Building Transparent Construction Performance Models via Techniques of Computational Intelligence

University of Alberta

Building Transparent Construction Performance Models via Techniques of Computational Intelligence

by

Long Chen ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

in

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2005

University of Alberta

Library Release Form

Name of Author: *Long Chen*

Title of Thesis: *Building Transparent Construction Performance Models via Techniques of Computational Intelligence*

Degree: *Master of Science*

Year this Degree Granted: *2005*

Date: 2005/09/30

ABSTRACT

Building transparent and highly interpretable models of the construction performance is generally of significant importance to construction managers. However, previous research focuses more on the approximation accuracy of construction performance models. Few studies have been done on the transparency of models, i.e., offering some understandable cause-effect relationships between the construction performance indicator and its influence factors.

The general objective of this thesis is to build transparent construction performance models using techniques of Computational Intelligence (CI). More specifically:

- First, a neural network, named General Regression Neural Network (GRNN) is selected as the basic modeling technique. Its new genetic algorithm based learning algorithm is introduced. The GRNN not only presents a high approximation rate, but also offers importance indices about the influence of inputs on the output.
- Secondly, a fuzzy clustering algorithm is introduced to granulate the inputs into their linguistic terms. The model built with the use of granulated data provides clearer influence factors and the indicator of resulting construction performance.

All the proposed methods are tested on the data collected from construction sites. The results demonstrate the feasibility and efficiency of the proposed models.

Acknowledgements

For his indispensable guidance during my graduate studies and research at the University of Alberta, my supervisor, Dr. Witold Pedrycz is gratefully acknowledged. I wish like to acknowledge Dr. Aminah Robinson, for her research project with Dr. Pedrycz on construction performance offering me the opportunity to study the topics discussed in this thesis. Additionally, I would like to thank Manjula Dissanayake for his comments and advice during our cooperation in the research project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

Building a transparent model of construction performance [1] is generally of significant importance to construction managers. Such a model assists them in understanding and managing the construction performance under the pressure of quick construction pace, short project life cycles, complex construction designs and ubiquitous human factors. In particular, when problems occur, such as when the construction performance greatly diverges from the planed one, a transparent construction performance model will help the manager to provide possible explanations for the observed problems and take remedial action. However, previous research focuses more on the approximation accuracy of construction performance models. Regardless of the regression analysis of the construction labor productivity [61-62], the neural networks modeling of construction productivity [47], or the fuzzy modeling of construction design performance [54], the first consideration in those studies is to accurately predict construction performance based on the values of influence factors. Little research work has been done on the transparency of models, i.e., offering some understanding of cause-effect relationships between the construction performance indicator and its influence factors.

1.1 Objectives & Thesis Organization

The general objective of this thesis is to build a transparent construction performance model using techniques of Computational Intelligence (CI) [2]. Because such a model is the kernel of construction performance analysis, its transparency is represented by two desired features, which constitute the specific objectives of this thesis:

- First, building a model that offers importance indices about the influence of inputs on the output.

- Secondly, ensuring the model granulates the inputs and output into their linguistic terms to provide more concrete influence factors and a more accurate indicator of resulting construction performance.

In Chapter 2, we introduce a framework for construction performance diagnosis; then, the two anticipated features for the transparent construction performance model in the framework are elaborated on. Chapter 3 investigates the models with first feature. The CI technique, General Regression Neural Network (GRNN) is selected as the basic modeling technique. The GRNN is strengthened to provide importance indices about the influence of inputs on the output, and its new genetic learning algorithm is also proposed. Chapter 4 discusses the second feature of information granulation, and another CI technique called fuzzy clustering is proposed as the basic granulation method to enhance the transparency of construction performance models. The concept of representation error is introduced, and is applied to improve the performance of the fuzzy clustering algorithm. Finally, Chapter 5 concludes this thesis and suggests some future work.

The remainder of this chapter presents an introduction to kernel CI techniques and a review of motivating previous research on CI applied in construction engineering and management. Section 1.2 first introduces the three most broadly used CI techniques, including neural networks [3], fuzzy sets [4], and genetic algorithms [5]. Then, the three techniques' combinations and their advantages and disadvantages are discussed. Section 1.3 provides a review of previous applications of CI techniques to construction engineering and management. Further attention will be devoted to preceding modeling techniques for construction performance and lessons learned from them.

1.2 Computational Intelligence

As defined by the IEEE Society of Computational Intelligence, the scope of Computational Intelligence is "the theory, design, application, and development of biologically and linguistically motivated computational paradigms emphasizing neural networks, connectionist systems, genetic algorithms, evolutionary programming, fuzzy systems, and hybrid intelligent systems in which these paradigms are contained" [6]. The three most essential techniques defined in this scope – neural networks, genetic algorithms and fuzzy sets & systems are introduced in the following subsections. The pros and cons of individual techniques and their combinations are also presented.

