional telecommunication flows. *Journal of Regional Science*, 34, (4) 503-527

Fischer, M. M. 2001, "Neural spatial interaction models," *In Geocomputational Modelling: Techniques and Applications*, M. M. Fischer & Y. Leung, eds., Heidelberg: Springer, pp. 195-219.

Fischer, M.M., Reismann, M., & Hlavackova-Schindler, K. 2003. Neural Network modeling of constrained spatial interaction flows: design, estimation, and performance issues. *Journal of Regional Science*, 43, (1) 35-61

Fischer, M. M. 2009, "Principles of neural spatial interaction modeling," *In Tool Kits in Regional Science:Theory, Models, and Estimation*, M. Sonis & J. D. H. Hewings, eds., Springer Berlin Heidelberg, pp. 199-214.

Fotheringham, A.S. 1983. A new set of spatial-interaction models: the theory of competing destinations. *Environment and Planning A*, 15, (1) 15-36

Fotheringham, A.S. & Knudsen, D.C. 1987. *Goodness-of-fit statistics* Norvich, Geo Books.

Fotheringham, A.S. & O'Kelly, M.E. 1989. *Spatial Interaction models: formulations and applications* Dordrecht - Boston - London, Kluwer Academic Publishers.

Furness, K.P. 1965. Time Function Iteration. *Traffic Engineering and Control*, 7, 458-460

Gen, M. & Cheng, R. 2000. *Genetic algorithms & engineering optimization* New York, John Wiley & Sons.

Goldberg, D.E. 1989. *Genetic algorithms in search, optimization, and machine learning* Boston , Addison-Wesley.

Gray, R.H. & Sen, A.K. 1983. Estimating gravity model parameters: A simplified approach based on the odds ratio. *Transportation Research Part B: Methodological*, 17, (2) 117-131

Hansen, W.G. 1959. How accessibility shapes land use. *Journal of the American Institute of Planners*, 25, (2) 73-76

Haupt, R.L. & Haupt, S.E. 2004. *Practical genetic algorithms*, 2nd ed. New Jersey, John Wiley & Sons.

Haykin, S. 1999. *Neural Networks: a comprehensive foundation*, 2nd ed. India, Pearson Prentice Hall.

Henn, V. 2000. Fuzzy route choice model for traffic assignment. *Fuzzy Sets and Systems*, 116, (1) 77-101

Herrera, F. 2005. Genetic fuzzy sstems status critical considerations and future directions. *International Journal of Computational Intelligence Research*, 1, (1) 59-67

Herrera, F. 2008. Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1, (1) 27-46

Herrera, F. & Vergeday J.L. (eds.) 1996. *Genetic algorithms and soft computing*, Heidelberg, Physica-Verlag.

Holland, J.H. 1975. *Adaptation in natural and artificial systems* Ann Arbor, University of Michigan Press.

Hornik, K., Stinchcombe, M., & White, H. 1989. multilayer feedforward networks are universal approximators. *Neural Networks*, 2, (5) 359-366

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4, (2) 251-257

Huff, D.L. 1963. A probabilistic analysis of shopping center trade areas. *Land Economics*, 39, (1) 81-90

Huff, D.L. 1965. A note on the limitations of intraurban gravity models. *Land Economics*, 38, (1) 64-66

Hyman, G.M. 1969. The calibration of trip distribution models. *Environment and Planning A*, 1, (1) 105-112

Ishak, S. & Franco, T. 2007, "Neural Networks," *In Transportation Research Circular E-C113: Artificial Intelligence in Transportation Information for Application*, TRB Artificial Intelligence and Advanced Computing committee, ed., Washington, DC: Transportation Research Board, pp. 17-32.

Ishibuchi, H. 2007. Multiobjective genetic fuzzy systems: review and future research directions, *In Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pp. 1-6.

Istanbul Metropolitan Municipality Transportation Planning Department 2008. *Istanbul Transportation Master Plan Household Survey: analytic study and model Calibration* Istanbul, Metropolitan Municipality of Istanbul.

Istanbul Metropolitan Planning Centre (IMP) 2006, *Istanbul Metropolitan Area Master Plan Report,* Metropolitan Municipality of Istanbul, Istanbul.

Ji-Rong, X. 2000. *A neural network approach to modelling and predicting intercity passenger flows.* (PhD Dissertation). Indiana University.

Kalic, M. & Teodorovic, D. 2003. Trip distribution modelling using fuzzy logic and a genetic algorithm. *Transportation Planning and Technology*, 26, (3) 213-238

Kalic, M. & Teodorovic, D. 1996. Solving the trip distribution problem by fuzzy rules generated by learning from examples (in Serbian), *In Proceedings of the XXIII Yugoslav Symposium on Operations Research*, pp. 777-780.

Kanafani, A. 1983. *Transportation demand analysis* New York, McGraw-Hill.

Karr, C. 1991. Genetic algorithms for fuzzy controllers. *AI Expert,* 6, (2) 26-33

Kim, D. 2001. Neural Networks for trip generation model. *Journal of the Eastern Asia Society for Transportation Studies*, 4, (2) 201-208

Knudsen, D.C. & Fotheringham, A.S. 1986. Matrix comparison, goodness-of-fit and spatial interaction modeling. *Journal of Regional Science Review*, 10, (2) 127-147

Konar, A. 2005. *Computational intelligence: principles, techniques and applications* Netherlands, Springer-Verlag Berlin Heidelberg.

Kosko, B. 1994. Fuzzy systems as universal approximators. *Computers, IEEE Transactions on*, 43, (11) 1329-1333

Leung, S.C.H. 2007. A non-linear goal programming model and solution method for the multi-objective trip distribution problem in transportation engineering. *Optimization and Engineering*, 8, (3) 277-298

Lotan, T. & Koutsopoulos, H.N. 1993. Models for route choice behavior in the presence of information using concepts from fuzzy set theory and approximate reasoning. *Transportation*, 20, (2) 129-155

Lowry, I. S. 1964, *A model of metropolis*, The Rand Corporation, Santa Monica , Memorandum, RM-4035-RC.

Mamdani, E.H. 1974. Application of fuzzy algorithms for control of simple dynamic plant. *Procedings of IEEE*, 121, (12) 1585-1588

Mamdani, E.H. & Assilian, S. 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7, (1) 1-13

Martin, B. V., Memmott, F. W., & Bone, A. j. 1961, *Principles and techniques of predicting future demand for urban area transportation* M.I.T., Rep. No. 38.

McFadden, D. 1974, "Conditional logit analysis of qualitative choice behavior," *In Frontiers In Econometrics*, P. Zarembka, ed., New york: Academic press, pp. 105-142.

Mendel, J.M. & Mouzouris, G.C. 1997. Designing fuzzy logic systems. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 44, (11) 885-895

Meyer, M.D. & Miller, E.J. 2001. *Urban transportation planning: a decision-oriented approach*, Second Edition ed. Boston, McGraw-Hill.

Mozolin, M., Thill, J.-C., & Lynn Usery, E. 2000. Trip distribution forecasting with multilayer perceptron neural networks: A critical evaluation. *Transportation Research Part B: Methodological*, 34, (1) 53-73

Munakata, T. 2008. *Fundamentals of the new artificial intelligence: neural evolutionary fuzzy and more*, 2nd ed. ed. Springer.

Mussone, L. 1999. A review of feedforward neural networks in transportation research. *e & i Elektrotechnik und Informationstechnik*, 116, (6) 360-365

Openshaw, S. 1993, "Modeling spatial interaction using neural net," *In Geographical Information Systems, Spatial Modeling, and Policy evaluation* , M. M. Fischer & P. Nijkamp, eds., New York: Springer Berlin, pp. 147-164.

Openshaw, S. 1997, "Building fuzzy spatial interaction models," *In Advances in Spatial Analysis*, A. Getis & M. M. Fischer, eds., Berlin: Springer, pp. 360-383.

Openshaw, S. 1998. Neural network, genetic, and fuzzy logic models of spatial interaction. *Environment and Planning A*, 30, (10) 1857-1872

Oppenheim, N. 1995. *Urban travel demand modeling: From Individual Choices to General Equilibrium* New York, Wiley Interscience.

Ortuzar Juan de Dios & Willumsen Luis G. 2001. *Modelling transport*, Third Ed. ed. Chichester, John Wiley & Sons.

Pedrycz, J. 1996. *Fuzzy sets engineering* Boca Raton, CRC Press.

Pedrycz, J. & Gomide, F. 1998. *An introduction to fuzzy sets: Analysis and Design* Cambridge, Massachusetts, MIT Press.

Pham, D.T. & Karaboga, D. 1991. Optimum design of fuzzy logic controllers using genetic algorithms. *Journal of Systems Engineering*, 2, (1), 114-118

Ravenstein, E.G. 1885. The laws of migration. *Journal of the Statistical Society of London*, 48, (2) 167-235

Ravenstein, E.G. 1889. The laws of migration. *Journal of the Royal Statistical Society*, 52, (2) 241-305

Ross, T.J. 2004. *Fuzzy logic with engineering applications*, Second edition ed. England, John Wiley & Sons.

Roy, J.R. 2004. *Spatial interaction modelling: A Regional Science Context* Berlin, Springer - Verlag .

Roy, J.R. & Thill, J.C. 2004. Spatial interaction modelling. *Papers in Regional Science*, 83, (1) 339-361

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. 1986. Learning representations by back-propagating errors. *Nature*, 323, (6088) 533-536

Sadek, A. W. 2007, "Artificial intelligence applications in transportation," *In Transportation Research Circular E-C113: Artificial Intelligence in Transportation Information for Application*, TRB Artificial Intelligence and Advanced Computing committee, ed., Washington, DC: Transportation Research Board, pp. 1-6.

Schneider, M. 1959. Gravity models and trip distribution theory. *Papers and Proceedings of the Regional Science Association*, 5, 51-56

Sen, A. & Soot, S. 1981. Selected procedures for calibrating the generalized gravity model. *Papers in Regional Science*, 48, (1) 165-176

Sen, A. 1986. Maximum-likelihood-estimation of gravity model parameters. *Journal of Regional Science*, 26, (3) 461-474

Sen, A. & Smith, T.E. 1995. *Gravity models of spatial interaction behavior* Berlin - Heidelberg, Springer-Verlag.

Shafahi, Y., Nourbakhsh, S. M., & Seyedabrishami, S. 2008. Fuzzy trip distribution models for discretionary trips, *In Intelligent Transportation Systems, 2008. 11th International IEEE Conference on*, pp. 557-562.

Sivanandam, S.N. & Deepa, S.N. 2008. *Introduction to genetic algorithms* Berlin Heidelberg, Springer-Verlag.

Smith, D.P. & Hutchinson, B.G. 1981. Goodness of fit statistics for trip distribution models. *Transportation Research Part A: General*, 15, (4) 295-303

Smith, S.F. 1980. *A learning system based on genetic adaptive algorithms.* (Doctoral Dissertation). Department of Computer Science.

Stouffer, S.A. 1940. Intervening opportunities: a theory relating mobility and distance. *American Sociological Review*, 5, (6) 845-867

Sugeno, M. & Kang, G.T. 1988. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28, (1) 15-33

Taaffe, E.J., Gauthier, H.L., & O'Kelly, M.E. 1996. *Geography of transportation*, 2nd ed. New Jersey, Prentice Hall.

Takagi, T. & Sugeno, M. 1985. Fuzzy Identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15, (1) 116-132

Tapkin, S. & Akyilmaz, Ö. 2009. A new approach to neural trip distribution models: NETDIM. *Transportation Planning and Technology*, 32, (1) 93-114

Teodorovic, D. & Kikuchi, S. 1990. Transportation route choice model using fuzzy inference technique, *In Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on*, pp. 140-145.

Teodorovic, D. 1994. Fuzzy sets theory applications in traffic and transportation. *European Journal of Operational Research*, 74, (3) 379-390

Teodorovic, D. & Vukadinovic, K. 1998. *Traffic control and transport planning: a fuzzy sets and neural networks Approach* Dordrecht, Kluwer.

Teodorovic, D. 1999. Fuzzy logic systems for transportation engineering: the state of the art. *Transportation Research Part A: Policy and Practice*, 33, (5) 337-364

Thrift, P. 1991. Fuzzy logic synthesis with genetic algorithms. *In Proceedings of fourth international conference on henetic algorithms (ICGA'91), Morgan Kaufman, San Diego,* pp. 509-513.

Tillema, F., van Zuilekom, K.M., & van Maarseveen, M.F.A.M. 2006. Comparison of neural networks and gravity models in trip distribution. *Computer-Aided Civil and Infrastructure Engineering*, 21, 104-119

TRB - Transportation Reseach Board 1998. *Travel estimation techniques for urban planning (NCHRP Report 365)* Washington, D.C., National Academy Press.

TRB - Transportation Reseach Board 2007. *Artificial intelligence in transportation: information for application* Washignton.

TurkStat 2010, *Press Release: address based population registration system population census results, 2009*, Turkish Statistical Institute, Ankara, 15.

Valenzula-Rendon, M. 1991. The fuzzy classifier system: motivations and first results. *In Proceedings of first international conference on parallel problem solving from nature*, Scwefel H.P. & Manner R., eds., Berlin: Springer, pp. 330- 334.

Vuchic, V.R. 2005. *Urban transit: operations, planning, and economics* New Jersey, John Wiley & Sons.

Wang, L. X. Fuzzy systems are universal approximators, *In Fuzzy Systems, 1992., IEEE International Conference on*, pp. 1163-1170.

Wang, L.-X. & Mendel, J.M. 1992. Generating fuzzy rules by learning from examples. *Systems, Man and Cybernetics, IEEE Transactions on*, 22, (6) 1414-1427

Williams, H.C.W.L. 1977. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and Planning A*, 9, (3) 285-344

Williams, I. 1976. A comparison of some calibration techniques for doubly constrained models with an exponential cost function. *Transportation Research*, 10, (2) 91-104

Wilson, A.G. 1967. A statistical theory of spatial distributions. *Transportation Research*, 1, 253-269

Wilson, A.G. 1970. *Entropy in urban and regional modelling* London, Pion.

Wilson, A.G. 1974. *Urban and regional models in geography and planning* London - New York, John Wiley & Sons.

Zadeh, L.A. 1965. Fuzzy sets. *Information and Control*, 8, (3) 338-353

Zadeh, L.A. 1973. Outline of a new approach to the analysis of complex systems and decision processes. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3, (1) 28-44

APPENDIX A

COMPUTER PROGRAMS

Present study deals with large data sets and includes a number of iterative mathematical algorithms. Additionally, many of the algorithms used in this study are not included in built-up software packages. So, it is found appropriate to create own computer programs/codes for nearly all mathematical and statistical algorithms.

The selected programming/computing software is MATLAB and used version is R2009b. The MATLAB product family provides a high-level programming language, an interactive technical computing environment, and functions for: algorithm development, data analysis and numeric computation.

The following sections include calibration, training and learning algorithms for each of the trip distribution models introduced earlier. Description of the program and algorithm details are given with '%' denotes. One can run any of the program, if he or she copy and paste the code to a MATLAB script file (.m - file) and load a suitable data set.