1.2.1 Neural Networks

Neural Networks or Artificial Neural Networks (ANN) [7-8] represent an artificial simulation of biological nervous systems. Just like the object it simulates, ANN is also a parallel processing structure that combines a number of interconnected simple processing elements named Artificial Neurons. Figure 1.1 illustrates a typical Artificial Neuron. The input-output mapping represented in this neuron is:

$$y = f(I), I = \sum_{i=1}^{n} w_i x_i ,$$

in which w_i are weights, x_i are inputs, and the f is activation function.

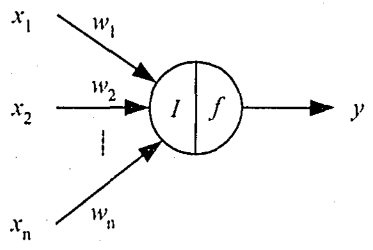


Figure 1.1 - The structure of a typical artificial neuron

The typical architecture of the ANN is a sequence of layers with full connections between the layers. Each layer constitutes a number of neurons. Figure 1.2 is an example of the well-known ANN, named Feed-forward Neural Network [8].

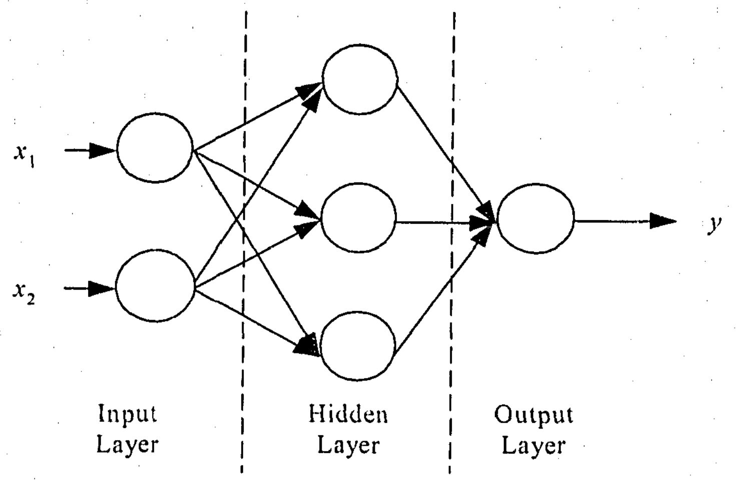


Figure 1.2 - The structure of a three-layered Feed-forward Neural Network

Learning and Recall are the two main modes of ANN operation [7]. In the learning mode, the ANN tunes weights of connections between neurons to capture the knowledge embedded in historical data. In the recall mode, the ANN maps inputs to the desired output based on the knowledge represented by the architecture and connections of ANN. To effectively adjust the connections (weights) in the learning mode of ANN, a learning algorithm is a must.

To capture the knowledge embedded in historical data, usually, a performance index, such as the difference between the outputs of historical data and those of ANN, is defined and the learning problem is changed into a search task – exploring the parameter space to minimize the performance index with respect to the parameters (in ANN, the parameters are connection weights).

Many learning algorithms for ANN have been brought forward in past decades. However, most of them – including the highly successful learning algorithm, "back-propagation"[9], suffer from the limitation of traditional search/optimization methods, i.e., the problem of falling into the local minima of the performance index. In the techniques applied in the newly developed learning algorithms of ANN to alleviate the above-mentioned problem, the one most widely used is the genetic algorithm. The next subsection provides a brief introduction to this algorithm.

1.2.2 Genetic Algorithms

The Genetic Algorithm (GA) is one of the main instances of Evolutionary optimization [2], which include Genetic Algorithms [10-12], Evolutionary Programming [13-14], Evolutionary Strategies [15-16], Genetic Programming [17-18] and Learning Classification Systems [19-20]. Evolutionary Computation is a powerful search/optimization paradigm inspired by the mechanics of Darwinian selection and biological evolution. There are similar steps in all the Evolutionary Computation models. Figure 1.3 illustrates the fundamental steps in a genetic algorithm.

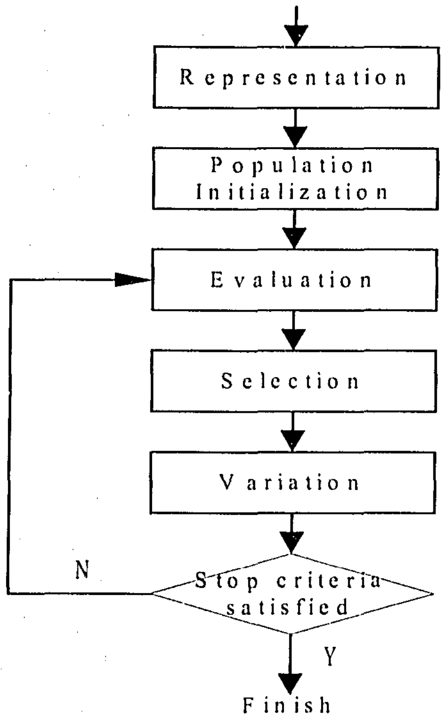


Figure 1.3 - The schema of a typical Genetic Algorithm

First, in the **Representation step,** "a mapping from the state space of possible solutions to a state space of encoded solutions within a particular data structure"[21] is defined. The particular data structure varies from a vector of binary integers, or a vector of real variables to a complex tree structure, or a symbolic expression. When the vector of binary integers is applied as the particular data structure, the Genetic Algorithm is called Binary-Coded Genetic Algorithm (BCGA) [10]. Similarly, if the vector of real variables is applied, the Genetic Algorithm is called Real-Coded Genetic Algorithm (RCGA) [22-23]. This thesis will apply RCGA in the learning of a proposed neural network. Chapter 3 provides further details.

The Genetic Algorithm holds a population of individuals for evolution. Each individual encodes a possible solution, as described in the **Representation step**. In the **Population Initialization step**, the population of individuals in the space of encoded solutions is randomly generated. The number of individuals in the population is referred to as population size.

In the **Evaluation step**, a fitness value is assigned to each individual to represent the quality of the solution encoded in the individual. The fitness function is defined to establish the fitness values, and normally relies on the performance indices of corresponding solutions. For example, if the problem we try to solve by GA is to minimize the performance index, the individual with lower performance index will be issued a higher fitness value by the fitness function.

The **Selection step** is the one of the most important steps in the evolution. In this step, individuals in the population are selected for the next step of the algorithm the **Variation step** based on their fitness values. The **Selection step** is an imitation of selection procedure in biological evolution: high quality individuals are selected for the next generation, and low quality ones are washed out.

The **Variation step** is another important step in evolution procedure; it is a simulation of biological mutation and mating. In this step, individuals selected by the last step, in a probability, are modified by variation operations taking into account the new aspects of solution space. The typical variation operations are crossover and mutation. After this step, a new population of individuals or a new generation is produced.

Finally, the algorithm will return to the Evaluation step unless the stop criteria are satisfied. The typical stop criterion is achieving the maximum number of generations. When the stop criteria of GA are satisfied, the best individual in the last generation of population is selected. The solution encoded in the selected individual is the best solution to the problem searched by the GA.

Basically, the Genetic Algorithm provides a stochastic search method in the complex state space of possible solutions. Although it is principally a blind search method, the Selection step offers the possibility of leading the direction of the search to the ideal area of the solution space, where the optimal solutions can be found with higher probability. In the meantime, the Variation step suggests the possibility of jumping out of local minima. Due to the above-mentioned properties, the GA was recommended as a primary method in complex search tasks such as the learning of ANN.

1.2.3 Fuzzy Sets & Systems

Normally, the elements discussed by a classic set either belong to the set or not; there is a "clear" boundary between "belong" and "not belong". A good example of this kind of classic set is "All the students in the class aged over 20". Obviously, every student in the class either belongs to the set or not. More formally, set A, defined in domain of X, is described by its characteristic function $A(x): X \to \{0,1\}$:

$$A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases},$$

in which $A(x)$ represents the belongingness of element x to the set A.

Although the set theory is currently the foundation of modern mathematics, it is insufficient for the real problems involving imprecision, uncertainty, and ambiguity. For example, when people are talking about the set, "All the old students in the class", each student in the class will no longer either "belong" or "not belong" to the set. There is a "fuzzy" boundary between "belong" and "not belong". To mathematically tackle this type of set, Dr. Zadeh defined the concept of **fuzzy set** in 1965 [24]. For a fuzzy set A, it is also fully determined by the characteristic function. However, the characteristic function is not simply a function taking the value of 0 or 1; it is a function taking the value in the interval [0,1]. Formally, the fuzzy set A, is defined by the characteristic function, **membership function** $A(x): X \rightarrow \{0,1\}$, in which the $A(x)$ still represents the belongingness of x to the set A. For instance, we can define the fuzzy set A="All the old students in the class" by the membership function $A(x): X \rightarrow \{0,1\}$;

$$A(x)=\begin{cases} 0, & 0<x\leq 18 \\ (x-18)/10, & 18<x\leq 28 \\ 1, & x>28 \end{cases}$$

The main benefit offered by fuzzy set theory is that it provides researchers with an outstanding tool to deal with the imprecision, uncertainty, and ambiguity in the real world, especially the linguistic descriptions given by human beings [25-26]. Linguistic terms such as "Warm Water", "High Speed", and "Low Temperature" can now be tackled by well-defined fuzzy sets. Furthermore, researchers have utilized linguistic terms in expert systems and built a new system modeling technique, Fuzzy System [27-28], in which the system is fully defined by a group of fuzzy IF-THEN rules. For example, Figure 1.4 lists a simple fuzzy rule set for a construction system, in which the worker's skill level and the supervisor's supervision level completely determine the construction productivity.