A.1. Doubly-Constrained Gravity Model Calibration with Maximum Likelihood Estimation

```
---------------------------------------------------------------------------------------------------------------------------------------
%% MAXIMUM LIKELIHOOD CALIBRATION ALGORITHM FOR DOUBLYCONSTRAINED GRAVITY MODEL

%% Definition of the Program
%This program calibrates friction factor parameter (beta) for Doubly Constraint GM
%The code is written for both of the Cost Functions: Power (cij^-b) and Exponential(e^-b*cij)
%Calibration criteria is to replicate the Observed Total Travel Cost
%See detailed formula and its explanations in Fortheringham and O'Kelly (1989), at pages 49-56
%See Hyman's iterative parameter estimation method in Hyman (1969)
%At the end, code produces an estimate of beta parameter (b) and modelled trip matrix with the parameter estimate
%Produced in            :13.08.2009
%Last Modified in       :08.06.2010                    ***written by Mert Kompil     ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load -mat HBW_TRAIN_DATA;       %load data from a -mat file (-mat file includes FRICTION_TRAIN and
                                OBS_TRIPS_TRAIN matrices)

%% Select Cost Function Type (Power or Exponential)
answer        = input(' Please Select The Cost Function :\n Enter "1" for POWER and "2" for EXPONENTIAL
FUNCTIONS  ----- > ');
if  answer     ==1;
   type=1;
else
   type=2;
end;

%% Rename Data Matrices
t_time=HBW_FRICTION_TRAIN;              %enter the name of friction matrix here (distanc/time)(cij)
obs_trips=HBW_OBS_TRAIN;                %enter the name of trip matrix (Tij)
clear ('HBW_FRICTION_TRAIN','answer','HBW_OBS_TRAIN', 'HBW_ROW_DATA_TRAIN')
 %Clear old matrixes

%% Enter Maximum Iteration Numbers and Convergence Criteria
iter1=100;                      %maximum iteration number for the loop finding beta (b) parameter
iter2=100;                      %maximum iteration number for the loop finding the balancing factors (Ai&Bj)
conv1=0.001;                    %convergence criteria for balancing factors (Ai&Bj)
conv2=0.0001;                   %convergence criteria for beta (b)parameter estimate

%% Create Starting matrices and parameters
prod_tot=sum(obs_trips,2)';             %produce production totals matrix
attr_tot=sum(obs_trips);                %produce attraction totals matrix
logt_time=log(t_time);                  %take log of friction matrix
if  type==1                             %compute Observed Total Travel Cost
   obs_ttc=sum(sum(obs_trips.*logt_time));          %use "log(cij)" for Power Cost Function
else
   obs_ttc=sum(sum(obs_trips.*t_time));             %use only "cij" for Exponential Cost Function
end;
b=ones(iter1+1,1);                      %produce a row vector for beta (b) parameters
b(1)=(3/(2*obs_ttc));                   %compute the initial beta (b0) parameter
mod_ttc=ones(iter1+1,1);                %produce a row vector for Modeled Total Travel Costs (c0,c1,...cn) values
[m,n]=size(obs_trips);                  %produce matrix index
mod_trips=ones(m,n);                    %modelled trip matrix
A0=ones(m,n);                           %computation table for Ais
B0=ones(m,n);                           %computation table for Bjs
Ai=ones(m,iter2);                       %produce a matrix for Ais to make comparison through iterations
Bj=ones(iter2,n);                       %produce a matrix for Bjs to make comparison through iterations
B=ones(1,n);                            %set the initial Bjs as one
 %% Start Iteration Loops
for r=1:iter1                   %START LOOP 1: for beta (b) parameter
```

```
%% Find Balancing Factors (Ais&Bjs)
    for k=1:iter2          %START LOOP 2: for balancing factors (Ai&Bj)
        if type==1                      %if the Power Cost function is selected
           for i=1:m,                   %find Ais
              for j=1:n
                 A0(i,j)=B(j)*attr_tot(j)*(t_time(i,j)^b(r));
              end;
           end;
              A=(1./sum(A0,2))';
           for i=1:m                    %find Bjs
              for j=1:n
                 B0(i,j)=A(i)*prod_tot(i)*(t_time(i,j)^b(r));
              end;
           end;
        end;
        if type==2                      %if the Exponential Cost function is selected
           for i=1:m,                   %find Ais
              for j=1:n
                 A0(i,j)=B(j)*attr_tot(j)*(exp(t_time(i,j)*b(r)));
              end;
           end;
              A=(1./sum(A0,2))';
           for i=1:m                    %find Bjs
              for j=1:n
                 B0(i,j)=A(i)*prod_tot(i)*(exp(t_time(i,j)*b(r)));
              end;
           end;
        end;
        B=(1./sum(B0));
        Ai(:,k)=A;                      %write Ais to the related column in matrix for comparison
        Bj(k,:)=B;                      %write Bjs to the related row in matrix for comparison
        if k>1                          %compare balancing factors
           if sqrt((sum((Ai(:,k)-Ai(:,(k-1))).^2))/k)< conv1 && sqrt(sum((Bj(k,:)-Bj((k-1),:).^2),2)/k) < conv1;
           break                        %if convergence criteria is satisfied, stop process, otherwise continue
           end;
        end;
    end;                %END OF LOOP 2: for balancing factors (Ai&Bj)

%% Compute Modeled Trip Matrix
    if type==1                          %if the Power Cost function is selected
       for i=1:m,
          for j=1:n
             mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(t_time(i,j)^b(r));          %compute the modelled trips
          end;
      end;
    end;
    if type==2                          %if the Exponential Cost function is selected
       for i=1:m,
          for j=1:n
             mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(exp(t_time(i,j)*b(r)));     %compute the modelled trips
          end;
      end;
    end;

%% Estimate a Better beta (b) Parameter
    if  type==1                                 %compute Modeled Total Travel Cost (C0,C1,...Cn)
       mod_ttc(r)=sum(sum(mod_trips.*logt_time));                %use "log(cij)" for Power Cost Function
    else
       mod_ttc(r)=sum(sum(mod_trips.*t_time));                   %use only "cij" for Exponential Cost Function
    end;
    if  r==1                            % use this formula for the first iteration (b1) to estimate beta parameter
       b(r+1)=b(r)*mod_ttc(r)/obs_ttc;
    else                                % use this formula for subsequent iterations (b2,b3,...bn)
       b(r+1)=((obs_ttc-mod_ttc(r-1))*b(r)-((obs_ttc-mod_ttc(r))*b(r-1)))/(mod_ttc(r)-mod_ttc(r-1));
    end;
    if  r>1
       if  abs((b(r+1))-(b(r))) < (conv2)       % compare the last two beta (b) estimates
```

```
        break                                   % if convergence criteria is satisfied, stop process, otherwise continue
        end;
      end;
end;                            %END OF LOOP 1: for beta (b) parameter

%% Show the Estimated Parameter on Screen
beta=b(r+1);
if  type==1
   fprintf('\n beta calibration for Power Cost Function is successfully completed in %g iterations \n',r)
else
   fprintf('\n beta calibration for Exponential Cost Function is successfully completed in %g iterations \n',r)
end;
fprintf('\n beta ----> %f\n',beta)                  %show the best value estimate for beta (b) parameter on the screen
clear ('obs_ttc','iter1','iter2','conv1','conv2','A','B','m','n','mod_ttc','logt_time',...
     'k','r','i','j','b','c','prod_tot','attr_tot','type','A0','Ai','B0','Bj')    %clear temporary matrixes

%% Output of the Program
i)   the modelled trips (mod_trips) and,
ii)  the beta (beta) parameter estimate
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

A.2. Doubly-Constrained Gravity Model Calibration with Trip Length Distribution Based Estimation

```
---------------------------------------------------------------------------------------------------------------------------
%% MINIMIZING TRIP LENGTH DISTRIBUTION RMSE WİTH A LINE SEARCH ALGORITHM

%% Definition of the Program
%The code is written for both of the Cost Functions: Power (cij^-b) and Exponential(e^-b*cij)
%Calibration criteria is to replicate the Observed Trip Length Distribution according to the given trip length intervals
%During the process, you have to set an interval number (bin interval) and a maximum value (end of bins)
%At the end, code produces an estimate of beta parameter (b) and, Observed and Modelled Trip Length Distributions
%Produced in          :10.08.2009
%Last Modified in     :08.06.2010                    ***written by Mert Kompil    ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load -mat HBW_Train_Data;          %load data from a -mat file (-mat file includes DISTANCE_TRAIN,
                                   OBS_TRIPS_TRAIN and ROW_DATA_TRAIN matrices)

%% Select Cost Function Type (Power or Exponential)
answer       = input(' Please Select The Cost Function :\n Enter "1" for POWER and "2" for EXPONENTIAL
FUNCTIONS  ----- > ');
if  answer     ==1;
   type=1;
else
   type=2;
end;
%% Set an Interval Number for Each Bins
answer1       = input('\n Please Enter an Interval Number for Each Bins  ----- > ');

%% Set a Value That The Bins End
answer2       = input('\n Please Enter a Value that the Bins End  ----- > ');

%% Rename Data Matrices
t_time=HBW_FRICTION_TRAIN;                    %enter the name of friction matrix here (distanc/time)(cij)
obs_trips=HBW_OBS_TRAIN;                      %enter the name of trip matrix (Tij)
row_data=HBW_ROW_DATA_TRAIN;
clear ('HBW_FRICTION_TRAIN','answer','HBW_OBS_TRIPS_TRAIN', 'HBW_ROW_DATA_TRAIN')
%Clear old matrixes

%% Set a Vector of beta (b) parameters for line search
b=(-0.1:-0.1:3)';                  %For more than one parameter, the input should be a column vector
[w,c]=size(b);                     %matrix index

%% Compute Observed Trip Length Distribution
maxi=max(max(t_time));                        %the maximum friction value
mini=0;                                       %the minimum friction value - set zero as default value
bin_int=answer1;                              %interval for each bins
max_bin=answer2;                              %the value that bins end
num_bins=floor((max_bin/bin_int)+1);          %number of bins
OTLD=zeros(num_bins,4);                       %create a matrix for observed trip length distribution
OTLD(:,1)=(1:1:num_bins);
OTLD(:,2)=(bin_int:bin_int:(max_bin+bin_int))';
OTLD(num_bins,2)=maxi;
[h,s]=size(row_data);                         %matrix index
for i=1:h                                     %produce observed trip length distribution
   for k=1:num_bins-1
      if mini <= row_data(i,5) && row_data(i,5)< OTLD(1,2)
        OTLD(1,3)= OTLD(1,3)+row_data(i,6);
        break
      end;
      if OTLD(k,2) <= row_data(i,5) && row_data(i,5)< OTLD(k+1,2)
        OTLD(k+1,3)= OTLD(k+1,3)+row_data(i,6);
        break
```

```
      end;
   end;
end;
clear ('answer1','answer2','k')                    %Clear temporary values

%% Enter Maximum Iteration Number and Convergence Criteria for Balancing Factors
iter=100;                                          %maximum iteration number for the loop finding the balancing
factors (Ai&Bj)
conv=0.001;                                        %convergence criteria for balancing factors (Ai&Bj)

%% Create Starting matrices and parameters for modeled trips
prod_tot=sum(obs_trips,2)';                        %produce production totals matrix
attr_tot=sum(obs_trips);                           %produce attraction totals matrix
rmse=ones(w,1);                                    %create a column vector for RMSE Results
row_data_2=ones(h,c);                              %create a matrix to write modeled tirps in rows
MTLD00=zeros(num_bins,w);                          %create a matrix to write modeled Trip Length Distribution Counts
MTLD01=MTLD00;                                     %create a matrix to write modeled TLD  Percents
[m,n]=size(obs_trips);                             %matrix index
mod_trips=ones(m,n);                               %create a matrix for modeled trip matrix
A0=ones(m,n);                                      %computation table for Ais
B0=ones(m,n);                                      %computation table for Bjs
Ai=ones(m,iter);                                   %produce a matrix for Ais to make comparison through iterations
Bj=ones(iter,n);                                   %produce a matrix for Bjs to make comparison through iterations
B=ones(1,n);                                       %set the initial Bjs as one
%% Start Computation for each beta (b) input
for g=1:w          %START LOOP 1: produce modeled trip matrix for given beta (b)

%% Find Balancing Factors (Ais&Bjs)
      for k=1:iter          %START LOOP 2: for balancing factors (Ai&Bj)
            if type==1                             %if the Power Cost function is selected
               for i=1:m,                          %find Ais
                  for j=1:n
                     A0(i,j)=B(j)*attr_tot(j)*(t_time(i,j)^b(g));
                  end;
               end;
                  A=(1./sum(A0,2))';
               for i=1:m                           %find Bjs
                  for j=1:n
                     B0(i,j)=A(i)*prod_tot(i)*(t_time(i,j)^b(g));
                  end;
               end;
            end;
            if type==2                             %if the Exponential Cost function is selected
               for i=1:m,                          %find Ais
                  for j=1:n
                     A0(i,j)=B(j)*attr_tot(j)*(exp(t_time(i,j)*b(g)));
                  end;
               end;
                  A=(1./sum(A0,2))';
               for i=1:m                           %find Bjs
                  for j=1:n
                     B0(i,j)=A(i)*prod_tot(i)*(exp(t_time(i,j)*b(g)));
                  end;
               end;
            end;
            B=(1./sum(B0));
            Ai(:,k)=A;                             %write Ais to the related column in matrix for comparison
            Bj(k,:)=B;                             %write Bjs to the related row in matrix for comparison
            if k>1                                 %compare balancing factors
               if sqrt((sum((Ai(:,k)-Ai(:,(k-1))).^2))/k)< conv && sqrt(sum((Bj(k,:)-Bj((k-1),:).^2),2)/k) < conv;
               break                               %if convergence criteria is satisfied, stop process, otherwise continue
               end;
            end;
      end;              %END OF LOOP 2: for balancing factors (Ai&Bj)
      clear ('k','i','j')    %clear temporary values

%% Compute Modeled Trip Matrix
```

```
    if type==1                          %if the Power Cost function is selected
       for i=1:m,
          for j=1:n
             mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(t_time(i,j)^b(g));          %compute the modeled trips
          end;
       end;
    end;
    if type==2                          %if the Exponential Cost function is selected
       for i=1:m,
          for j=1:n
             mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(exp(t_time(i,j)*b(g)));      %compute the modeled trips
          end;
       end;
    end;

%% Write Modeled Trip Matrix to Rows to Compute Modeled TLD
    v=1;
    for o=1:m
       temp=mod_trips(:,v);
       row_data_2(o,g)=temp(o);
    end;
    v=v+1;
    t=1;
    x=0;
    for u=1:n-1
       for f=t:m*u
       temp=mod_trips(:,v);
       row_data_2(m+f,g)=temp(x+f);
       end;
       t=t+m;
       v=v+1;
       x=x-m;
    end;

%% Compute Modeled TLD
    for i=1:h
       for k=1:num_bins-1
          if mini <= row_data(i,5) && row_data(i,5)< OTLD(1,2)
             MTLD00(1,g)= MTLD00(1,g)+row_data_2(i,g);
             break
          end;
          if OTLD(k,2) <= row_data(i,5) && row_data(i,5)< OTLD(k+1,2)
             MTLD00(k+1,g)= MTLD00(k+1,g)+row_data_2(i,g);
             break
          end;
       end;
    end;

%% Compute RMSE
    for i=1:num_bins
    OTLD(i,4)=(OTLD(i,3)/sum(OTLD(:,3)))*100;
    MTLD01(i,g)=(MTLD00(i,g)/sum(MTLD00(:,g)))*100;
    end;
    rmse(g) =(sqrt((sum((OTLD(:,4)-MTLD01(:,g)).^2))))/num_bins;
end;        %END OF LOOP 1: for beta (b) line search

%% Write Results in Matrices
results=ones(w,2);
results(:,1)=b;
results(:,2)=rmse(:,1);                 %combine parameter values and rmse results in matrix
sorted_results=sortrows(results,2);     %sort the rows according to the minimum to maximum rmse
beta=sorted_results(1,1);               %beta is the value which has minimum rmse
for i=1:w                               %find the beta value previous index and call TLDs results from
                                        previously created matrices
  if  beta==b(i,1);
    break
  end;
```

```
   index=i;
end;
TLD=OTLD;
TLD(:,5)=MTLD00(:,i);       %write modeled TLD counts and percents next to  observed TLD counts and percents
TLD(:,6)=MTLD01(:,i);

%% Show the Estimated Parameter and RMSE Plot on Screen
if  type==1
   fprintf('\n Line Search ALgorithm for Power Cost Function is successfully completed. \n %g paremeters have tried
\n',w)
else
   fprintf('\n Line Search ALgorithm for Exponential Cost Function is successfully completed. \n %g paremeters have
tried \n',w)
end;
fprintf('\n beta ----> %f\n',beta)                          %show the best value estimate for beta (b) parameter on the screen
plot(results(:,2)); figure(gcf); title('RMSE CHANGE')          %plot RMSE change

clear('A','A0','Ai','B','B0','Bj','MTLD00','MTLD01','OTLD','attr_tot','prod_tot','b','bin_int','num_bins',...
     'c', 'conv','f','g','h','i','index','iter','j','k','m','max_bin','maxi','mini','mod_trips','n','o',...
     'x', 'v','u','w','type','temp','t_time','s','t','row_data_2','row_data','rmse','results0','obs_trips');
%clear temporary matrices

%% Output of the Program
%   The outputs of the program is :
%   i)  The beta (b) parameter estimate which minimizes the difference between OTLD and MTLD
%   ii) The result matrix that includes beta (b) values and paired RMSEs
%   iii)A plot of RMSE Change
%   iv) Sorted result matrix that includes beta values with RMSE from lowest to highest order
%   v)  A final matrix that includes in order: bin no, bin interval, OTLD counts, OTLD percentage, MTLD counts and
MTLD percentage
-------------------------------------------------------------------------------------------------------------------------------------------
```

A.3. Weighted Least Squares Transformation for Doubly-Constrained Gravity Model Calibration

```
---------------------------------------------------------------------------------------------------------------------------
%% WEIGHTED LEAST SQUARES TRANSFORMATION FOR DOUBLY-CONSTRAINED GM
% Transform Approach: Add a Constant to Zero Interaction Cells or to Zero and Non-Zero Interaction Cells

%% Definition of the Program
%This transformation is especially for rectangle trip matrices, also it can be used for square matrices
%You can select one of the two cost functions (power and exponential)during the transformation
%The problem of zero interactions will be eliminated with the addition of a constant to trip matrix(to Zero Interaction
Cells or to Whole Cells)
%The regression equation and its definitions can be found in Sen and Soot(1981)or Fortheringham and O'Kelly
(1989) at pages 46-47
%At the end ,the code produces dependent variable(Y), independent variable(X) and Weight variable (W), all column
vectors in result matrix
%The output of the code can be calibrated in any statistical software having a tool for weighted least squares
estimation
%Produced in:07.08.2009
%Modified in:08.06.2010                    ***written by Mert Kompil     ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load -mat HBW_TRAIN_DATA;       %load data from a -mat file (-mat file includes DISTANCE_TRAIN,
                                        OBS_TRIPS_TRAIN and ROW_DATA_TRAIN matrices)

%% Select the Type of Cost Function (Power or Exponential)
answer1= input(' Please Select The Cost Function :\n Enter "1" for POWER and "2" for EXPONENTIAL
FUNCTIONS  ----- > ');
if  answer1==1;
   type=1;
else
   type=2;
end;

%% Select the Type of Adding a Constant to Trip Matrix: to Zero Interaction Cells or to Whole Cells
fprintf('\n Please Select to Add Predetermined Constant:')
answer2= input('\n Enter "1" for ZERO Interaction Cells or "2" for both ZERO and NON-ZERO Interaction Cells ----
- > ');
if  answer2==1;
   zeros=1;
else
   zeros=2;
end;

%% Set the Constant to add Trip Matrix
answer3= input('\n Enter the Constant to Add (such as "0.1","0.5" etc..) ----- > ');
add=answer3;                    %constant to add zero interaction cells or both zero and non-zero cells

%% Rename Data Matrices
t_time=HBW_FRICTION_TRAIN;              %enter the name of friction matrix here (distance/time)(cij)
obs_trips=HBW_OBS_TRAIN;                %enter the name of trip matrix (Tij)
row_data=HBW_ROW_DATA_TRAIN;            %enter the name of row data matrix
                                        %here the row_data includes TAZ IDs in column 1-2, frictions in
                                        %column 5 and interactions in column 6
clear
('HBW_FRICTION_TRAIN','HBW_OBS_TRIPS_TRAIN','HBW_ROW_DATA_TRAIN','answer1','answer2','answ
er3')    %Clear old matrixes

%% Create Starting matrices and Index Numbers
[m,n]=size(obs_trips);          %Matrix Index
[k,r]=size(row_data);           %Matrix Index
obs_trips2=ones(m,n);
row_data2=ones(k,2);
```