IF Skill Level is High and Supervision is Good Then Productivity is High

IF Skill Level is High and Supervision is Bad Then Productivity is Average

IF Skill Level is Low and Supervision is Good Then Productivity is Average

IF Skill Level is Low and Supervision is Bad Then Productivity is Low

Figure 1.4 - A simple fuzzy rule set of a fuzzy system

Given the membership functions defining the fuzzy sets or linguistic terms, the group of IF-THEN rules in the fuzzy rule set fully provide a knowledge-based representation of the input-output mapping of the system. However, building acceptable membership functions for linguistic terms is not a straightforward task. Chapter 4 will offer a modified fuzzy clustering based algorithm for generating membership functions from historical data.

1.2.4 Hybrid Systems

As mentioned in previous subsections, there are a number of specific characteristics possessed by Neural Networks, Genetic Algorithms, and Fuzzy Sets & Systems. To maximize their advantages and minimize their disadvantages, hybrid systems that apply two or more techniques in one system have been proposed in the last few decades and integrated as an important part of CI techniques. Table 1.1 lists the pros and cons of the simple CI techniques and their major hybrids.

Techniques	Simple Descriptions	Advantages	Disadvantages
Neural Networks (NN)	Linked and layered neurons that map inputs to outputs	- Learning capability - No need to be concerned about the systems internal state	- Black-box - Not easy to determine the NN's structure - May suffer from over-fitting
Fuzzy Systems (FS)	Incorporate expert knowledge in fuzzy If-Then rules	- Easy to express expert knowledge - Fuzzy approximate reasoning of FS is a mature technique	- Less learning capability - Hard to cope with complex systems where expert knowledge is scarce.
Genetic Algorithms (GA)	A search algorithm finding the best solution in a search space	- Not easy to be trapped in local best solution - Easily parallelized	- Somewhat slow - Tricky to encode a solution into a individual in GA - Some parameters in GA are not easy to determine
Genetic Neural Networks (GNN) [29-31]	Neural Networks learned via GA	- Provide greater learning power - Avoid falling into local minima inherent in traditional learning methods	- Similar to NN
Neuro-fuzzy Systems(NFS) [32-33]	Represent fuzzy systems with NN	- Incorporate learning capability into FS - Maintain the advantages of fuzzy systems	- Difficult to cope with the complex systems where expert knowledge is scarce.
Fuzzy Neural Networks (FNN)[34-35]	Neural Networks with Specific Fuzzy Neuron	- Provide NN some transparency	- Simple FNN's approximation capability is limited
Genetic FNN or Genetic NFS [36-39]	Applying GA in FNN and NFS' learning algorithms	- Provide greater learning power - Avoid falling into local minima inherent in traditional learning methods	- Similar to FNN and NFS
Fuzzy C-means Cluster [40-41]	Clustering technique. All data has membership values to all the clusters	- Able to map clusters to Fuzzy membership function	- Determining the number of clusters and the fuzzfication factor is not trivial

Table 1.1 - The pros and cons of the computational techniques and their hybrids

1.3 Inspirational Previous Studies

In Civil Engineering, especially in Construction Engineering and Management, interest in the application of biologically and linguistically motivated computing techniques has grown very quickly in the last few decades. This is a predictable result of characteristics of the typical tasks in Construction Engineering and Management, which are strongly interwoven with human factors. The specific examples of these characteristics include the ability to learn and generalize, the ability to search and optimize effectively and the ability to cope with uncertainty and ambiguity. Since CI techniques inherently have the capabilities mentioned above, they are extensively applied in the field of Construction Engineering and Management.

Subsection 1.3.1 will first review some typical applications of CI techniques in Construction Engineering and Management. Then, the following two subsections will provide an overview of two inspirational techniques of construction performance modeling. The lessons learned from the two techniques are discussed.

1.3.1 CI in Construction Engineering and Management

First, because of their capability to learn and generalize, CI techniques such as Neural Networks, Neural-Fuzzy/Fuzzy-Neural Systems and their variations with evolutionary learning, were broadly applied in the analysis, modeling and prediction problems of construction projects. For example, Sawhney and Mund [42] utilized the Adaptive Probabilistic Neural Network to help managers select an appropriate crane for construction operations. Emsley, et al, [43] applied several Neural Network models in the total construction cost prediction problem. Hegazy and Ayed [44] tried back-propagation, simplex optimization and Genetic Algorithm to develop an acceptable Neural Network, which was used "to effectively manage construction cost data and develop a parametric cost-estimating model for highway projects". Adeli and Karim [45] developed a neural dynamic model for the problem of efficiently scheduling construction projects. Cheng and Ko [46] made an effort to apply an object-oriented fuzzy neural network to several construction problems including the cost prediction of plant maintenance and the duration estimation of slurry walls. The research most related to the objective of this thesis is Sonmez and Rowings's research [47] on Neural Network based modeling of a typical construction performance indicator – construction labor productivity. Subsection 1.3.1 provides further details about this research. Finally, Boussabaine's review [48] on the applications of artificial neural networks in construction management in 1996, although a little outdated, is still of interest.

Secondly, because fuzzy sets and systems and their variations are capable of embedding expert knowledge, and dealing with uncertainty and ambiguity in human-factor intensive procedures, they are naturally applied in construction engineering and management. Ayyub [49] has provided a systems framework for fuzzy sets in Civil Engineering. This work is also a good reference review on fuzzy set applications in Civil Engineering. More particularly, Tah and Carr [50] proposed a fuzzy logic based risk assessment method for construction projects. Lam, et al, [51] applied fuzzy set theory in dealing with qualitative or linguistic variables when solving multiple-objective decision-making problems in construction financial management. Chao and Skibniewski [52] introduced a fuzzy logic based evaluation method for selection in alternative construction technologies. Robinson and Zhou's research [53] on a fuzzy expert system for design performance prediction and evaluation is a noteworthy fuzzy system based model of a specific construction performance – design performance. It will be reviewed further, and the lessons learned from it will be discussed in subsection 1.3.2.

Finally, the Genetic Algorithm as an optimization and search method is logically combined with neural networks and fuzzy systems to synthesize more powerful CI techniques for dealing with construction problems. However, the advantages of global searching and flexibility of problem representation also make GA itself a great choice for some combinational optimization problems in construction, as in the following examples. To efficiently manage space for construction facilities on high-rise buildings, Jang, et al, [54] established GA modeling assumptions to properly allocate space for facilities. Zheng, et al, [57] proposed a GA enabled technique for the problem of optimization of multi-resource leveling. Natsuaki, et al, [56] recommended a GA based approach to determine the laying sequences of construction components in bridge building; for this particular problem, an improved GA variation operator is introduced. Further applications of GA in construction engineering and management can be found in [58-60].

1.3.2 Neural Networks based Construction Labor Productivity Modeling

In 1998, Rifat Somez and J. E. Rowings provided a methodology for construction labor productivity modeling, which is a combination of the regression analysis and neural networks. First, the paper reviewed some past research that analyzed the influence of one or two factors on a specific construction performance index – labor productivity. Because of the small number of influence factors, the relationship between the impacting factors and productivity is somewhat transparent. For example, in a "factor model" developed by [61-62], the impact of crew size on productivity is clearly depicted in a curve like the one in Figure 1.5.

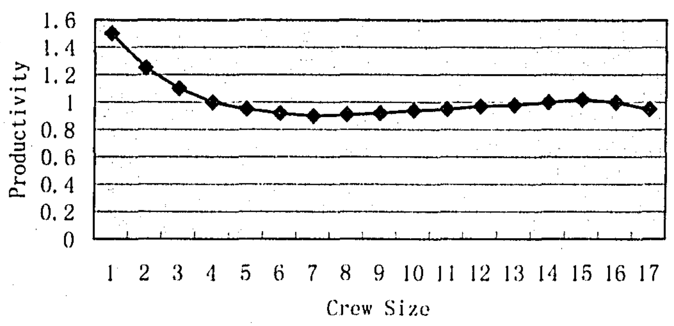


Figure 1.5 - The impact of crew size on productivity

Unfortunately, as the number of factors considered in the model increases, the transparency of the model quickly decreases. To model the complex system with a large number of influence factors, Somez and Rowings introduced a methodology consisting of fours stages. In the first stage, some basic statistics of the original data are calculated; based on these, the user selects the factors that might affect labor productivity. In the second stage, a regression model, or more specifically, a linear model is built based on the factors identified in the last stage. The regression model helps the user to determine the most significant factors for labor productivity. In the third stage, a neural network is trained, based on the selected significant factors. Finally, in the fourth stage, the neural network is validated to determine if additional factors should be added to the regression model, and in the long run, to decide if the neural network should include more significant factors to consider.

It is clear that the kernel idea of Somez and Rowings' paper is building a parsimony model for labor productivity because "productivity models including few significant factors predict better than models based on many factors without considering significance". However, although there are only a few most important factors are considered in the neural network model, the neural network applied in the paper does not offer any transparency to the user in order to analyze each factor's impact on labor productivity. For instance, the sequence of the factors' importance on the labor productivity is not available in the proposed model. Such a sequence is helpful when a problem occurs and the user must decide which factors should be carefully analyzed. This need inspired the application of a different neural network, which offers importance indices of factors' impact on construction performance (Chapter 4). The importance indices provide the construction performance model with some amount of transparency.