```
if  type==1
   t_time2=log(t_time);                    %Take Log of friction matrix for power cost function
   row_data2(:,1)=log(row_data(:,5));
else
   t_time2=t_time;                         %There is no need to take log of friction matrix for exponential cost function
   row_data2(:,1)=row_data(:,5);
end;
if  zeros==1                               %Add the predetermined constant only to zero interaction cells
   for i=1:m
      for j=1:n
         if  obs_trips(i,j)==0;
            obs_trips2(i,j)=log(add);
         else
            obs_trips2(i,j)=log(obs_trips(i,j));
         end;
      end;
   end;
   for s=1:k
      if  row_data (s,6)==0;
      row_data2(s,2)=add;
      else
      row_data2(s,2)=row_data(s,6);
      end;
   end;
else                                       %Add the predetermined constant to both zero and non-zero interaction cells
   obs_trips2=log((obs_trips+add));
   row_data2(:,2)=(row_data(:,6)+add);
end;
row_mean=mean(obs_trips2,2);               %compute row means for trips
row_mean2=mean(t_time2,2);                 %compute row means for frictions
column_mean=mean(obs_trips2)';             %compute column means for trips
column_mean2=mean(t_time2)';               %compute column means for frictions
grand_mean=mean(mean(obs_trips2));              %compute grand mean for trips
grand=(ones(k,1))*grand_mean;                   %write grand mean in a vector
grand_mean2=mean(mean(t_time2));                %compute grand mean for frictions
grand2=(ones(k,1))*grand_mean2;                 %write grand mean in a vector
weight=(row_data2(:,2).^0.5);              %compute the weight (W) variable from interactions
reg_mat=ones(k,8);                         %Create a matrix to write 8 columns of transformation variables

%% Compute Each Transformation Variables to Calculate Dependent (Y) and Independent (X) Variables
% Column 1
   reg_mat(:,1)=log(row_data2(:,2));
%%
% Column 2
for t=1:n-1
   if t==1
      for i=1:m
      reg_mat(i,2)=row_mean(i,1);
      reg_mat(((t*m)+i),2)=row_mean(i,1);
      end;
   else
      for i=1:m
      reg_mat(((t*m)+i),2)=row_mean(i,1);
      end;
   end;
end;
% Column 3
h=0;
for t=1:n
   for i=1:m
   reg_mat(((h*m)+i),3)=column_mean(t);
   end;
   h=h+1;
end;

% Column 4
   reg_mat(:,4)=grand(:,1);
```

```
% Column 5
   reg_mat(:,5)=row_data2(:,1);
% Column 6
for t=1:n-1
   if t==1
      for i=1:m
      reg_mat(i,6)=row_mean2(i,1);
      reg_mat(((t*m)+i),6)=row_mean2(i,1);
      end;
   else
      for i=1:m
      reg_mat(((t*m)+i),6)=row_mean2(i,1);
      end;
   end;
end;
% Column 7
h=0;
for t=1:n
   for i=1:m
   reg_mat(((h*m)+i),7)=column_mean2(t);
   end;
   h=h+1;
end;
% Column 8
   reg_mat(:,8)=grand2(:,1);

%% Compute Dependent Variable (Y), Independent Variable (X) and Weight (W)
regression_transformation=ones(k,3);                                    %result matrix
regression_transformation(:,1)=(reg_mat(:,1)-reg_mat(:,2)-reg_mat(:,3)+reg_mat(:,4));   %dependent variable (Y)
regression_transformation(:,2)=(reg_mat(:,5)-reg_mat(:,6)-reg_mat(:,7)+reg_mat(:,8));   %independent variable (X)
regression_transformation(:,3)=weight;                                  %weight (W)
if  type==1                                                             %Show transformation selections on screen
   fprintf('\n Transformation sucsessfully completed for Power Cost Function \n')
else
   fprintf('\n Transformation sucsessfully completed for Exponential Cost Function \n')
end;
if zeros==1
   fprintf('\n The constant "%g" added to zero interaction cells \n',add)
else
   fprintf('\n The constant "%g" added to both zero and non-zero interaction cells \n',add)
end;
clear('i','m','n','k','r','t','s','j','t_time','t_time2','obs_trips','obs_trips2','zeros',...
     'column_mean','column_mean2','grand_mean','grand_mean2','add','reg_mat','type',...
     'h','des_tot','orj_tot','weight','row_data','row_data2','row_mean','row_mean2','grand','grand2')
%clear temporary inputs

%% Output of the Program
%The outputs is 'regression_transformation' matrix which includes Y,X and W variables in an order
%You can save the matrix as a text file, writing command window "save filename.txt -ascii"
%Then saved file can be opened in any statistical software (SPSS)  in order to run weighted least squares estimation
```

A.4. Neural Network Based Trip Distribution Model (NNTDM) Training Algorithm

```
%% NEURAL NETWORK BASED TRIP DISTRIBUTION MODEL (NNTDM) TRAINING ALGORITHM

%% Definition of the Program
%This program trains a Neural Network using Matlab - Neural Network Toolbox
%The Neural Network Toolbox implementation issues can be seen in Demuth et.al (2009)
%The learning of the network is based on Levenberg-Marquardt Learning Algorithm
%The production, attraction and friction are the inputs, and trips are the output of the network
%At the end, code produces trained network and unscaled network outputs
%Produced in          :January - 2009
%Last Modified in     :23.06.2010                ***written by Mert Kompil     ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load HBW_TRAIN_DATA.mat  %load data from a -mat file (-mat file includes training data inputs and outputs)

%% Min-Max Normalization of data set between 0.1 and 1
input_data=HBW_ROW_DATA_TRAIN(:,3:5)';                  %rename input matrix
output_data=HBW_ROW_DATA_TRAIN(:,6)';                   %rename output vector
[sc_input,sc_tr_in] = mapminmax(input_data,0.1,1);      %normalize (scale)inputs
[sc_output,sc_tr_out] = mapminmax(output_data,0.1,1);   %normalize (scale)output
%unscale = mapminmax('reverse',simulated_result,sc_tr_out)   %use this expression in order
                                                        %to unscale the network output

%% Create and Configurate the Network
net = newff(sc_input,sc_output,9,{'logsig','logsig'});  %determine number of neurons and activation functions
rand('seed',333)                                        %set a random seed
net = init(net);                                        %initialize the network with random seed
net.inputs{1}.processParams{3}.ymin=0;                  %change predefined lower range for input activation
net.outputs{2}.processParams{2}.ymin=0;                 %change predefined lower range for output activation
net.divideParam.trainRatio = 0.8;                       %adjust training data ratio for over-training process
net.divideParam.valRatio = 0.2;                         %adjust validation data ratio for over-training process
net.divideParam.testRatio = 0;                          %set third part ratio as zero

%% Adjust Learning Algorithm and Parameters
net.trainFcn='trainlm';
net.trainParam.mu=0.2;                                  %set Initial Learning Rate
net.trainParam.mu_dec=0.1;                              %set Learning Rate decrease factor
net.trainParam.mu_inc=3;                                %set Learning Rate increase factor
net.trainParam.epochs=12;                       %determine maximum number of epoch
net.trainParam.max_fail=125;                    %determine maximum number of cross-validation checks to stop

%% Train Network
[net,tr] = train(net,sc_input,sc_output);       %train the network

%% Output of the Program
%The outputs are i)the trained network (net) and ii) training parameters and configuration (tr)
%After some trials, the network training has to be stopped at best epoch
%Then the best trained parameters can be used to simulate any appropriate data set
%The expression "sim(net,sc_input)" can be used to simulate the network
```

A.5. Goodness-of-Fit Statistics - Micro Level

```
---------------------------------------------------------------------------------------------------------------------------------
%% GOODNESS OF FIT STATISTICS FOR DOUBLY CONSTRAINT GRAVITY MODEL - MICRO LEVEL
% For SRMSE, r square, Slope, ARV, Phi Statistic

%% Definition of the Program
%This program uses previously estimated beta (b)parameter as an input to produce modeled trip matrix
%Then produces goodness of fit statistics comparing observed and modeled trips
%The code can be used for both of the Cost Functions: Power (cij^-b) and Exponential(e^-b*cij)
%The observed trip matrix can be square or rectangle
%See formulas of some of the used goodness of fit statistics in Fortheringham and O'Kelly (1986; 1989)and Smith
and Hutchinson (1981)
%At the end, the code produces i) Standardized Root Mean Square Error (SRMSE),ii) Straigt Line Statistics
(intercept, slope, r square), %iii)Average Relative Variance (ARV), iv)Phi Statistic (phi)

%Produced in          :09.08.2009
%Last Modified in    :08.06.2010                    ***written by Mert Kompil    ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load -mat HBW_TRAIN_DATA;     %load data from a -mat file (-mat file includes
FRICTION_TRAIN,OBS_TRIPS_TRAIN and ROW_DATA_TRAIN matrices)

%% Select Cost Function Type (Power or Exponential)
answer        = input(' Please Select The Cost Function :\n Enter "1" for POWER and "2" for EXPONENTIAL
FUNCTIONS  ----- > ');
if  answer     ==1;
   type=1;
else
   type=2;
end;

%% Rename Data Matrices
t_time=HBW_FRICTION_TRAIN;              %enter the name of friction matrix here (distance/time)(cij)
obs_trips=HBW_OBS_TRAIN;                %enter the name of trip matrix (Tij)
row_data=HBW_ROW_DATA_TRAIN;            %enter the name of row data matrix
                                        %here the row_data includes TAZ IDs in column 1-2, frictions in
                                        %column 5 and interactions in column 6
clear ('HBW_FRICTION_TRAIN','HBW_OBS_TRAIN', 'HBW_ROW_DATA_TRAIN','answer')
%Clear old matrixes

%% Enter Maximum Iteration Number and Convergence Criteria for Balancing Factors
iter=100;                       %maximum iteration number for the loop finding the balancing factors (Ai&Bj)
conv=0.001;                     %convergence criteria for balancing factors (Ai&Bj)

%% Set Pre-Calibrated beta (b) parameter or parameters
b=[-1.94 -2.05 -1.84]';         %optimum parameter estimates (MLH, WLS, TLD RMSE)for power cost function
%b=[-0.12 -0.21 -0.31]';        %optimum parameter estimates (MLH, WLS, TLD RMSE)for exponential cost function
%b=(-0.05:-0.05:-4)';           %For more than one parameter, the input should be a column vector
[w,c]=size(b);                  %matrix index

%% Create Starting matrices and parameters
prod_tot=sum(obs_trips,2)';             %produce production totals matrix
attr_tot=sum(obs_trips);                %produce attraction totals matrix
tot_trips=sum(sum(obs_trips));          %compute total trips
srmse=ones(w,1);                        %create a column vector for SRMSE Results
arv=ones(w,1);                          %create a column vector for ARAE Results
phi=ones(w,1);                          %create a column vector for Phi Statistic Results
nrmse=ones(w,1);                        %create a column vector for NRMSE Results
intercept=ones(w,1);                    %create a column vector for Intercept value Results
slope=ones(w,1);                        %create a column vector for Slope value Results
rsquare=ones(w,1);                      %create a column vector for R square Results
```

```
chisquare=ones(w,1);                    %create a column vector for Chi Square Results
[m,n]=size(obs_trips);                  %matrix index
[h,s]=size(row_data);                   %matrix index
mod_trips=ones(m,n);                    %create a matrix for modeled trip matrix
result_matrix=row_data;                 %create a matrix to write modeled trips in rows
A0=ones(m,n);                           %computation table for Ais
B0=ones(m,n);                           %computation table for Bjs
Ai=ones(m,iter);                        %produce a matrix for Ais to make comparison through iterations
Bj=ones(iter,n);                        %produce a matrix for Bjs to make comparison through iterations
B=ones(1,n);                            %set the initial Bjs as one

%% Start Computation for each beta (b) input
for g=1:w       %START LOOP 1: produce modeled trip matrix for given beta (b)

%% Find Balancing Factors (Ais&Bjs)
    for k=1:iter        %START LOOP 2: for balancing factors (Ai&Bj)
        if type==1                  %if the Power Cost function is selected
          for i=1:m,                %find Ais
            for j=1:n
              A0(i,j)=B(j)*attr_tot(j)*(t_time(i,j)^b(g));
            end;
          end;
            A=(1./sum(A0,2))';
          for i=1:m                 %find Bjs
            for j=1:n
              B0(i,j)=A(i)*prod_tot(i)*(t_time(i,j)^b(g));
            end;
          end;
        end;
        if type==2                  %if the Exponential Cost function is selected
          for i=1:m,                %find Ais
            for j=1:n
              A0(i,j)=B(j)*attr_tot(j)*(exp(t_time(i,j)*b(g)));
            end;
          end;
            A=(1./sum(A0,2))';
          for i=1:m                 %find Bjs
            for j=1:n
              B0(i,j)=A(i)*prod_tot(i)*(exp(t_time(i,j)*b(g)));
            end;
          end;
        end;
        B=(1./sum(B0));
        Ai(:,k)=A;                  %write Ais to the related column in matrix for comparison
        Bj(k,:)=B;                  %write Bjs to the related row in matrix for comparison
        if k>1                      %compare balancing factors
          if sqrt((sum((Ai(:,k)-Ai(:,(k-1))).^2))/k)< conv && sqrt(sum((Bj(k,:)-Bj((k-1),:).^2),2)/k) < conv;
          break                     %if convergence criteria is satisfied, stop process, otherwise continue
          end;
        end;
    end;            %END OF LOOP 2: for balancing factors (Ai&Bj)

%% Compute Modeled Trip Matrix
    if type==1                      %if the Power Cost function is selected
      for i=1:m,
        for j=1:n
          mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(t_time(i,j)^b(g));       %compute the modeled trips
        end;
      end;
    end;
    if type==2                      %if the Exponential Cost function is selected
      for i=1:m,
        for j=1:n
          mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(exp(t_time(i,j)*b(g)));    %compute the modeled trips
        end;
      end;
    end;
```

```
 %% Write Modeled Trip Matrix to Rows for Statistical Computation
v=1;
p=s+g;
for o=1:m
   temp=mod_trips(:,v);
   result_matrix(o,p)=temp(o);
end;
v=v+1;
t=1;
x=0;
for u=1:n-1
   for f=t:m*u
   temp=mod_trips(:,v);
   result_matrix(m+f,p)=temp(x+f);
   end;
   t=t+m;
   v=v+1;
   x=x-m;
end;

%% Calculate Goodness of Fit Statistics
%SRMSE      (Standardized Root Mean Squared Error)
srmse(g)=(sqrt((sum((result_matrix(:,6)-result_matrix(:,p)).^2)./(m*n))))/(sum(result_matrix(:,6))/(m*n));
%Straight Line Statistics (Intercept, Slope and R square)
stats={'beta','rsquare',};
obs=result_matrix(:,6);
mod=result_matrix(:,p);
str_line = regstats(obs,mod,'linear',stats);
rsquare(g) = str_line.rsquare(1,1);
intercept (g)=str_line.beta(1,1);
slope(g)=str_line.beta(2,1);
%ARV (Average Relative Variance)
var_full=39.0287;                                   %variance of full data
%var_train= var(result_matrix(:,6));                %variance of train data
arv(g)= (sum((result_matrix(:,6)-result_matrix(:,p)).^2))/(var_full*m*n);
% Phi Statistic
phi_trip_0=(result_matrix(:,6)+0.000001);   %add a small constant before computation of phi statistic
phi_trip=phi_trip_0./tot_trips;
phi (g)= sum(phi_trip.*(abs(log(phi_trip_0./result_matrix(:,p)))));
%Chi Square     (Pearson Chi Square Statistic)
%chisquare(g)=nansum(((result_matrix(:,6)-result_matrix(:,p)).^2)./(result_matrix(:,p)));
%NRMSE          (Normalized Root Mean Squared Error)
%nrmse(g)=(sqrt((sum((result_matrix(:,6)-result_matrix(:,p)).^2)./(m*n))))/(max(max(obs_trips))-
min(min(obs_trips))));
end;              %END OF LOOP 2: for each beta (b) parameter

%% Show the Results in Workplace
type_1_results_structure={'beta', 'SRMSE','r square','Slope','ARV', 'phi',;...
b, srmse, rsquare, slope, arv,  phi};
type_1_results=[b srmse rsquare slope arv phi];
fprintf('\n Goodness of Fit Statistics for Doubly Constraint SIM has produced\n')
clear   ('A','B','attr_tot','prod_tot','conv','type','i','j','g','w','r','m','n','iter','k','i','srmse','nrmse','rsquare','phi_trip','phi',...

'obs_mtc','mod_mtc','t_time','mod_trips','obs_trips','row_data','u','v','x','t','temp','b','intercept','slope','arv','phi_trip_0',...
      'str_line','stats','obs','mod','f','h','o','p','s','c','result_matrix','A0','Ai','B0','Bj','chisquare','var_full','tot_trips')
%clear temporary variables

%% Output of the Program
%The outputs are combined into a result matrix and also into a structure.
%In an order of columns, the result matrix includes
%i)     beta (b) values used as an input
%ii)    Standardized Root Mean Square Error (SRMSE)
%ii)    Regression Statistics (R square,Slope,Intercept),
%iii)   Average Relative Variance (ARV)
%iv)    Phi Statistic (phi)
%v)     Mean Travel Cost Error (MTCE)
```

A.6. Goodness-Of-Fit Statistics - Macro Level

```
---------------------------------------------------------------------------------------------------------------------------------
%% GOODNESS OF FIT STATISTICS FOR DOUBLY CONSTRAINT GRAVITY MODEL - MACRO LEVEL
% For MTCE, TLD RMSE, TLD ARAE F5, TLD ARAE L5

%% Definition of the Program
%This program uses previously estimated beta (b)parameter as an input to produce modeled trip matrix
%Then produces goodness of fit statistics comparing observed and modeled Trip Length Distribution (TLD)
%The code can be used for both of the Cost Functions: Power (cij^-b) and Exponential(e^-b*cij)
%During the process, you have to set an interval number (bin interval) and a maximum value (end of bins)
%At the end, the code produces i) Mean Travel Cost Error, ii) Trip Length Distribution Root Mean Square Error
(TLD RMSE),
%iii)Trip Length Distribution Average Relative Absolute Error for the first five intervals (TLD ARAE F5)
%iv)Trip Length Distribution Mean Absolute Percentage Error for the last five intervals (TLD ARAE L5)

%Produced in          :15.08.2009
%Last Modified in  :06.08.2010                    ***written by Mert Kompil     ***mertkompil@gmail.com

%% Load Data
clc;
clear;
load -mat HBW_Train_Data;      %load data from a -mat file (-mat file includes
FRICTION_TRAIN,OBS_TRIPS_TRAIN and ROW_DATA_TRAIN matrices)

%% Select Cost Function Type (Power or Exponential)
answer         = input(' Please Select The Cost Function :\n Enter "1" for POWER and "2" for EXPONENTIAL
FUNCTIONS  ----- > ');
if  answer      ==1;
   type=1;
else
   type=2;
end;

%% Set an Interval Number for Each Bins
answer1         = input('\n Please Enter an Interval Number for Each Bins  ----- > ');

%% Set a Value That The Bins End
answer2         = input('\n Please Enter a Value that the Bins End  ----- > ');

%% Rename Data Matrices
t_time=HBW_FRICTION_TRAIN;                %enter the name of friction matrix here (distance/time)(cij)
obs_trips=HBW_OBS_TRAIN;                  %enter the name of trip matrix (Tij)
row_data=HBW_ROW_DATA_TRAIN;              %enter the name of row data matrix
                                          %here the row_data includes TAZ IDs in column 1-2, frictions in
                                          %column 5 and interactions in column 6

clear ('HBW_FRICTION_TRAIN','HBW_OBS_TRAIN', 'HBW_ROW_DATA_TRAIN','answer')
%Clear old matrixes

%% Enter Maximum Iteration Number and Convergence Criteria for Balancing Factors
iter=100;                         %maximum iteration number for the loop finding the balancing factors (Ai&Bj)
conv=0.001;                       %convergence criteria for balancing factors (Ai&Bj)

%% Set Pre-Calibrated beta (b) parameter or parameters
b=[-1.94 -2.05 -1.84]';           %optimum parameter estimates (MLH, WLS, TLD RMSE)for power cost function
%b=[-0.12 -0.21 -0.31]';          %optimum parameter estimates (MLH, WLS, TLD RMSE)for exponential cost function
%b=(-0.05:-0.05:-4)';             %For more than one parameter, the input should be a column vector
[w,c]=size(b);                            %matrix index

%% Compute Observed Trip Length Distribution
maxi=max(max(t_time));                    %the maximum friction value
mini=0;                                   %the minimum friction value - set zero as default value
bin_int=answer1;                          %interval for each bins
```

```
max_bin=answer2;                    %the value that bins end
num_bins=floor((max_bin/bin_int)+1);                %number of bins
OTLD=zeros(num_bins,4);                             %create a matrix for observed trip length distribution
OTLD(:,1)=(1:1:num_bins);
OTLD(:,2)=(bin_int:bin_int:(max_bin+bin_int))';
OTLD(num_bins,2)=maxi;
[h,s]=size(row_data);               %matrix index
for i=1:h                           %produce observed trip length distribution
   for k=1:num_bins-1
      if mini <= row_data(i,5) && row_data(i,5)< OTLD(1,2)
        OTLD(1,3)= OTLD(1,3)+row_data(i,6);
        break
      end;
      if OTLD(k,2) <= row_data(i,5) && row_data(i,5)< OTLD(k+1,2)
        OTLD(k+1,3)= OTLD(k+1,3)+row_data(i,6);
        break
      end;
   end;
end;
clear ('answer1','answer2','k')             %Clear temporary values

%% Create Starting matrices and parameters for modeled trips
prod_tot=sum(obs_trips,2)';                 %produce production totals matrix
attr_tot=sum(obs_trips);                    %produce attraction totals matrix
tot_trips=sum(sum(obs_trips));              %Sum observed trips
obs_mtc=sum(sum(obs_trips.*t_time))/tot_trips;          %compute Observed Mean Travel Cost
MTCE=ones(w,1);                                         %create a column vector for MTCE Results
TLD_RMSE=ones(w,1);                 %create a column vector for TLD RMSE Results
TLD_ARAE_F5=ones(w,1);              %create a column vector for TLD ARAE Results for first 5 bins
TLD_ARAE_L5=ones(w,1);              %create a column vector for TLD ARAE Results for last 5 bins
arae=ones(5,2);                     %create a temporary matrix for arae calculations
row_data_2=ones(h,c);               %create a matrix to write modeled tirps in rows
MTLD00=zeros(num_bins,w);           %create a matrix to write modeled Trip Length Distribution Counts
MTLD01=MTLD00;                      %create a matrix to write modeled TLD Percents
[m,n]=size(obs_trips);              %matrix index
mod_trips=ones(m,n);                %create a matrix for modeled trip matrix
A0=ones(m,n);                       %computation table for Ais
B0=ones(m,n);                       %computation table for Bjs
Ai=ones(m,iter);                    %produce a matrix for Ais to make comparison through iterations
Bj=ones(iter,n);                    %produce a matrix for Bjs to make comparison through iterations
B=ones(1,n);                        %set the initial Bjs as one

%% Start Computation for each beta (b) input
for g=1:w       %START LOOP 1: produce modeled trip matrix for given beta (b)

%% Find Balancing Factors (Ais&Bjs)
    for k=1:iter        %START LOOP 2: for balancing factors (Ai&Bj)
        if type==1                  %if the Power Cost function is selected
          for i=1:m,                %find Ais
            for j=1:n
              A0(i,j)=B(j)*attr_tot(j)*(t_time(i,j)^b(g));
            end;
          end;
            A=(1./sum(A0,2))';
          for i=1:m                 %find Bjs
            for j=1:n
              B0(i,j)=A(i)*prod_tot(i)*(t_time(i,j)^b(g));
            end;
          end;
        end;
        if type==2                  %if the Exponential Cost function is selected
          for i=1:m,                %find Ais
            for j=1:n
              A0(i,j)=B(j)*attr_tot(j)*(exp(t_time(i,j)*b(g)));
            end;
          end;
            A=(1./sum(A0,2))';
```

```
            for i=1:m                    %find Bjs
               for j=1:n
                  B0(i,j)=A(i)*prod_tot(i)*(exp(t_time(i,j)*b(g)));
               end;
            end;
         end;
         B=(1./sum(B0));
         Ai(:,k)=A;                      %write Ais to the related column in matrix for comparison
         Bj(k,:)=B;                      %write Bjs to the related row in matrix for comparison
         if k>1                          %compare balancing factors
            if sqrt((sum((Ai(:,k)-Ai(:,(k-1))).^2))/k)< conv && sqrt(sum((Bj(k,:)-Bj((k-1),:).^2),2)/k) < conv;
            break                        %if convergence criteria is satisfied, stop process, otherwise continue
            end;
         end;
      end;              %END OF LOOP 2: for balancing factors (Ai&Bj)

%% Compute Modeled Trip Matrix
      if type==1                         %if the Power Cost function is selected
         for i=1:m,
            for j=1:n
               mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(t_time(i,j)^b(g));          %compute the modeled trips
            end;
         end;
      end;
      if type==2                         %if the Exponential Cost function is selected
         for i=1:m,
            for j=1:n
               mod_trips(i,j)= A(i)*B(j)*prod_tot(i)*attr_tot(j)*(exp(t_time(i,j)*b(g)));      %compute the modeled trips
            end;
         end;
      end;

%% Write Modeled Trip Matrix to Rows to Compute Modeled TLD
      v=1;
      for o=1:m
         temp=mod_trips(:,v);
         row_data_2(o,g)=temp(o);
      end;
      v=v+1;
      t=1;
      x=0;
      for u=1:n-1
         for f=t:m*u
         temp=mod_trips(:,v);
         row_data_2(m+f,g)=temp(x+f);
         end;
         t=t+m;
         v=v+1;
         x=x-m;
      end;

%% Compute Modeled TLD
      for i=1:h
         for k=1:num_bins-1
            if mini <= row_data(i,5) && row_data(i,5)< OTLD(1,2)
               MTLD00(1,g)= MTLD00(1,g)+row_data_2(i,g);
               break
            end;
            if OTLD(k,2) <= row_data(i,5) && row_data(i,5)< OTLD(k+1,2)
               MTLD00(k+1,g)= MTLD00(k+1,g)+row_data_2(i,g);
               break
            end;
         end;
      end;

%% Compute MTCE  (Mean Travel Cost Error)
      mod_mtc=sum(sum(mod_trips.*t_time))/sum(sum(mod_trips));
```

```
      MTCE(g)=(obs_mtc-mod_mtc);

%% Compute TLD RMSE
      for i=1:num_bins
      OTLD(i,4)=(OTLD(i,3)/sum(OTLD(:,3)))*100;
      MTLD01(i,g)=(MTLD00(i,g)/sum(MTLD00(:,g)))*100;
      end;
      TLD_RMSE(g) =(sqrt((sum((OTLD(:,4)-MTLD01(:,g)).^2))))/num_bins;

%% Compute TLD ARAE for the First and the Last Five Length Intervals
      for i=1:5
         arae(i,1)=(abs(OTLD(i,4)-MTLD01(i,g))./OTLD(i,4));
         arae(i,2)=(abs(OTLD((num_bins-5)+i,4)-MTLD01((num_bins-5)+i,g))./OTLD((num_bins-5)+i,4));
      end;
      TLD_ARAE_F5(g) =(sum(arae(:,1))./5);
      TLD_ARAE_L5(g) =(sum(arae(:,2))./5);
end;          %END OF LOOP 1: for beta (b) line search

%% Write Results in a matrix
type_2_results_structure={'beta', 'MTCE','TLD_RMSE','TLD_ARAE_F5','TLD_ARAE_L5';...
b, MTCE, TLD_RMSE, TLD_ARAE_F5, TLD_ARAE_L5};
type_2_results=[b MTCE TLD_RMSE TLD_ARAE_F5 TLD_ARAE_L5];
%show parameter values, TLD RMSE and TLD ARAE values in same matrix

%% Show Results and Plots on Screen
if  type==1
   fprintf('\n Line Search ALgorithm for Power Cost Function is successfully completed. \n %g paremeters have tried
\n',w)
else
   fprintf('\n Line Search ALgorithm for Exponential Cost Function is successfully completed. \n %g paremeters have
tried \n',w)
end;
figure;
plot(type_2_results(:,2)); title('RMSE CHANGE')                    %plot RMSE change
figure;
plot(type_2_results(:,3)); title('TLD FIRST FIVE ARAE ')           %plot TLD FIRST FIVE ARAE

clear('A','A0','Ai','B','B0','Bj','MTLD00','MTLD01','OTLD','attr_tot','prod_tot','b','bin_int','num_bins','TLD_RMSE','to
t_trips',...
     'c', 'conv','f','g','h','i','index','iter','j','k','m','max_bin','maxi','mini','mod_trips','n','o','arae','MTCE', 'obs_mtc',...
     'x',
'v','u','w','type','temp','t_time','s','t','row_data_2','row_data','rmse','TLD_ARAE_F5','TLD_ARAE_L5','obs_trips','mod_
mtc');

%% Output of the Program
%  The output of the program is included in a result matrix. The columns contain orderly,
%  i)      The beta (b) parameters used as input
%  ii)     Mean Travel Cost Error (MTCE)
%  iii)    Trip Length Distribution Root Mean Squared Error (TLD RMSE)
%  iv)   Trip Length Distribution Mean Absolute Percentage Error for the first five Bins (TLD ARAE First Five)
%  v)    Trip Length Distribution Mean Absolute Percentage Error for the last five Bins (TLD ARAE Last Five)
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

A.7. Fuzzy Rule-Based System (FRBS) - Rule Learning Algorithm

```
---------------------------------------------------------------------------------------------------------------------------------------
%% FUZZY RULE-BASED SYSTEM (FRBS) - RULE LEARNING ALGORITHM

%% Definition of the Program
%This program includes a fuzzy rulebase learning algorithm for the trip distribution problem
%The code uses production, attraction, friction data vectors as inputs and trip interactions as output
%The code uses predefined fuzzy membership functions and establishes a rulebase from numerical data
%The learning procedure basis on a modififed 'Wang-Mendel'(1992)rule learning process
%The final rules are selected after computing the weighted averages of the conflicting rules
%At the end, the code produces i)Observed rules for each of the data pairs (obs_rules),
%ii)Counted and summarized rule candidates (summarized_rules),
%iii)Finally established rule base (final_rulebase).

%Produced in                :10.02.2010
%Last Modified in           :20.06.2010      ***written by Mert Kompil      ***mertkompil@gmail.com

%% Load Data
clc; clear;
load -mat HBW_TRAIN_DATA.mat;              %include data vectors
load -mat MF_VECTORS.mat;                  %include MF points in a structure named 'mfvec'
 %% Construct Data Vectors
row_data=HBW_ROW_DATA_TRAIN;               %HBW train data in the row format
input_data=[row_data(:,3) row_data(:,4) row_data(:,5) row_data(:,6)];
round_data=[input_data(:,1) input_data(:,2) roundn(input_data(:,3),-1) input_data(:,4)];
data=round_data;                           %vectors of Production, Attraction, Friction and Trips
[m,~]=size(data);
clear ('HBW_ROW_DATA_TRAIN','HBW_OBS_TRAIN','HBW_FRICTION_TRAIN',...
    'input_data','row_data','round_data') %clear temporary files
 %% Establish Variable Intervals - Membership function Domains
x_prod=0:1:1200;               %production variable domain
x_attr=0:1:2050;               %attraction variable domain
x_fric=0:0.1:300;              %friction variable domain
x_fric=roundn(x_fric,-1);      %round friction variable domain
x_trips=0:1:885;               %trips variable domain
 %% CONSTRUCT MEMBERSHIP DOMAINS %%
%% production MFs
prodmf_1=trapmf(x_prod,mfvec.prod(1:4));       %call MF points from MF_VECTORS.mat
prodmf_2=trimf(x_prod,mfvec.prod(5:7));
prodmf_3=trimf(x_prod,mfvec.prod(8:10));
prodmf_4=trimf(x_prod,mfvec.prod(11:13));
prodmf_5=trapmf(x_prod,mfvec.prod(14:17));
%production MFS - Antecedents
ant_prod=[prodmf_1;prodmf_2;prodmf_3;prodmf_4;prodmf_5];
%% attraction MFs
attrmf_1=trapmf(x_attr,mfvec.attr(1:4));       %call MF points from MF_VECTORS.mat
attrmf_2=trimf(x_attr,mfvec.attr(5:7));
attrmf_3=trimf(x_attr,mfvec.attr(8:10));
attrmf_4=trimf(x_attr,mfvec.attr(11:13));
attrmf_5=trapmf(x_attr,mfvec.attr(14:17));
%attraction MFS - Antecedents
ant_attr=[attrmf_1;attrmf_2;attrmf_3;attrmf_4;attrmf_5];
%% friction MFs
fricmf_1=trapmf(x_fric,mfvec.fric(1:4));       %call MF points from MF_VECTORS.mat
fricmf_2=trimf(x_fric,mfvec.fric(5:7));
fricmf_3=trimf(x_fric,mfvec.fric(8:10));
fricmf_4=trimf(x_fric,mfvec.fric(11:13));
fricmf_5=trimf(x_fric,mfvec.fric(14:16));
fricmf_6=trapmf(x_fric,mfvec.fric(17:20));
%friction MFS - Antecedents
ant_fric=[fricmf_1;fricmf_2;fricmf_3;fricmf_4;fricmf_5;fricmf_6];
%% trips MFs
tripsmf_1=trapmf(x_trips,mfvec.trips(1:4));    %call MF points from MF_VECTORS.mat
tripsmf_2=trimf(x_trips,mfvec.trips(5:7));
```

```
tripsmf_3=trimf(x_trips,mfvec.trips(8:10));
tripsmf_4=trimf(x_trips,mfvec.trips(11:13));
tripsmf_5=trimf(x_trips,mfvec.trips(14:16));
tripsmf_6=trimf(x_trips,mfvec.trips(17:19));
tripsmf_7=trimf(x_trips,mfvec.trips(20:22));
tripsmf_8=trimf(x_trips,mfvec.trips(23:25));
tripsmf_9=trimf(x_trips,mfvec.trips(26:28));
tripsmf_10=trimf(x_trips,mfvec.trips(29:31));
tripsmf_11=trimf(x_trips,mfvec.trips(32:34));
tripsmf_12=trimf(x_trips,mfvec.trips(35:37));
tripsmf_13=trimf(x_trips,mfvec.trips(38:40));
tripsmf_14=trimf(x_trips,mfvec.trips(41:43));
tripsmf_15=trimf(x_trips,mfvec.trips(44:46));
tripsmf_16=trimf(x_trips,mfvec.trips(47:49));
tripsmf_17=trimf(x_trips,mfvec.trips(50:52));
tripsmf_18=trimf(x_trips,mfvec.trips(53:55));
tripsmf_19=trimf(x_trips,mfvec.trips(56:58));
tripsmf_20=trapmf(x_trips,mfvec.trips(59:62));
%Trips MFS - Consequents
cons_trips=[tripsmf_1;tripsmf_2;tripsmf_3;tripsmf_4;tripsmf_5;tripsmf_6;tripsmf_7;tripsmf_8;
          tripsmf_9;tripsmf_10;tripsmf_11;tripsmf_12;tripsmf_13;tripsmf_14;tripsmf_15;
          tripsmf_16;tripsmf_17;tripsmf_18;tripsmf_19;tripsmf_20;];
clear('prodmf_1','prodmf_2','prodmf_3','prodmf_4','prodmf_5','attrmf_1','attrmf_2','attrmf_3',...
'attrmf_4','attrmf_5','fricmf_1','fricmf_2','fricmf_3','fricmf_4','fricmf_5','fricmf_6',...
'tripsmf_1','tripsmf_2','tripsmf_3','tripsmf_4','tripsmf_5','tripsmf_6','tripsmf_7','tripsmf_8',...
'tripsmf_9','tripsmf_10','tripsmf_11','tripsmf_12','tripsmf_13','tripsmf_14','tripsmf_15',...
'tripsmf_16','tripsmf_17','tripsmf_18','tripsmf_19','tripsmf_20',);       %clear temporary files

%% IDENTIFY MEMBERSHIP DEGREES FOR EACH DATA PAIRS %%
%% production labels
res_prod=ones(5,m);
[~,s]=size(x_prod);
for i=1:m
   k=data(i,1);
   for j=1:s
      if k==x_prod(j)
        ind=j;
        break
      end;
   end;
   res_prod(:,i)=ant_prod(:,ind);
end;
 %% attraction labels
res_attr=ones(5,m);
[~,s]=size(x_attr);
for i=1:m
   k=data(i,2);
   for j=1:s
      if k==x_attr(j)
        ind=j;
        break
      end;
   end;
   res_attr(:,i)=ant_attr(:,ind);
end;
 %% friction labels
res_fric=ones(6,m);
[~,s]=size(x_fric);
for i=1:m
   k=data(i,3);
   for j=1:s
      if k==x_fric(j)
        ind=j;
        break
      end;
   end;
   res_fric(:,i)=ant_fric(:,ind);
```

```
end;
 %% trips labels
res_trips=ones(20,m);
[~,s]=size(x_trips);
for i=1:m
   k=data(i,4);
   for j=1:s
      if k==x_trips(j)
        ind=j;
        break
      end;
   end;
   res_trips(:,i)=cons_trips(:,ind);
end;
clear ('x_attr','x_trips','x_prod','x_fric','h','s','k','i','j','ind') %clear temporary files
 %% ASSIGN MF LABELS HAVING MAXIMUM MEMBERSHIP DEGREE  %%
%% production labels
[h,s]=size(res_prod);
rules=zeros(s,9);
for i=1:s
   max_prod=max(res_prod(:,i));
   for j=1:h
      if  res_prod(j,i)==max_prod
         rules(i,1)=j;
         break
      end;
   end;
   rules(i,5)=max_prod;
end;
 %% attraction labels
[h,s]=size(res_attr);
for i=1:s
   max_attr=max(res_attr(:,i));
   for j=1:h
      if  res_attr(j,i)==max_attr
         rules(i,2)=j;
         break
      end;
   end;
   rules(i,6)=max_attr;
end;
%% friction labels
[h,s]=size(res_fric);
for i=1:s
   max_fric=max(res_fric(:,i));
   for j=1:h
      if  res_fric(j,i)==max_fric
         rules(i,3)=j;
         break
      end;
   end;
   rules(i,7)=max_fric;
end;
%% trips labels
[h,s]=size(res_trips);
for i=1:s
   max_trips=max(res_trips(:,i));
   for j=1:h
      if  res_trips(j,i)==max_trips
         rules(i,4)=j;
         break
      end;
   end;
   rules(i,8)=max_trips;
end;
%% IDENTIFY RULE CANDIDATES AND COMPUTE THEIR STRENGTHS FOR EACH OF THE DATA
PAIRS %%
```

```
rules(:,9)=rules(:,5).*rules(:,6).*rules(:,7).*rules(:,8); %Com
obs_rules=rules;                    % Observed Rule Candidates and Membership Degrees for Each Data Pairs
 clear ('max_attr','max_trips','max_prod','max_fric','res_attr','res_trips','res_prod','res_fric',...
'h','s','i','j','m','n','ant_attr','cons_trips','ant_prod','ant_fric','mfvec')                    %clear temporary files

%% COUNT AND SUMMARIZE RULE CANDIDATES
[m,~]=size(rules);
mf_prod=5; mf_attr=5; mf_fric=6; mf_trips=20; % Assign number of MFs
write=zeros(m,5);
t=1;
for i=1:mf_prod
   for r=1:mf_attr
      for j=1:mf_fric
         for h=1:mf_trips
            for k=1:m
               if  rules(k,1)==i && rules(k,2)==r && rules(k,3)==j && rules(k,4)==h
                  write(t,:)=[i r j h rules(k,9)];
                  t=t+1;
               end;
            end;
         end;
      end;
   end;
end;
write2=write(:,1:1:4);
write2(m+1,:)=write2(1,:);
t=1; p=1; g=1;
summarized_rules=zeros(1,5);
for i=p:m
   if  write2(i,:)-write2(i+1,:)==0
      g=g+1;
   else
      summarized_rules(t,:)=write(i,:);
      summarized_rules(t,5)=g;
      p=p+1;
      t=t+1;
      g=1;
   end;
end;
[v,~]=size(summarized_rules);
ssum=ones(v,2);
ssum(1,1)=summarized_rules(1,5);
k=2;
for i=1:v-1
   h=summarized_rules(i+1,5);
   ssum(k,1)=ssum(k-1,1)+h;
   k=k+1;
end;
[v,~]=size(ssum);
k=1;
for i=1:v
   s=ssum(i);
   ssum(i,2)=sum(write(k:s,5));
   k=s+1;
end
summarized_rules=[summarized_rules ssum(:,2)];
summarized_rules(:,6)=summarized_rules(:,6)./summarized_rules(:,5); % Counted and Summarized Rule Candidates
clear ('g','h','i','j','k','m','n','p','s','t','v','y','write','write2','ssum','r')          %clear temporary files
 %% COMPUTE WEIGHTED AVERAGE OF RULE CANDIDATES - ESTABLISH RULE BASE %%
srules=[summarized_rules summarized_rules(:,4).*summarized_rules(:,5)];
[m,n]=size(srules);
n_rules=mf_prod*mf_attr*mf_fric;
write=zeros(n_rules,7);
t=1;
for i=1:mf_prod
   for r=1:mf_attr
      for j=1:mf_fric
```

```
            for k=1:m
               if  srules(k,1)==i && srules(k,2)==r && srules(k,3)==j
                  write(t,1:4)=[i r j k];
               end;
            end;
            t=t+1;
         end;
      end;
end;
write2=write;
z=0; h=0;
for i=1:n_rules
   if write2(i,4)==0
     z=z+1;
   end;
end;
write3=zeros(n_rules-z,7);
for s=1:n_rules-z
   for i=1+h:n_rules
      if  write2(i,4)~=0
         h=i;
         write3(s,:)=write2(i,:);
         break
      end;
   end;
end;
write3(1,5)=write3(1,4);
for i=1:n_rules-z-1
   b=write3(i+1,4);
   c=write3(i,4);
   write3(i+1,5)=b-c;
end
k=0; y=0;
for i=1:n_rules-z
      p=write3(i,5);
   for j=k+1:p+k
      y=y+srules(j,7);
   end;
   write3(i,6)=y;
   y=0;
   k=k+p;
end;
k=0; y=0; p=0;
for i=1:n_rules-z
      p=write3(i,5);
   for j=k+1:p+k
      y=y+srules(j,5);
   end;
   write3(i,7)=y;
   y=0;
   k=k+p;
end;
z=write3(:,6)./write3(:,7);
z=round(z);
final_rulebase=[write3(:,1:3) z];          %Establish Final Rule Base

clear ('m','n','i','j','p','k','y','t','write','write2','write3','c','b','h','mf_attr','mf_prod','mf_fric',...
      'r','s','z','mf_trips','n_rules','rules','srules','data') %clear temporary files

%% OUTPUT OF THE PROGRAM %%
% The present code produces three outputs:
% i)Observed rules for each of the data pairs => obs_rules,
% ii)Counted and summarized rule candidates => summarized_rules,
% iii)Finally established rule base (before experts control) => final_rulebase.
```