1.3.3 Fuzzy System based Design Performance Modeling

For design performance prediction and evaluation, Robinson and Zhou introduced a multilayered fuzzy system based methodology. The fuzzy set theory is applied because of the intrinsic subjectivity and uncertainty in factors influencing specific construction performance – design performance.

The structure of proposed the multilayered fuzzy system is illustrated in Figure 1.6. The first two layers involve input factors and the last tow layers concentrate on output factors or design performances. As depicted in Figure 1.6, each input factor in the second layer is determined by several sub-factors in the first layer; and the sub-factors in the output layers is determined by the factors in the second layer. Finally, the sub-factors in the output layers establish the output factors or design performance values in the fourth layer. To decide the factors or sub-factors in the last three layers of the model, sub-models should be built between the resulting factors and their influence factors. For example, there is a sub-model between sub-factor 1.1, 1.2, and the Input Factor 1 in the input factors layers. Robinson and Zhou suggested a method of generating fuzzy IF-THEN rules for those sub-models; in other words, they have proposed an approach to build those sub-models as fuzzy systems. In order to apply fuzzy IF-THEN rules, a linguistic description of each factor or sub-factor is required naturally. Fayek and Sun introduced a new approach to define membership functions, which are used to describe the linguistic terms.

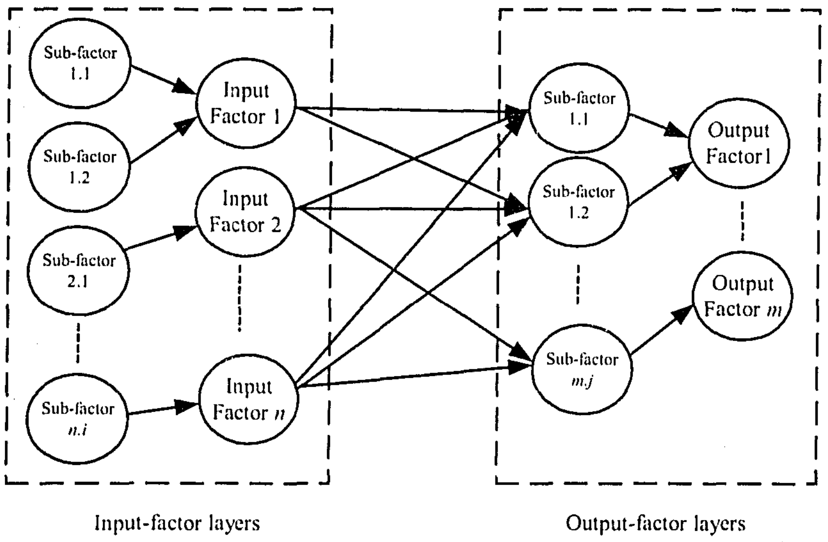


Figure 1.6 - The structure of the fuzzy model proposed by Fayek and Sun

The lesson learned from this model of design performance is twofold. On the one hand, the introduction of fuzzy sets or linguistic descriptions dramatically boosts our capability to cope with the subjectivity and uncertainty inherent in construction problems. Therefore, the framework used in this thesis also applied an information granulation step, which transforms numeric inputs to the membership values of their linguistic description. On the other hand, the fuzzy system applied in construction performance modeling still does not offer any information on the importance indices of influence factors. In addition, as mentioned in the conclusion of Robinson and Zhou's paper, "the model does not achieve a high success rate for numerical prediction". This inspired this choice in this thesis of using a neural network continually as the basic modeling technique.

1.4 Conclusion

This chapter outlines the objective of this thesis – building transparent construction performance models that assist managers to understand cause-effect relationships between a construction performance indicator and its influence factors. In order to achieve this objective, the techniques of computational intelligence are selected as the main modeling methods.

After an overview of major CI techniques and their advantages and disadvantages, the application of CI techniques in Construction Engineering and Management is reviewed. Further effort is devoted to discussing the previous research on neural networks and fuzzy systems based construction performance modeling. Two lessons are learned from this research. This first lesson is that neural networks offering importance indices of input variables should be introduced because such a model offers greater transparency to managers. The second lesson is that transparency of models can be further enhanced through information granulation that transforms numeric variables into memberships of their linguistic descriptions. Chapters 3 and 4 of this thesis elaborate on our contributions in accordance with these two lessons.

Chapter 2
The Framework of Model-based Construction Performance Diagnosis

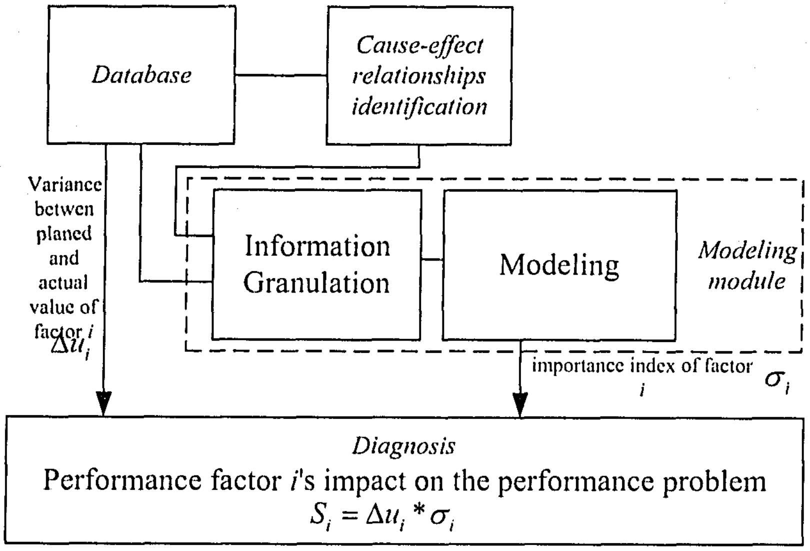
Managers of construction projects encounter a typical construction performance diagnosis problem when the actual performance is considerably different from the anticipated one. They are then required to spend considerable time on studying all the possible causes of the discrepancy, in order to avoid them in the future. Obviously, in the process of diagnosis, computer tools, which automatically identify the possible reasons for the performance problem, and quantify their impacts, could be of great assistance to the managers. This chapter introduces an available framework of such a computer tool, which implements model based construction performance analysis. A high-quality construction performance model is required for the success of this framework. In Section 2.1 the overall picture of the diagnosis framework is described. The specific requirements or desired features of a transparent construction performance model in the framework of diagnosis are explained in Section 2.2

2.1 The Framework of Diagnosis

Dissanayake proposed this framework of construction performance model based diagnosis in [63]. The four major modules and their relationships in the framework are illustrated in Figure 2.1.

The first module is the ***cause-effect relationships identification*** module. In this module, managers first select an activity on which they wish to conduct the diagnosis; then they provide all the possible factors that may affect the construction performance considered in this activity. In other words, this module builds the original cause-effect model for a specific activity. When the managers consider an identical performance problem in the same activity of construction on another occasion, this module is able to list all the possible causes in the original cause-effect model.

The second module in the diagnosis framework is a ***database*** program. This program stores and manages all the historical data according to the causal relationships identified in previous module. In addition, the expert knowledge about the membership functions of linguistic terms depicting the possible factors is also saved in this database.



Database
Cause-effect relationships identification
Variance betwen planed and actual value of factor i
Δu_i
Information Granulation
Modeling
Modeling module
importance index of factor i
σ_i
Diagnosis
Performance factor i's impact on the performance problem
$S_i = \Delta u_i * \sigma_i$

Figure 2.1 - The Framework of Diagnosis

The third module is the kernel of the framework – the ***modeling module***; this module includes two steps. In the first step, the historical data is granulated by changing the original value to several membership values of its linguistic terms. For example, for the factor "Temperature", this step may change the value of temperature to values of 3 linguistic terms: "Low Temperature", "Normal Temperature", and "High Temperature". At this information granulation step, the definition of membership functions is not a trivial. Although asking the experts or managers to define the membership function is not impossible, the automatic method is more viable. Chapter 4 will introduce a clustering based algorithm that generates membership functions from historical data.

The second step of the modeling module is to build a model between the inputs – the linguistic terms of influence factors, and the output – the construction performance considered in a specific activity. The model built in this step will not only provide a accurate input-output mapping of construction performance, it will also offer indicators about the importance of each input on the output, namely the quantitative importance indices. In Chapter 3, a General Regression Neural Network based model holding such characteristics will be proposed. In Chapter 4, the proposed neural network will be assessed on granulated data.

The final module in the framework of diagnosis is the ***diagnosis*** module. In this module, the planned value of construction performance (output) is compared with the actual value. If there is a considerable discrepancy between the two values, managers may wish to determine the chief reasons for the discrepancy. So initially the diagnosis module computes the variance between the planned values of influence factors and their actual values. Assume the planned values are u_i, and the actual values are v_i (i=1,2,...,n; n is the number of influence factors), the variances $\Delta u_i = | u_i - v_i |$. Then the variances are multiplied with the importance indices of performance factors - σ_i, which are generated from the modeling module (σ_i is the importance of the i^{th} influence factor). The resulting values $S_i = \Delta u_i * \sigma_i$ are the indicators of the performance factors' influence in the discovered performance problem - namely the discrepancy between the actual and planned values of the performance indicator considered in current construction project.

2.2 The Desired Features of a Transparent Model

In a specific activity in construction, given the considered performance indicator is p, and there are n influence factors x_1, x_2, ..., x_n related to the performance, the framework described in last section requires the model of the mapping between inputs x_1, x_2, ..., x_n and output p to be accurate and transparent.