A.8. Fuzzy Rule-Based System (FRBS) - Implication Algorithm

```
---------------------------------------------------------------------------------------------------------------------------------------
%% FUZZY RULE-BASED SYSTEM (FRBS) - IMPLICATION ALGORITHM

%% Definition of the Program
%This program is a fuzzy implication of the trip distribution problem
%The code uses production, attraction, friction data vectors as inputs
%And estimates trip interactions as output
%The code uses the predefined fuzzy membership functions and rulebase
%The implication and the defuzzification techniques are: the "max-product" and the "centroid" techniques
%At the end, the code produces i) Fuzzy System Output (fuzzy_output),
%ii) Modelled and Balanced Trip Matrix (Modelled_trips), %iii)Mean Squared Error (MSE),
%iv)Standardized Root Mean Squared Error (SRMSE)

%Produced in              :15.04.2010
%Last Modified in         :20.06.2010      ***written by Mert Kompil     ***mertkompil@gmail.com

%% Load Data
clc;
clear all;
load -mat HBW_TRAIN_DATA.mat;            %include data vectors
load -mat MF_VECTORS.mat;                %include MF points in a structure named 'mfvec'
load -mat RULEBASE.mat                   %include rulebase (150*1)

%% Construct Data Vectors
row_data=HBW_ROW_DATA_TRAIN;        %HBW train data in the row format
obs_trips=HBW_OBS_TRAIN;             %Observed trips in matrix format
ttime=HBW_FRICTION_TRAIN;            %Friction in matrix format
obs_input=row_data(:,3:5);
obs_output=row_data(:,6);
round_input=[obs_input(:,1) obs_input(:,2) roundn(obs_input(:,3),-1)];
in1=round_input(:,1); in2=round_input(:,2); in3=round_input(:,3);          % Input vectors
[m,~]=size(row_data);
clear ('HBW_ROW_DATA_TRAIN','HBW_FRICTION_TRAIN','HBW_OBS_TRAIN')      %clear temporary files

%% Establish Variable Intervals - Membership function Domains
x_prod=0:1:1200; x_attr=0:1:2050; x_fric=0:0.1:300; x_fric=roundn(x_fric,-1); %x_trips=0:1:885;
x_trips0=0:0.005:0.2; x_trips1=1:1:39; x_trips2=40:5:140; x_trips3=145:10:235; x_trips4=245:20:885;
x_trips=[x_trips0 x_trips1 x_trips2 x_trips3 x_trips4];
clear ('x_trips0','x_trips1','x_trips2','x_trips3','x_trips4')          %clear temporary files

%% CONSTRUCT MEMBERSHIP DOMAINS %%
%% production MFs
prodmf_1=trapmf(x_prod,mfvec.prod(1:4));      %call MF points from MF_VECTORS.mat
prodmf_2=trimf(x_prod,mfvec.prod(5:7));
prodmf_3=trimf(x_prod,mfvec.prod(8:10));
prodmf_4=trimf(x_prod,mfvec.prod(11:13));
prodmf_5=trapmf(x_prod,mfvec.prod(14:17));
%production MFS - Antecedents
ant_prod=[prodmf_1;prodmf_2;prodmf_3;prodmf_4;prodmf_5]';
%% attraction MFs
attrmf_1=trapmf(x_attr,mfvec.attr(1:4));      %call MF points from MF_VECTORS.mat
attrmf_2=trimf(x_attr,mfvec.attr(5:7));
attrmf_3=trimf(x_attr,mfvec.attr(8:10));
attrmf_4=trimf(x_attr,mfvec.attr(11:13));
attrmf_5=trapmf(x_attr,mfvec.attr(14:17));
%attraction MFS - Antecedents
ant_attr=[attrmf_1;attrmf_2;attrmf_3;attrmf_4;attrmf_5]';
%% friction MFs
fricmf_1=trapmf(x_fric,mfvec.fric(1:4));      %call MF points from MF_VECTORS.mat
fricmf_2=trimf(x_fric,mfvec.fric(5:7));
fricmf_3=trimf(x_fric,mfvec.fric(8:10));
fricmf_4=trimf(x_fric,mfvec.fric(11:13));
```

```
fricmf_5=trimf(x_fric,mfvec.fric(14:16));
fricmf_6=trapmf(x_fric,mfvec.fric(17:20));
%friction MFS - Antecedents
ant_fric=[fricmf_1;fricmf_2;fricmf_3;fricmf_4;fricmf_5;fricmf_6]';
%% trips MFs
tripsmf_1=trapmf(x_trips,mfvec.trips(1:4));        %call MF points from MF_VECTORS.mat
tripsmf_2=trimf(x_trips,mfvec.trips(5:7));
tripsmf_3=trimf(x_trips,mfvec.trips(8:10));
tripsmf_4=trimf(x_trips,mfvec.trips(11:13));
tripsmf_5=trimf(x_trips,mfvec.trips(14:16));
tripsmf_6=trimf(x_trips,mfvec.trips(17:19));
tripsmf_7=trimf(x_trips,mfvec.trips(20:22));
tripsmf_8=trimf(x_trips,mfvec.trips(23:25));
tripsmf_9=trimf(x_trips,mfvec.trips(26:28));
tripsmf_10=trimf(x_trips,mfvec.trips(29:31));
tripsmf_11=trimf(x_trips,mfvec.trips(32:34));
tripsmf_12=trimf(x_trips,mfvec.trips(35:37));
tripsmf_13=trimf(x_trips,mfvec.trips(38:40));
tripsmf_14=trimf(x_trips,mfvec.trips(41:43));
tripsmf_15=trimf(x_trips,mfvec.trips(44:46));
tripsmf_16=trimf(x_trips,mfvec.trips(47:49));
tripsmf_17=trimf(x_trips,mfvec.trips(50:52));
tripsmf_18=trimf(x_trips,mfvec.trips(53:55));
tripsmf_19=trimf(x_trips,mfvec.trips(56:58));
tripsmf_20=trapmf(x_trips,mfvec.trips(59:62));
cons_trips=[tripsmf_1;tripsmf_2;tripsmf_3;tripsmf_4;tripsmf_5;tripsmf_6;tripsmf_7;
        tripsmf_8;tripsmf_9;tripsmf_10;tripsmf_11;tripsmf_12;tripsmf_13;tripsmf_14;
        tripsmf_15;tripsmf_16;tripsmf_17;tripsmf_18;tripsmf_19;tripsmf_20];

clear ('prodmf_1','prodmf_2','prodmf_3','prodmf_4','prodmf_5','attrmf_1','attrmf_2','attrmf_3',...
'attrmf_4','attrmf_5','fricmf_1','fricmf_2','fricmf_3','fricmf_4','fricmf_5','fricmf_6',...
'tripsmf_1','tripsmf_2','tripsmf_3','tripsmf_4','tripsmf_5','tripsmf_6','tripsmf_7','tripsmf_8',...
'tripsmf_9','tripsmf_10','tripsmf_11','tripsmf_12','tripsmf_13','tripsmf_14','tripsmf_15',...
'tripsmf_16','tripsmf_17','tripsmf_18','tripsmf_19','tripsmf_20');        %clear temporary files

%% COMPUTE ANTECEDENTS OF INPUT PAIRS WITH MAX-PRODUCT IMPLICATION %%
antecedents=zeros(150,m);
parfor i=1:m
mf_p=interp1q(x_prod,ant_prod,in1(i));            %Identify MF labels for production variable
mf_a=interp1q(x_attr,ant_attr,in2(i));            %Identify MF labels for attraction variable
mf_f=interp1q(x_fric,ant_fric,in3(i));            %Identify MF labels for friction variable
antecedents(:,i)=[...
(mf_p(1).*mf_a(1).*mf_f(1));(mf_p(1).*mf_a(2).*mf_f(1));(mf_p(1).*mf_a(3).*mf_f(1));
(mf_p(1).*mf_a(4).*mf_f(1));(mf_p(1).*mf_a(5).*mf_f(1));(mf_p(2).*mf_a(1).*mf_f(1));
(mf_p(2).*mf_a(2).*mf_f(1));(mf_p(2).*mf_a(3).*mf_f(1));(mf_p(2).*mf_a(4).*mf_f(1));
(mf_p(2).*mf_a(5).*mf_f(1));(mf_p(3).*mf_a(1).*mf_f(1));(mf_p(3).*mf_a(2).*mf_f(1));
(mf_p(3).*mf_a(3).*mf_f(1));(mf_p(3).*mf_a(4).*mf_f(1));(mf_p(3).*mf_a(5).*mf_f(1));
(mf_p(4).*mf_a(1).*mf_f(1));(mf_p(4).*mf_a(2).*mf_f(1));(mf_p(4).*mf_a(3).*mf_f(1));
(mf_p(4).*mf_a(4).*mf_f(1));(mf_p(4).*mf_a(5).*mf_f(1));(mf_p(5).*mf_a(1).*mf_f(1));
(mf_p(5).*mf_a(2).*mf_f(1));(mf_p(5).*mf_a(3).*mf_f(1));(mf_p(5).*mf_a(4).*mf_f(1));
(mf_p(5).*mf_a(5).*mf_f(1));(mf_p(1).*mf_a(1).*mf_f(2));(mf_p(1).*mf_a(2).*mf_f(2));
(mf_p(1).*mf_a(3).*mf_f(2));(mf_p(1).*mf_a(4).*mf_f(2));(mf_p(1).*mf_a(5).*mf_f(2));
(mf_p(2).*mf_a(1).*mf_f(2));(mf_p(2).*mf_a(2).*mf_f(2));(mf_p(2).*mf_a(3).*mf_f(2));
(mf_p(2).*mf_a(4).*mf_f(2));(mf_p(2).*mf_a(5).*mf_f(2));(mf_p(3).*mf_a(1).*mf_f(2));
(mf_p(3).*mf_a(2).*mf_f(2));(mf_p(3).*mf_a(3).*mf_f(2));(mf_p(3).*mf_a(4).*mf_f(2));
(mf_p(3).*mf_a(5).*mf_f(2));(mf_p(4).*mf_a(1).*mf_f(2));(mf_p(4).*mf_a(2).*mf_f(2));
(mf_p(4).*mf_a(3).*mf_f(2));(mf_p(4).*mf_a(4).*mf_f(2));(mf_p(4).*mf_a(5).*mf_f(2));
(mf_p(5).*mf_a(1).*mf_f(2));(mf_p(5).*mf_a(2).*mf_f(2));(mf_p(5).*mf_a(3).*mf_f(2));
(mf_p(5).*mf_a(4).*mf_f(2));(mf_p(5).*mf_a(5).*mf_f(2));(mf_p(1).*mf_a(1).*mf_f(3));
(mf_p(1).*mf_a(2).*mf_f(3));(mf_p(1).*mf_a(3).*mf_f(3));(mf_p(1).*mf_a(4).*mf_f(3));
(mf_p(1).*mf_a(5).*mf_f(3));(mf_p(2).*mf_a(1).*mf_f(3));(mf_p(2).*mf_a(2).*mf_f(3));
(mf_p(2).*mf_a(3).*mf_f(3));(mf_p(2).*mf_a(4).*mf_f(3));(mf_p(2).*mf_a(5).*mf_f(3));
(mf_p(3).*mf_a(1).*mf_f(3));(mf_p(3).*mf_a(2).*mf_f(3));(mf_p(3).*mf_a(3).*mf_f(3))
(mf_p(3).*mf_a(4).*mf_f(3));(mf_p(3).*mf_a(5).*mf_f(3));(mf_p(4).*mf_a(1).*mf_f(3))
(mf_p(4).*mf_a(2).*mf_f(3));(mf_p(4).*mf_a(3).*mf_f(3));(mf_p(4).*mf_a(4).*mf_f(3));
(mf_p(4).*mf_a(5).*mf_f(3));(mf_p(5).*mf_a(1).*mf_f(3));(mf_p(5).*mf_a(2).*mf_f(3));
(mf_p(5).*mf_a(3).*mf_f(3));(mf_p(5).*mf_a(4).*mf_f(3));(mf_p(5).*mf_a(5).*mf_f(3));
```

```
(mf_p(1).*mf_a(1).*mf_f(4));(mf_p(1).*mf_a(2).*mf_f(4));(mf_p(1).*mf_a(3).*mf_f(4));
(mf_p(1).*mf_a(4).*mf_f(4));(mf_p(1).*mf_a(5).*mf_f(4));(mf_p(2).*mf_a(1).*mf_f(4));
(mf_p(2).*mf_a(2).*mf_f(4));(mf_p(2).*mf_a(3).*mf_f(4));(mf_p(2).*mf_a(4).*mf_f(4));
(mf_p(2).*mf_a(5).*mf_f(4));(mf_p(3).*mf_a(1).*mf_f(4));(mf_p(3).*mf_a(2).*mf_f(4));
(mf_p(3).*mf_a(3).*mf_f(4));(mf_p(3).*mf_a(4).*mf_f(4));(mf_p(3).*mf_a(5).*mf_f(4));
(mf_p(4).*mf_a(1).*mf_f(4));(mf_p(4).*mf_a(2).*mf_f(4));(mf_p(4).*mf_a(3).*mf_f(4));
(mf_p(4).*mf_a(4).*mf_f(4));(mf_p(4).*mf_a(5).*mf_f(4));(mf_p(5).*mf_a(1).*mf_f(4));
(mf_p(5).*mf_a(2).*mf_f(4));(mf_p(5).*mf_a(3).*mf_f(4));(mf_p(5).*mf_a(4).*mf_f(4));
(mf_p(5).*mf_a(5).*mf_f(4));(mf_p(1).*mf_a(1).*mf_f(5));(mf_p(1).*mf_a(2).*mf_f(5));
(mf_p(1).*mf_a(3).*mf_f(5));(mf_p(1).*mf_a(4).*mf_f(5));(mf_p(1).*mf_a(5).*mf_f(5));
(mf_p(2).*mf_a(1).*mf_f(5));(mf_p(2).*mf_a(2).*mf_f(5));(mf_p(2).*mf_a(3).*mf_f(5));
(mf_p(2).*mf_a(4).*mf_f(5));(mf_p(2).*mf_a(5).*mf_f(5));(mf_p(3).*mf_a(1).*mf_f(5));
(mf_p(3).*mf_a(2).*mf_f(5));(mf_p(3).*mf_a(3).*mf_f(5));(mf_p(3).*mf_a(4).*mf_f(5));
(mf_p(3).*mf_a(5).*mf_f(5));(mf_p(4).*mf_a(1).*mf_f(5));(mf_p(4).*mf_a(2).*mf_f(5));
(mf_p(4).*mf_a(3).*mf_f(5));(mf_p(4).*mf_a(4).*mf_f(5));(mf_p(4).*mf_a(5).*mf_f(5));
(mf_p(5).*mf_a(1).*mf_f(5));(mf_p(5).*mf_a(2).*mf_f(5));(mf_p(5).*mf_a(3).*mf_f(5));
(mf_p(5).*mf_a(4).*mf_f(5));(mf_p(5).*mf_a(5).*mf_f(5));(mf_p(1).*mf_a(1).*mf_f(6));
(mf_p(1).*mf_a(2).*mf_f(6));(mf_p(1).*mf_a(3).*mf_f(6));(mf_p(1).*mf_a(4).*mf_f(6));
(mf_p(1).*mf_a(5).*mf_f(6));(mf_p(2).*mf_a(1).*mf_f(6));(mf_p(2).*mf_a(2).*mf_f(6));
(mf_p(2).*mf_a(3).*mf_f(6));(mf_p(2).*mf_a(4).*mf_f(6));(mf_p(2).*mf_a(5).*mf_f(6));
(mf_p(3).*mf_a(1).*mf_f(6));(mf_p(3).*mf_a(2).*mf_f(6));(mf_p(3).*mf_a(3).*mf_f(6));
(mf_p(3).*mf_a(4).*mf_f(6));(mf_p(3).*mf_a(5).*mf_f(6));(mf_p(4).*mf_a(1).*mf_f(6));
(mf_p(4).*mf_a(2).*mf_f(6));(mf_p(4).*mf_a(3).*mf_f(6));(mf_p(4).*mf_a(4).*mf_f(6));
(mf_p(4).*mf_a(5).*mf_f(6));(mf_p(5).*mf_a(1).*mf_f(6));(mf_p(5).*mf_a(2).*mf_f(6));
(mf_p(5).*mf_a(3).*mf_f(6));(mf_p(5).*mf_a(4).*mf_f(6));(mf_p(5).*mf_a(5).*mf_f(6))];
end;

%% AGGREGATE WITH CONSEQUENTS AND DEFUZZIFY USING CENTROID DEFUZZIFICATION %%
rules=rulebase;
[y,z]=size(x_trips);
[nrules,g]=size(rulebase);
fuzzy_output=zeros(m,1);
consequent=zeros(nrules,z);
cons0=zeros(nrules,z);
for i=1:nrules
cons0(i,:)=cons_trips(rules(i),:);
end;
for i=1:m
   for j=1:z
   consequent(:,j)=(cons0(:,j).*antecedents(:,i));
   end;
aggregation= max(consequent);                    %Aggregation
output0= defuzz(x_trips,aggregation,'centroid'); %Defuzzifization
fuzzy_output(i)=output0;                         %Fuzzy Output (Modelled trips before balancing)
end;
clear ('x_attr','x_prod','x_fric','x_trips','rules','rulebase','y','z','output0',...
'm','n','nrules','mfvec','i','j','g','ant_attr','ant_fric','ant_prod','cons0','cons_trips',...
'consequent','antecedents','aggregation','in1','in2','in3','round_input')

%% APPLY BALANCING WITH FURNESS ITERATION AND PRODUCE MODELLED MATRIX %%
mod=fuzzy_output;
obsmat=obs_trips;
[m,n]=size(obsmat);
modmat=zeros(m,n);
%% Convert Row Vector to Matrix
t=1;
c=m;
for i=1:n
   modmat(:,i)=mod((t:1:c)',1);
   t=t+m;
   c=c+m;
end;
%% Balance Matrix with Iterations
iter=100;
prodtot=sum(obs_trips,2)';
attrtot=sum(obs_trips);
attr_mod=sum(modmat);
```

```
diff_1=ones(iter,n); diff_2=ones(m,iter);
for d=1:iter
   diff_1(d,:)=attrtot./attr_mod;
   for i=1:m
      for j=1:n
      modmat(i,j)=diff_1(d,j).*modmat(i,j);
      end;
   end;
   prod_mod=sum(modmat,2)';
   diff_2(:,d)=prodtot./prod_mod;
   for i=1:m
      for j=1:n
      modmat(i,j)=diff_2(i,d).*modmat(i,j);
      end;
   end;
   attr_mod=sum(modmat);
end;
modelled_trips=modmat;                    %Modelled Trip Matrix

%% CALCULATE MSE and SRMSE
MSE=mse(obsmat-modmat);     %Mean Squared error
SRMSE=(sqrt((sum(sum((modmat-obsmat).^2)))./(m*n)))/(sum(sum(obsmat))/(m*n));   %Std. Root Mean Squ. Error

clear ('attr_mod','attrtot','prodtot','c','d','i','iter','j','m','n','mod','diff_1','diff_2',...
   'modmat','obs_input','obs_output','obs_trips','obsmat','prod_mod','row_data','t','ttime')

%% OUTPUT OF THE PROGRAM %%
% The present code produces three outputs:
% i)Fuzzy System Output =>fuzzy_output,
% ii)Modelled Trip Matrix After Balancing =>Modelled_trips,
% iii)Mean Squared Error =>MSE,
% iv)Standardized Root Mean Squared Error =>SRMSE.
-------------------------------------------------------------------------------------------------------------------------------------------------------
```

A.9. Genetic Fuzzy Rule-Based System (GFRBS) - Training Algorithm

```
--------------------------------------------------------------------------------------------------------------------------------
%% GENETIC FUZZY RULE-BASED SYSTEM (GFRBS) - TRAINING ALGORITHM

%% Definition of the Program
%This program learns rulebase of a fuzzy system for the trip distribution problem with GAs
%The code uses production, attraction, friction data vectors as inputs and trip interactions as output
%The GA code uses predefined fuzzy membership functions and finds an optimum rule base from numerical data
%The framework of the genetic algorithm is developed from the original work of D.E. Goldberg
%See his Pascal Codes in 'Genetic Algorithms in Search, Optimization and Machine Learning' 1986, pages:342-349
%Goldberg's SGA code is improved with elitisizm, ranking, and probabilistic - adaptive mutation
%And adopted to a GFRBS design
%The code calls 12 functions which are introduced next: binvecdec.m; flip.m; mfout.m; obj_function.m;...
%plotGA.m; randint.m; rw_selection.m; reportGA_1; reportGA_2; sp_crossover.m; stats.m; tld.m.
%At the end, the code produces i)Best Individual Progress (progress1),
%ii)Average Individual Progress (progress2),
%iii)A data structure including final population statistics and model results (result_pop).

%Produced in                    :10.05.2010
%Last Modified in               :20.06.2010        ***written by Mert Kompil      ***mertkompil@gmail.com


%% LOAD DATA
clc;
clear all;
load -mat HBW_DATA_STRUCTURE.mat;
load -mat MF_VECTORS.mat;
load -mat RULEBASE.mat
%% CREATE GLOBAL VARIABLES for FUNCTION CALLS
tic
global psize; global lbit; global maxgen; global crossprob; global mutprob; global nmutation;
global totcross; global maximum; global minimum; global avg; global sumfitness;
%% SET STARTING PARAMETERS FOR THE GA
   psize=20;                          %Population Size
   lbit=450;                          %Chromosome Length
   rbit=lbit/3;                       %Rulebase Length
   maxgen =250;                       %Maximum Number of Generations
   crossprob=1;                       %Crossover Probability
   mutprob=.001;                      %Mutation Probability
%% CREATE DATA SETS - WRITE VARIABLES INTO STRUCTURES
   %Trip Data
data=train_data;
data.bal_iter=100;
data.diff_1=ones(data.bal_iter,data.n);
data.diff_2=ones(data.m,data.bal_iter);
   %TLD Data
tlddata.trips=data.rowmat(:,6);
tlddata.ttime=data.rowmat(:,5);
tlddata.maxi=max(data.rowmat(:,5));                %the maximum friction value
tlddata.mini=0;                                    %the minimum friction value - set zero as default value
tlddata.bin_int=5;                                 %interval for each bins
tlddata.max_bin=150;                               %the value that bins end
tlddata.num_bins=floor((tlddata.max_bin/tlddata.bin_int)+1);          %number of bins
tlddata.TLD=zeros(tlddata.num_bins,4);             %create a matrix for observed trip length distribution
tlddata.TLD(:,1)=(1:1:tlddata.num_bins);
tlddata.TLD(:,2)=(tlddata.bin_int:tlddata.bin_int:(tlddata.max_bin+tlddata.bin_int))';
tlddata.TLD(tlddata.num_bins,2)=tlddata.maxi;
tlddata.OTLD=tld(data.rowmat(:,6),tlddata);
tlddata.MTLD=zeros(tlddata.num_bins,4);
tlddata.obs_mtc=sum(sum(data.obsmat.*data.ttime))/sum(sum(data.obsmat));
   %Fuzzy Data - Fuzzy Domains, MFs, Antecedents, Consequents
in1=data.rowmat(:,3);
in2=data.rowmat(:,4);
```

```
in3=roundn(data.rowmat(:,5),-1);
x_prod=0:1:1200; x_attr=0:1:2050; x_fric=0:0.1:300; x_fric=roundn(x_fric,-1);
x_trips0=0:0.005:0.2; x_trips1=1:1:39; x_trips2=40:5:140; x_trips3=145:10:235; x_trips4=245:20:885;
x_trips=[x_trips0 x_trips1 x_trips2 x_trips3 x_trips4];
fdata=mfout(data,mfvec,x_prod,x_attr,x_fric,x_trips,in1,in2,in3);
fdata.xtr=x_trips;
[~,fdata.nxtr]=size(x_trips);
[fdata.nrules,~]=size(rulebase);
fdata.consequent=zeros(fdata.nrules,fdata.nxtr);
fdata.fuzzyout= zeros(data.k,1);
fdata.cons0=zeros(fdata.nrules,fdata.nxtr);
  %Rulebase
ruleadd=rulebase(rbit+1:1:fdata.nrules);      %Enables partly search of rulebase
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CREATE INITIAL POPULATION RANDOMLY
for j = 1:psize
  sol.chrom      = round(rand(1,lbit));
  sol.indbin     = zeros(rbit,3);
  sol.indreal    = 0;
  sol.rules      = 0;
  sol.rulebase   = 0;
  sol.tlds       = tlddata.OTLD(:,4);
  sol.modmat     = 0;
  sol.modrow     = 0;
  sol.mserr      = 0;
  sol.srmse      = 0;
  sol.mtce       = 0;
  sol.tldrmse    = 0;
  sol.fitness    = 0;
  sol.p1         = 0;
  sol.p2         = 0;
  sol.xsite      = 0;
  prevgen(j)     = sol;  %Initial Population
end
%% DECODE BINARY NUMBERS INTO DECIMAL NUMBERS (RULE NUMBER)
for j=1:psize
  for i=3:3:lbit
    prevgen(j).indbin(i/3,:)   =prevgen(j).chrom(i-2:i);
  end
  for i=1:rbit
    prevgen(j).indreal(i,1)    =binvec2dec(prevgen(j).indbin(i,:))+1;
  end
  for i=1:rbit
    prevgen(j).rules(i,1)      =rulepool(i,prevgen(j).indreal(i));
  end
end
%%  SET NEW VARIABLES INTO INITIAL POPULATION - EVALUATE FITNESS
parfor j=1:psize
  prevgen(j).rulebase                 =[prevgen(j).rules; ruleadd];
  [modmat,mserr,srmse]                =obj_function(prevgen(j).rulebase,data,fdata);
  prevgen(j).modmat                   =modmat;
  prevgen(j).mserr                    =mserr;
  prevgen(j).srmse                    =srmse;
  [modrow,mtce,tldrmse,MTLD]          =stats(prevgen(j).modmat,data,tlddata);
  prevgen(j).modrow                   =modrow;
  prevgen(j).mtce                     =mtce;
  prevgen(j).tldrmse                  =tldrmse;
  prevgen(j).tlds(:,2)                =MTLD(:,4);
  prevgen(j).fitness                  =prevgen(j).mserr;
end
%% CALCULATE STARTING STATISTICS %%
maximum          = max([prevgen.fitness]);
avg              = mean([prevgen.fitness]);
minimum          = min([prevgen.fitness]);
fitnesssum       = sum([prevgen.fitness]);
nmutation         = 0;
totcross         = 0;
```

```
reportGA_1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CREATE SOME VARIABLES %%
progress1= ones(maxgen,1); progress2= ones(maxgen,1); gen = 0;
%% CREATE MUTATION VARIABLES (POISSON DISTRIBUTION) %%
    nmut0       = ceil(psize*lbit*mutprob);
    nmut1       = ceil(poissrnd(nmut0,maxgen,1).^2);
    nmut        = sort(nmut1,'descend');
%% SET ELITISIZM %%
elit=psize/10;      %Store 10% of the best chromosome
[s,ind] = sort([prevgen.fitness],'ascend');
for i=1:elit
  z=ind(i);
  nextgen(i)=prevgen(z);
end;
%% SET RANKING %%
pow=2.5;        %Use a power function
rank=zeros(psize,4);
rank(:,1)=(psize:-1:1)';
rank(:,2)=rank(:,1).^pow;
for i=1:psize
rank(i,3)=rank(i,2)./sum(rank(:,2));
end
rank(:,4)=ind';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% GENERATE POPULATIONS THROUGH FINAL SOLUTION - START GENERATIONS %%
while gen < maxgen
  gen = gen + 1
  k = elit+1;
  %APPLY SELECTION & CROSSOVER OPERATIONS
  while k <= psize %%%%%%
    % Apply Roulette Wheel Selection
    mate1 = rw_selection(psize, rank);
    mate2 = rw_selection(psize, rank);
    % Apply Single Point Crossover
    [nextgen(k).chrom nextgen(k+1).chrom jcross] = ...
    sp_crossover(prevgen(mate1).chrom, prevgen(mate2).chrom);
%%
  for i=3:3:lbit
    nextgen(k).indbin(i/3,:)          =nextgen(k).chrom(i-2:i);
  end
  for i=1:rbit
    nextgen(k).indreal(i,1)           =binvec2dec(nextgen(k).indbin(i,:))+1;
  end
  for i=1:rbit
    nextgen(k).rules(i,1)             =rulepool(i,nextgen(k).indreal(i));
  end
    nextgen(k).p1         = mate1;
    nextgen(k).p2         = mate2;
    nextgen(k).xsite      = jcross;

  for i=3:3:lbit
    nextgen(k+1).indbin(i/3,:)        =nextgen(k+1).chrom(i-2:i);
  end
  for i=1:rbit
    nextgen(k+1).indreal(i,1)         =binvec2dec(nextgen(k+1).indbin(i,:))+1;
  end
  for i=1:rbit
    nextgen(k+1).rules(i,1)           =rulepool(i,nextgen(k+1).indreal(i));
  end
    nextgen(k+1).p1       = mate1;
    nextgen(k+1).p2       = mate2;
    nextgen(k+1).xsite    = jcross;
    k = k + 2;
  end %%%%%
  %EVALUATE FITNESS
parfor j=1:psize
```

```
   nextgen(j).rulebase                    =[nextgen(j).rules; ruleadd];
   [modmat,mserr,srmse]                    =obj_function(nextgen(j).rulebase,data,fdata);
   nextgen(j).modmat                      =modmat;
   nextgen(j).mserr                       =mserr;
   nextgen(j).srmse                       =srmse;
   nextgen(j).fitness                     =nextgen(j).mserr;
end
   %APPLY MUTATION
      next            = nextgen;
      w1              = ceil(psize.*rand(nmut(gen),1));
      y1              = ceil(lbit.*rand(nmut(gen),1));
      for     i = 1:length(w1)
            if  next(w1(i)).chrom(y1(i))==1;
               next(w1(i)).chrom(y1(i))=0;
            else
               next(w1(i)).chrom(y1(i))=1;
            end
      end
   %COMPUTE MUTATED CHROMOSOMES FITNESS
   %Decode Choromosomes
for j=1:psize
   for i=3:3:lbit
      next(j).indbin(i/3,:)        =next(j).chrom(i-2:i);
   end
   for i=1:rbit
      next(j).indreal(i,1)         =binvec2dec(next(j).indbin(i,:))+1;
   end
   for i=1:rbit
      next(j).rules(i,1)           =rulepool(i,next(j).indreal(i));
   end
end
   %Evaluate Fitness
parfor j=1:psize
   next(j).rulebase                       =[next(j).rules; ruleadd];
   [modmat,mserr,srmse]                   =obj_function(next(j).rulebase,data,fdata);
   next(j).modmat                         =modmat;
   next(j).mserr                          =mserr;
   next(j).srmse                          =srmse;
   next(j).fitness                        =next(j).mserr;
end
   %APPLY ELITISIZM
      [t,ind0]         = sort([nextgen.fitness],'ascend');
      [s,ind1]         = sort([next.fitness],'ascend');
      for i=1:elit
         y=ind0(i);
         z=ind1(i);
         poolgen(i)=nextgen(y);
         poolgen(elit+i)=next(z);
      end;
   [g,ind2] = sort([poolgen.fitness],'ascend');
      for i=1:elit
         z=ind2(i);
         next(i)=poolgen(z);
      end;
   %APPLY RANKING
   [~,ind3] = sort([next.fitness],'ascend');
   rank(:,4)=ind3';
   %CALCULATE STATISTICS and REPORT FOR NEW GENERATION
   sumfitness                  = sum([next.fitness]);
   maximum                     = max([next.fitness]);
   minimum                     = min([next.fitness]);
   avg                         = mean([next.fitness]);
   bestofgen                   = minimum;
   progress1(gen)              = bestofgen;
   progress2(gen)              = avg;
   reportGA_2;
   prevgen                     = next;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% END OF GENERATIONS %%

%% RENAME FINAL POPULATION - COMPUTE TLDs of EACH INDIVIDUAL
result_pop=next;
parfor j=1:psize
   [modrow,mtce,tldrmse,MTLD]        =stats(result_pop(j).modmat,data,tlddata);
   result_pop(j).modrow              =modrow;
   result_pop(j).mtce                =mtce;
   result_pop(j).tldrmse              =tldrmse;
   result_pop(j).tlds(:,1)           =tlddata.OTLD(:,4);
   result_pop(j).tlds(:,2)           =MTLD(:,4);
end
%% SHOW BEST RESULT ON SCREEN - PLOT CONVERGENCE
result=min(progress1)
plotGA(progress1,progress2)
toc
%% CLEAR TEMPORARY FILES %%
clear('avg','bestofgen','crossprob','data','elit','fdata','fitnesssum','full_data','g','gen','i',...
   'in1','in2','in3','ind','ind0','ind1','ind2','ind3','j','jcross','k','lbit','mate1','mate2',...
   'maxgen','maximum','mfvec','minimum','mutprob','next','nextgen','nmut','nmut0','nmut1','test_data',...
    'nmutation','poolgen','pow','prevgen','psize','rank','rbit','ruleadd','rulebase','rulepool',...
    's','sol','sumfitness','t','testdata','tlddata','totcross','traindata','w1','x_attr','x_fric','result',...
    'x_prod','x_trips','x_trips0','x_trips1','x_trips2','x_trips3','x_trips4','y','y1','z','train_data')

%% OUTPUT OF THE PROGRAM %%
% The present code produces three outputs:
% i)Best Individual Progress Through Generations =>progress1,
% i)Average Population Progress Through Generations =>progress1,
% iii)Final Population Results =>resultpop
% resultpop includes best individuals with corresponding
   %rulebase,
   %modelled trips,
   %statistics of the modelled trips including MSE, SRMSE,TLD etc..
```
----

binvec2dec.m

```
%% Convert Binary Numbers to Decimal Numbers
function out = binvec2dec(vec)
%      Example:
%      binvec2dec([1 1 1 0 1]) returns 23
 if isempty(vec)
   error('daq:binvec2dec:argcheck', 'B must be defined.  Type "daqhelp binvec2dec" for more information.');
end
 % Error if B is not a double.
if (~isa(vec, 'double') && ~isa(vec, 'logical'))
   error('daq:binvec2dec:argcheck', 'B must be a binvec.');
end
 % Non-zero values map to 1.
vec = vec~=0;
 % Convert the binvec [0 0 1 1] to a binary string '1100';
h = deblank(num2str(fliplr(vec')))';
 % Convert the binary string to a decimal number.
out = bin2dec(h);
```
----

flip.m

```
% Decide 1 or 0 with Bit-Flip
function bit = flip(p)
   if p == 1
      bit = 1;
   elseif p == 0
      bit = 0;
```

```
    else
        bit = rand <= p;
    end
```

---

mfout.m

```
%% Set Fuzzy MFs - Identify Antecedents
function fuzzydata = mfout(data,mfvec,x_prod,x_attr,x_fric,x_trips,in1,in2,in3)
%% MEMBERSHIP FUNCTIONS
    %production
prodmf_1=trapmf(x_prod,mfvec.prod(1:4));
prodmf_2=trimf(x_prod,mfvec.prod(5:7));
prodmf_3=trimf(x_prod,mfvec.prod(8:10));
prodmf_4=trimf(x_prod,mfvec.prod(11:13));
prodmf_5=trapmf(x_prod,mfvec.prod(14:17));
ant_prod=[prodmf_1;prodmf_2;prodmf_3;prodmf_4;prodmf_5]';
    %attraction
attrmf_1=trapmf(x_attr,mfvec.attr(1:4));
attrmf_2=trimf(x_attr,mfvec.attr(5:7));
attrmf_3=trimf(x_attr,mfvec.attr(8:10));
attrmf_4=trimf(x_attr,mfvec.attr(11:13));
attrmf_5=trapmf(x_attr,mfvec.attr(14:17));
ant_attr=[attrmf_1;attrmf_2;attrmf_3;attrmf_4;attrmf_5]';
    %friction
fricmf_1=trapmf(x_fric,mfvec.fric(1:4));
fricmf_2=trimf(x_fric,mfvec.fric(5:7));
fricmf_3=trimf(x_fric,mfvec.fric(8:10));
fricmf_4=trimf(x_fric,mfvec.fric(11:13));
fricmf_5=trimf(x_fric,mfvec.fric(14:16));
fricmf_6=trapmf(x_fric,mfvec.fric(17:20));
ant_fric=[fricmf_1;fricmf_2;fricmf_3;fricmf_4;fricmf_5;fricmf_6]';
    %trips
tripsmf_1=trapmf(x_trips,mfvec.trips(1:4));
tripsmf_2=trimf(x_trips,mfvec.trips(5:7));
tripsmf_3=trimf(x_trips,mfvec.trips(8:10));
tripsmf_4=trimf(x_trips,mfvec.trips(11:13));
tripsmf_5=trimf(x_trips,mfvec.trips(14:16));
tripsmf_6=trimf(x_trips,mfvec.trips(17:19));
tripsmf_7=trimf(x_trips,mfvec.trips(20:22));
tripsmf_8=trimf(x_trips,mfvec.trips(23:25));
tripsmf_9=trimf(x_trips,mfvec.trips(26:28));
tripsmf_10=trimf(x_trips,mfvec.trips(29:31));
tripsmf_11=trimf(x_trips,mfvec.trips(32:34));
tripsmf_12=trimf(x_trips,mfvec.trips(35:37));
tripsmf_13=trimf(x_trips,mfvec.trips(38:40));
tripsmf_14=trimf(x_trips,mfvec.trips(41:43));
tripsmf_15=trimf(x_trips,mfvec.trips(44:46));
tripsmf_16=trimf(x_trips,mfvec.trips(47:49));
tripsmf_17=trimf(x_trips,mfvec.trips(50:52));
tripsmf_18=trimf(x_trips,mfvec.trips(53:55));
tripsmf_19=trimf(x_trips,mfvec.trips(56:58));
tripsmf_20=trapmf(x_trips,mfvec.trips(59:62));
cons_trips=[tripsmf_1;tripsmf_2;tripsmf_3;tripsmf_4;tripsmf_5;tripsmf_6;tripsmf_7;
          tripsmf_8;tripsmf_9;tripsmf_10;tripsmf_11;tripsmf_12;tripsmf_13;tripsmf_14;
          tripsmf_15;tripsmf_16;tripsmf_17;tripsmf_18;tripsmf_19;tripsmf_20];

%%
antecedents=zeros(150,data.k);
parfor i=1:data.k
mf_p=interp1q(x_prod,ant_prod,in1(i));
mf_a=interp1q(x_attr,ant_attr,in2(i));
mf_f=interp1q(x_fric,ant_fric,in3(i));
antecedents(:,i)=[...
(mf_p(1).*mf_a(1).*mf_f(1));(mf_p(1).*mf_a(2).*mf_f(1));(mf_p(1).*mf_a(3).*mf_f(1));
(mf_p(1).*mf_a(4).*mf_f(1));(mf_p(1).*mf_a(5).*mf_f(1));(mf_p(2).*mf_a(1).*mf_f(1));
(mf_p(2).*mf_a(2).*mf_f(1));(mf_p(2).*mf_a(3).*mf_f(1));(mf_p(2).*mf_a(4).*mf_f(1));
```

```
(mf_p(2).*mf_a(5).*mf_f(1));(mf_p(3).*mf_a(1).*mf_f(1));(mf_p(3).*mf_a(2).*mf_f(1));
(mf_p(3).*mf_a(3).*mf_f(1));(mf_p(3).*mf_a(4).*mf_f(1));(mf_p(3).*mf_a(5).*mf_f(1));
(mf_p(4).*mf_a(1).*mf_f(1));(mf_p(4).*mf_a(2).*mf_f(1));(mf_p(4).*mf_a(3).*mf_f(1));
(mf_p(4).*mf_a(4).*mf_f(1));(mf_p(4).*mf_a(5).*mf_f(1));(mf_p(5).*mf_a(1).*mf_f(1));
(mf_p(5).*mf_a(2).*mf_f(1));(mf_p(5).*mf_a(3).*mf_f(1));(mf_p(5).*mf_a(4).*mf_f(1));
(mf_p(5).*mf_a(5).*mf_f(1));(mf_p(1).*mf_a(1).*mf_f(2));(mf_p(1).*mf_a(2).*mf_f(2));
(mf_p(1).*mf_a(3).*mf_f(2));(mf_p(1).*mf_a(4).*mf_f(2));(mf_p(1).*mf_a(5).*mf_f(2));
(mf_p(2).*mf_a(1).*mf_f(2));(mf_p(2).*mf_a(2).*mf_f(2));(mf_p(2).*mf_a(3).*mf_f(2));
(mf_p(2).*mf_a(4).*mf_f(2));(mf_p(2).*mf_a(5).*mf_f(2));(mf_p(3).*mf_a(1).*mf_f(2));
(mf_p(3).*mf_a(2).*mf_f(2));(mf_p(3).*mf_a(3).*mf_f(2));(mf_p(3).*mf_a(4).*mf_f(2));
(mf_p(3).*mf_a(5).*mf_f(2));(mf_p(4).*mf_a(1).*mf_f(2));(mf_p(4).*mf_a(2).*mf_f(2));
(mf_p(4).*mf_a(3).*mf_f(2));(mf_p(4).*mf_a(4).*mf_f(2));(mf_p(4).*mf_a(5).*mf_f(2));
(mf_p(5).*mf_a(1).*mf_f(2));(mf_p(5).*mf_a(2).*mf_f(2));(mf_p(5).*mf_a(3).*mf_f(2));
(mf_p(5).*mf_a(4).*mf_f(2));(mf_p(5).*mf_a(5).*mf_f(2));(mf_p(1).*mf_a(1).*mf_f(3));
(mf_p(1).*mf_a(2).*mf_f(3));(mf_p(1).*mf_a(3).*mf_f(3));(mf_p(1).*mf_a(4).*mf_f(3));
(mf_p(1).*mf_a(5).*mf_f(3));(mf_p(2).*mf_a(1).*mf_f(3));(mf_p(2).*mf_a(2).*mf_f(3));
(mf_p(2).*mf_a(3).*mf_f(3));(mf_p(2).*mf_a(4).*mf_f(3));(mf_p(2).*mf_a(5).*mf_f(3));
(mf_p(3).*mf_a(1).*mf_f(3));(mf_p(3).*mf_a(2).*mf_f(3));(mf_p(3).*mf_a(3).*mf_f(3))
(mf_p(3).*mf_a(4).*mf_f(3));(mf_p(3).*mf_a(5).*mf_f(3));(mf_p(4).*mf_a(1).*mf_f(3))
(mf_p(4).*mf_a(2).*mf_f(3));(mf_p(4).*mf_a(3).*mf_f(3));(mf_p(4).*mf_a(4).*mf_f(3));
(mf_p(4).*mf_a(5).*mf_f(3));(mf_p(5).*mf_a(1).*mf_f(3));(mf_p(5).*mf_a(2).*mf_f(3));
(mf_p(5).*mf_a(3).*mf_f(3));(mf_p(5).*mf_a(4).*mf_f(3));(mf_p(5).*mf_a(5).*mf_f(3));
(mf_p(1).*mf_a(1).*mf_f(4));(mf_p(1).*mf_a(2).*mf_f(4));(mf_p(1).*mf_a(3).*mf_f(4));
(mf_p(1).*mf_a(4).*mf_f(4));(mf_p(1).*mf_a(5).*mf_f(4));(mf_p(2).*mf_a(1).*mf_f(4));
(mf_p(2).*mf_a(2).*mf_f(4));(mf_p(2).*mf_a(3).*mf_f(4));(mf_p(2).*mf_a(4).*mf_f(4));
(mf_p(2).*mf_a(5).*mf_f(4));(mf_p(3).*mf_a(1).*mf_f(4));(mf_p(3).*mf_a(2).*mf_f(4));
(mf_p(3).*mf_a(3).*mf_f(4));(mf_p(3).*mf_a(4).*mf_f(4));(mf_p(3).*mf_a(5).*mf_f(4));
(mf_p(4).*mf_a(1).*mf_f(4));(mf_p(4).*mf_a(2).*mf_f(4));(mf_p(4).*mf_a(3).*mf_f(4));
(mf_p(4).*mf_a(4).*mf_f(4));(mf_p(4).*mf_a(5).*mf_f(4));(mf_p(5).*mf_a(1).*mf_f(4));
(mf_p(5).*mf_a(2).*mf_f(4));(mf_p(5).*mf_a(3).*mf_f(4));(mf_p(5).*mf_a(4).*mf_f(4));
(mf_p(5).*mf_a(5).*mf_f(4));(mf_p(1).*mf_a(1).*mf_f(5));(mf_p(1).*mf_a(2).*mf_f(5));
(mf_p(1).*mf_a(3).*mf_f(5));(mf_p(1).*mf_a(4).*mf_f(5));(mf_p(1).*mf_a(5).*mf_f(5));
(mf_p(2).*mf_a(1).*mf_f(5));(mf_p(2).*mf_a(2).*mf_f(5));(mf_p(2).*mf_a(3).*mf_f(5));
(mf_p(2).*mf_a(4).*mf_f(5));(mf_p(2).*mf_a(5).*mf_f(5));(mf_p(3).*mf_a(1).*mf_f(5));
(mf_p(3).*mf_a(2).*mf_f(5));(mf_p(3).*mf_a(3).*mf_f(5));(mf_p(3).*mf_a(4).*mf_f(5));
(mf_p(3).*mf_a(5).*mf_f(5));(mf_p(4).*mf_a(1).*mf_f(5));(mf_p(4).*mf_a(2).*mf_f(5));
(mf_p(4).*mf_a(3).*mf_f(5));(mf_p(4).*mf_a(4).*mf_f(5));(mf_p(4).*mf_a(5).*mf_f(5));
(mf_p(5).*mf_a(1).*mf_f(5));(mf_p(5).*mf_a(2).*mf_f(5));(mf_p(5).*mf_a(3).*mf_f(5));
(mf_p(5).*mf_a(4).*mf_f(5));(mf_p(5).*mf_a(5).*mf_f(5));(mf_p(1).*mf_a(1).*mf_f(6));
(mf_p(1).*mf_a(2).*mf_f(6));(mf_p(1).*mf_a(3).*mf_f(6));(mf_p(1).*mf_a(4).*mf_f(6));
(mf_p(1).*mf_a(5).*mf_f(6));(mf_p(2).*mf_a(1).*mf_f(6));(mf_p(2).*mf_a(2).*mf_f(6));
(mf_p(2).*mf_a(3).*mf_f(6));(mf_p(2).*mf_a(4).*mf_f(6));(mf_p(2).*mf_a(5).*mf_f(6));
(mf_p(3).*mf_a(1).*mf_f(6));(mf_p(3).*mf_a(2).*mf_f(6));(mf_p(3).*mf_a(3).*mf_f(6));
(mf_p(3).*mf_a(4).*mf_f(6));(mf_p(3).*mf_a(5).*mf_f(6));(mf_p(4).*mf_a(1).*mf_f(6));
(mf_p(4).*mf_a(2).*mf_f(6));(mf_p(4).*mf_a(3).*mf_f(6));(mf_p(4).*mf_a(4).*mf_f(6));
(mf_p(4).*mf_a(5).*mf_f(6));(mf_p(5).*mf_a(1).*mf_f(6));(mf_p(5).*mf_a(2).*mf_f(6));
(mf_p(5).*mf_a(3).*mf_f(6));(mf_p(5).*mf_a(4).*mf_f(6));(mf_p(5).*mf_a(5).*mf_f(6))];
end;
fuzzydata.const=cons_trips;
fuzzydata.ant=antecedents;
```

---

obj_function.m

```
%% Evaluate Fitness of the Chromosomes
function  [modmat,mse_out,srmse_out]= obj_function(rules,data,fdata)
%% FIND FUZZY OUTPUT %%%%%%%%%%
for i=1:fdata.nrules
fdata.cons0(i,:)=fdata.const(rules(i),:);
end;
%%
for i=1:data.k
   for j=1:fdata.nxtr
   fdata.consequent(:,j)=(fdata.cons0(:,j).*fdata.ant(:,i));
   end;
aggregation= max(fdata.consequent);
output0= defuzz(fdata.xtr,aggregation,'centroid');
```

```
fdata.fuzzyout(i)=output0;
end;

%% FIND SRMSE %%%%%%%%%
mod=fdata.fuzzyout;
obsmat=data.obsmat;
m=data.m;
n=data.n;
%% Convert to Matrix
t=1;
c=m;
for i=1:n
   data.modmat(:,i)=mod((t:1:c)',1);
   t=t+m;
   c=c+m;
end;
%% Balance Matrix
modmat=data.modmat;
attr_mod=sum(modmat);
for d=1:data.bal_iter
   data.diff_1(d,:)=data.attrtot./attr_mod;
   for i=1:m
      for j=1:n
      modmat(i,j)=data.diff_1(d,j).*modmat(i,j);
      end;
   end;
   prod_mod=sum(modmat,2)';
   data.diff_2(:,d)=data.prodtot./prod_mod;
   for i=1:m
      for j=1:n
      modmat(i,j)=data.diff_2(i,d).*modmat(i,j);
      end;
   end;
   attr_mod=sum(modmat);
end;
 %% Calculate MSE
mse_out=mse(obsmat-modmat);
%% Calculate SRMSE
srmse_out=(sqrt((sum(sum((modmat-obsmat).^2))./(m*n)))/(sum(sum(obsmat))/(m*n));
```

---

plotGA.m

```
%% Plot Convergence
function  plotGA(progress1,progress2)
% DRAW CONVERGENCE I
figure;
semilogy(progress1);
title('Convergence to Solution');
xlabel('Iterations');
ylabel('Best Individual in Whole Generation')
fprintf('\n')
% DRAW CONVERGENCE II
figure;
semilogy(progress2);
title('Average Population Progress ');
xlabel('Iterations');
ylabel('Average Fitness')
fprintf('\n')
```

---

randint.m

```
% Creat a random integer from a closed interval [a,b]
function r = randint(low,high)
   if low >= high
      r = low;
```

```
   else
      r = low + fix(rand* (high - low + 1));
   end
```

---

reportGA_1.m

```
%% Report Starting Options
fprintf('\n')
fprintf('\n')
fprintf('   GA Parameters\n')
fprintf('   ----------------------------------------------------\n')
fprintf('   Population size (psize)                    =   %d\n', psize)
fprintf('   Chromosome length (lbit)                    =   %d\n',lbit)
fprintf('   Maximum # of generation (maxgen)     =   %d\n',maxgen)
fprintf('   Crossover probability (crossprob)          =   %f\n' ,crossprob)
fprintf('   Mutation  probability (mutprob)           =   %f\n',mutprob)
fprintf('   ----------------------------------------------------\n')
fprintf('\n')
fprintf('\n')
fprintf('   Starting Population Statistics\n')
fprintf('   ----------------------------------------------------\n')
fprintf('   Maximum fitness = %f\n',maximum)
fprintf('   Average fitness = %f\n',avg)
fprintf('   Minimum fitness = %f\n',minimum)
fprintf('   Sum  of fitness  = %f\n',fitnesssum)
fprintf('   ----------------------------------------------------\n')
```

---

reportGA_2.m

```
%% Report Population Fitness in Each Genration
fprintf('\n')
fprintf('\n')
fprintf('Generation Report\n')
fprintf('-------------------------------------------------------------------------------------------------------------------------------\n');
fprintf('%1s %d %46s %d\n', 'Generation', gen-1, '|Generation', gen);
fprintf('-------------                                |-------------\n')
fprintf(' #     code            x          fitness     |  #   mates   xsite        code           x          fitness   ');
fprintf('   \n')
fprintf('-----------------------------------------------|-------------------------------------------------------------------------------\n');

for j = 1:psize
   fprintf('%2d)    ', j);
   fprintf('%d', prevgen(j).chrom);
   fprintf('    %10d', prevgen(j).srmse);
   fprintf('    %10d', prevgen(j).fitness);
   fprintf('%9s %3d) (%2d,%2d)  ', '|', j, next(j).p1, next(j).p2);
      if next(j).xsite(1) ~= -1
         fprintf('[%2d]          ', next(j).xsite(1));
      else
         fprintf('[No]          ');
      end
   fprintf('%d', next(j).chrom);
   fprintf('    %10d', next(j).srmse);
   fprintf('    %11d\n', next(j).fitness);
end
fprintf('-------------------------------------------------------------------------------------------------------------------------------\n');
fprintf('Generation %d', gen);
fprintf('\n')
fprintf('Statistics:')
fprintf('  sum = %g', fitnesssum);
fprintf('  max = %g', maximum);
fprintf('  min = %g', minimum);
fprintf('  avg = %g', avg);
fprintf('\n')
fprintf('           ')
```

```
fprintf('  nmutation = %d', nmut(gen));
fprintf('  totcross = %d', totcross);
fprintf('  solution for the generation -----> %d\n', bestofgen);
fprintf('-------------------------------------------------------------------------------------------------------------------------------------------\n');
fprintf('\n')
fprintf('\n')
```

---

rw_selection.m

```
% Apply Roulette Wheel Slection
function p = rw_selection(psize, rank)
   partsum = 0;
   j = 0;
   randnum =  rand;
   while (partsum < randnum && j < psize)
      j = j+1;
      partsum = partsum + rank(j,3);
   end
p=rank(j,4);
```

---

sp_crossover.m

```
%% Apply Crossover
function [child1 child2 jcross] = sp_crossover(parent1, parent2)
   global crossprob;
   global totcross;
   global lbit;

   if flip(crossprob)  == 1
      jcross = randint(1, lbit - 1);
      totcross = totcross + 1;

      for j = 1:jcross
         child1(j) = (parent1(j));
         child2(j) = (parent2(j));
      end

      for j = jcross+1:lbit
         child1(j) = (parent2(j));
         child2(j) = (parent1(j));
      end
   else
      jcross = -1;
      for j = 1:lbit
         child1(j) = (parent1(j));
         child2(j) = (parent2(j));
      end
   end
```

---

stats.m

```
%% Calculate Statistics
function  [mod_row,mtce_out,tldrmse_out,tld_out]= stats(modmat,data,tlddata)
m=data.m;
n=data.n;
k=data.k;
%% Calculate MTCE
mod_mtc=sum(sum(modmat.*data.ttime))/sum(sum(modmat));
mtce=(tlddata.obs_mtc-mod_mtc);
%% Calculate TLD RMSE
      convert=ones(k,1);
      v=1;
      for o=1:m
         temp=modmat(:,v);
```

```
            convert(o,1)=temp(o);
        end;
        v=v+1;
        t=1;
        x=0;
        for u=1:n-1
            for f=t:m*u
            temp=modmat(:,v);
            convert(m+f,1)=temp(x+f);
            end;
            t=t+m;
            v=v+1;
            x=x-m;
        end;
       mod_row=convert;
tld_out=tld(mod_row,tlddata);
tlddata.MTLD=tld_out;
tld_rmse =(sqrt((sum((tlddata.OTLD(:,4)-tlddata.MTLD(:,4)).^2))))/tlddata.num_bins;
%%
mtce_out=mtce;
tldrmse_out=tld_rmse;
```

---

tld.m

```
%% Compute Trip Length Distribution
function tld_output = tld(trips,tlddata)
row_trips=trips;
row_ttime=tlddata.ttime;
TLD0=tlddata.TLD;
[h,~]=size(row_trips);                          %matrix index
for i=1:h                                       %produce observed trip length distribution
    for k=1:tlddata.num_bins-1
        if tlddata.mini <= row_ttime(i) && row_ttime(i) < TLD0(1,2)
            TLD0(1,3)= TLD0(1,3)+row_trips(i);
            break
        end;
        if TLD0(k,2) <= row_ttime(i) && row_ttime(i)< TLD0(k+1,2)
            TLD0(k+1,3)= TLD0(k+1,3)+row_trips(i);
            break
        end;
    end;
end;

for i=1:tlddata.num_bins
    TLD0(i,4)=(TLD0(i,3)/sum(TLD0(:,3)))*100;
end;
tld_output=TLD0;
```

APPENDIX B

RULE BASES

B.1. Rule Base of the Fuzzy Rule-Based System (FRBS)

Rule Number	Antecedents	Consequents
Rule 1	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_3
Rule 2	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_3
Rule 3	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_4
Rule 4	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_5
Rule 5	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_6
Rule 6	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_5
Rule 7	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_8
Rule 8	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{11}
Rule 9	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{11}
Rule 10	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{15}
Rule 11	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_{11}
Rule 12	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{11}
Rule 13	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{11}
Rule 14	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{12}
Rule 15	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{17}
Rule 16	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_6
Rule 17	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{13}
Rule 18	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{15}
Rule 19	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{14}
Rule 20	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{19}
Rule 21	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_8
Rule 22	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{16}
Rule 23	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{18}
Rule 24	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{19}
Rule 25	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{20}
Rule 26	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_2
Rule 27	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_2
Rule 28	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_4
Rule 29	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_4
Rule 30	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_5
Rule 31	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_2
Rule 32	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_4
Rule 33	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_6
Rule 34	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_7
Rule 35	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_8
Rule 36	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_3
Rule 37	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_6
Rule 38	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_7
Rule 39	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_8
Rule 40	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{13}
Rule 41	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_5
Rule 42	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_7
Rule 43	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_9
Rule 44	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_{10}
Rule 45	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{12}

Rule 46	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_3
Rule 47	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_8
Rule 48	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_{11}
Rule 49	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_{15}
Rule 50	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{16}
Rule 51	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_1
Rule 52	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_1
Rule 53	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_2
Rule 54	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_2
Rule 55	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_2
Rule 56	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_2
Rule 57	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_2
Rule 58	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_4
Rule 59	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_6
Rule 60	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_4
Rule 61	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_2
Rule 62	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_4
Rule 63	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_5
Rule 64	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_1
Rule 65	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_7
Rule 66	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_3
Rule 67	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_5
Rule 68	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_6
Rule 69	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_{10}
Rule 70	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_{10}
Rule 71	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_3
Rule 72	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_4
Rule 73	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_7
Rule 74	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_9
Rule 75	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_{10}
Rule 76	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 77	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_1
Rule 78	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_1
Rule 79	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_2
Rule 80	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_2
Rule 81	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 82	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_2
Rule 83	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_2
Rule 84	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_3
Rule 85	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_4
Rule 86	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 87	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_2
Rule 88	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_3
Rule 89	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_5
Rule 90	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_6
Rule 91	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 92	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_3
Rule 93	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_4
Rule 94	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_6
Rule 95	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_7
Rule 96	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 97	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_3
Rule 98	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_4
Rule 99	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_7
Rule 100	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_8
Rule 101	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 102	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 103	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_1
Rule 104	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_2
Rule 105	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_1

Rule 106	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 107	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 108	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 109	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_4
Rule 110	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_2
Rule 111	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 112	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_2
Rule 113	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 114	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_3
Rule 115	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_3
Rule 116	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 117	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_2
Rule 118	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 119	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_6
Rule 120	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_5
Rule 121	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 122	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_2
Rule 123	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 124	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_4
Rule 125	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_5
Rule 126	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 127	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 128	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 129	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 130	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 131	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 132	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 133	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 134	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 135	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_2
Rule 136	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 137	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 138	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 139	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 140	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 141	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 142	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 143	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 144	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 145	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 146	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 147	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 148	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 149	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 150	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_2

B.2. Rule Base of the Genetic Fuzzy Rule-Based System (GFRBS)

Rule Number	Antecedents	Consequents
Rule 1	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_5
Rule 2	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_5
Rule 3	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_5
Rule 4	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_6
Rule 5	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_5
Rule 6	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_7
Rule 7	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_8
Rule 8	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{10}
Rule 9	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{10}
Rule 10	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{13}
Rule 11	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_9
Rule 12	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{11}
Rule 13	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{10}
Rule 14	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{13}
Rule 15	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{15}
Rule 16	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_7
Rule 17	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{13}
Rule 18	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{15}
Rule 19	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{16}
Rule 20	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{16}
Rule 21	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_1	THEN T_i is MF_9
Rule 22	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_1	THEN T_i is MF_{17}
Rule 23	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_1	THEN T_i is MF_{19}
Rule 24	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_1	THEN T_i is MF_{20}
Rule 25	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_1	THEN T_i is MF_{20}
Rule 26	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_4
Rule 27	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_3
Rule 28	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_6
Rule 29	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_5
Rule 30	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_5
Rule 31	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_2
Rule 32	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_5
Rule 33	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_6
Rule 34	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_7
Rule 35	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_7
Rule 36	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_3
Rule 37	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_6
Rule 38	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_6
Rule 39	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_7
Rule 40	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{11}
Rule 41	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_4
Rule 42	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_5
Rule 43	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_9
Rule 44	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_{12}
Rule 45	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{13}
Rule 46	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_2	THEN T_i is MF_6
Rule 47	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_2	THEN T_i is MF_7
Rule 48	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_2	THEN T_i is MF_{11}
Rule 49	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_2	THEN T_i is MF_{12}
Rule 50	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_2	THEN T_i is MF_{14}
Rule 51	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_3
Rule 52	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_2
Rule 53	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_4
Rule 54	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_3
Rule 55	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_5

Rule 56	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_1
Rule 57	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_2
Rule 58	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_5
Rule 59	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_6
Rule 60	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_5
Rule 61	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_2
Rule 62	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_3
Rule 63	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_5
Rule 64	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_5
Rule 65	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_6
Rule 66	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_3
Rule 67	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_5
Rule 68	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_6
Rule 69	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_8
Rule 70	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_9
Rule 71	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_3	THEN T_i is MF_3
Rule 72	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_3	THEN T_i is MF_4
Rule 73	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_3	THEN T_i is MF_6
Rule 74	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_3	THEN T_i is MF_8
Rule 75	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_3	THEN T_i is MF_{10}
Rule 76	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 77	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_1
Rule 78	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_2
Rule 79	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_3
Rule 80	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_3
Rule 81	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 82	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_2
Rule 83	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_2
Rule 84	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_4
Rule 85	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_3
Rule 86	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 87	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_2
Rule 88	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_3
Rule 89	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_3
Rule 90	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_6
Rule 91	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_2
Rule 92	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_1
Rule 93	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_5
Rule 94	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_5
Rule 95	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_6
Rule 96	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_4	THEN T_i is MF_1
Rule 97	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_4	THEN T_i is MF_2
Rule 98	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_4	THEN T_i is MF_5
Rule 99	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_4	THEN T_i is MF_6
Rule 100	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_4	THEN T_i is MF_7
Rule 101	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 102	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 103	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_1
Rule 104	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_1
Rule 105	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_1
Rule 106	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 107	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 108	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_1
Rule 109	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_4
Rule 110	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_2
Rule 111	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 112	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 113	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 114	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_3
Rule 115	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_3

Rule 116	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 117	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 118	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 119	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_4
Rule 120	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_3
Rule 121	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_5	THEN T_i is MF_1
Rule 122	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_5	THEN T_i is MF_1
Rule 123	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_5	THEN T_i is MF_2
Rule 124	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_5	THEN T_i is MF_4
Rule 125	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_5	THEN T_i is MF_4
Rule 126	IF P_i is MF_1 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 127	IF P_i is MF_1 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 128	IF P_i is MF_1 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 129	IF P_i is MF_1 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 130	IF P_i is MF_1 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 131	IF P_i is MF_2 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 132	IF P_i is MF_2 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 133	IF P_i is MF_2 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 134	IF P_i is MF_2 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 135	IF P_i is MF_2 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 136	IF P_i is MF_3 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 137	IF P_i is MF_3 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 138	IF P_i is MF_3 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 139	IF P_i is MF_3 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 140	IF P_i is MF_3 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 141	IF P_i is MF_4 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 142	IF P_i is MF_4 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 143	IF P_i is MF_4 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 144	IF P_i is MF_4 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 145	IF P_i is MF_4 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1
Rule 146	IF P_i is MF_5 and A_i is MF_1 and F_i is MF_6	THEN T_i is MF_1
Rule 147	IF P_i is MF_5 and A_i is MF_2 and F_i is MF_6	THEN T_i is MF_1
Rule 148	IF P_i is MF_5 and A_i is MF_3 and F_i is MF_6	THEN T_i is MF_1
Rule 149	IF P_i is MF_5 and A_i is MF_4 and F_i is MF_6	THEN T_i is MF_1
Rule 150	IF P_i is MF_5 and A_i is MF_5 and F_i is MF_6	THEN T_i is MF_1

VITA

PERSONAL INFORMATION

Date of Birth	**1977**
Place of Birth	**Denizli**
e-mail	**mertkompil@iyte.edu.tr** or **mertkompil@gmail.com**

EDUCATION

2004 – 2010	**Ph.D. Program in City Planning** Izmir Institute of Technology, Faculty of Architecture, Dep. of City and Regional Planning, Izmir, Turkey.
2001 – 2004	**Master of Science Program in City Planning** Izmir Institute of Technology, Faculty of Architecture, Dep. of City and Regional Planning, Izmir, Turkey.
2000 – 2001	**Non-Degree Advanced English Education Course** Izmir Institute of Technology, Graduate School of Engineering and Science, The English Preparatory School, Izmir, Turkey.
1996 – 2000	**Bachelor of City and Regional Planning** Dokuz Eylül University, Faculty of Architecture, Dep. of City and Regional Planning, Izmir, Turkey.

WORK EXPERIENCE

2002 – 2010	Research and Teaching Assistant Izmir Institute of Technology, Faculty of Architecture, Dep. of City and Regional Planning, Izmir, Turkey.
Teaching Experience	Quantitative Techniques in Planning, 8 semesters. Statistical Methods for Planners, 4 semesters. Real Estate Economics, 7 semesters. Planning Design Studios, 6 semesters.

RESEARCH INTERESTS

Primary Interests	Spatial Interaction Models, Trip Distribution Models, Transport Modelling, Soft Computing Applications (Fuzzy Logic – Artificial Neural Networks – Genetic Algorithms) in Transportation and Urban Geography.
Secondary Interests	Retail Geography and Retail Location Analysis, Quantitative Techniques in Regional Science, Statistical Analysis and Probabilistic Modelling, Real Estate Appraisal and Engineering Economics.