The accuracy of the construction performance model is a basic requirement. Whenever managers apply the model in the diagnostic procedure, the first requirement is the model should accurately capture the nature of the system; otherwise it could not be reliably applied to the prediction, analysis, or control of the system. This is the reason the researchers in recent decades have recommended so many different techniques to model construction performance. From the linear regression model to the neural network model, one of the key goals in construction performance modeling is to achieve better accuracy.

The desired features related to a transparency model are represented in the two steps of the modeling module of the diagnosis framework. First, the transformation from an input variable to its linguistic description in the information granulation step increases the transparency of the performance model. For instance, compared with a model considering only a variable such "Temperature" as an influence factor of construction performance, considering the linguistic terms of "Temperature", such as "Low Temperature", "Normal Temperature", and "High Temperature" as influence factors is more natural and useful to managers. In particular, for a variable such as "Skill Level", where quantitative values may not be accurate, managers are inclined to utilize linguistic descriptions, such as "High Skill Level" or "Low Skill Level". Additionally, with linguistic terms as inputs to the model, we can draw clearer conclusions, such as "High Temperature is very important to construction performance", instead of "Temperature is very important to construction". In conclusion, the transformation from a variable to its linguistic terms, as a typical information granulation procedure, helps managers to more easily deal with the imprecision, uncertainty and ambiguity involved in construction. Therefore, the first desired feature of a transparent construction performance model is the inclusion of information granulation.

The second step in the modeling module of the proposed framework represents another desired feature for the transparent model – defining importance index of each input. This means that the model of construction performance cannot be a naïve black box that simply maps the inputs to the outputs. The model should offer some transparency, or at least, should give some hints on the measurement of each input's impact on the output. Therefore, the models built in modeling module should provide more helpful conclusions, such as "High

Skill Level holds the importance of 10 to construction performance and Normal Temperature holds the importance of 2 to construction performance". Such quantitatively-based conclusions are extremely valuable when managers attempt to analyze a construction performance problem. Therefore, the second desired feature of a transparency model for the construction performance is that the model should offer some importance indices to indicate the inputs' influences on the output of the model.

2.3 Conclusion

This chapter introduced the framework of construction performance diagnosis proposed by Dissanayake. As a model based diagnosis framework, two specific requirements or desired features for the transparent construction performance model - information granulation and importance index - are elaborated.

The following two chapters of the thesis are investigations on the two desired features of the construction performance model applied in the diagnosis framework. Chapter 3 introduces the General Regression Neural Network learned by genetic algorithms as the basic modeling techniques. Compared with traditional techniques, this model not only has a more accurate prediction rate, it fulfills the requirement of providing the importance index of each input variable. Chapter 4 proposes a new fuzzy C-means clustering based membership function generation method. This method helps managers to automatically generate membership functions of linguistic terms of potential impacting factors. With the information granules formed by membership functions, the transparency of the construction performance model is enhanced.

Chapter 3
The Model with Importance Indices

As shown in Chapter 2, in addition to the necessity for accurate approximation capability, there are two desired features for a useful model of construction performance applied in the framework of construction performance diagnosis. The first desired feature is that the model should offer some importance indices marking the influence of each input factor on the output of model, i.e., construction performance. The second is the necessity for information granulation in order to deal with the linguistic terms used by managers to describe their knowledge about the influence factors, and resulting construction performance.

This chapter focuses on the models that offer the first feature. The necessity for information granulation and the proposed fuzzy information granulation method is discussed in Chapter 4. In this chapter, Section 3.1 suggests two computational intelligent modeling techniques, which provide importance indices of influence factors on the resulting construction performance: a fuzzy neural network - OR/AND neuron, and a general neural network - General Regression Neural Network (GRNN). The OR/AND neuron is learned though a gradient-based algorithm, while the proposed GRNN updates separate smoothing parameters for each dimension through a real-coded genetic algorithm. Section 3.2 compares the two proposed modeling techniques. Although both OR/AND neuron and GRNN provide indices of importance, experiments on real data collected in construction sites show that GRNN leads to higher accuracy in approximating of the mapping between inputs and output. The genetic learning based GRNN is therefore selected as our final modeling technique to capture the mapping between influence factors and resulting construction performance.

3.1 OR/AND Neuron and GRNN

Among all the models that present importance indices of inputs' influence on output, the most widely applied ones are the linear regression models, OR/AND neuron and kernel based models holding adaptive parameters for each dimension of inputs. Because of high nonlinearity in real construction problems, linear analysis is not suitable for our task. This section discusses two other competitive models - the OR/AND neuron and the kernel based model, General Regression Neural Network (GRNN).

3.1.1 OR/AND Neuron

The OR/AND neuron [64] is a three layered fuzzy neural network (FNN) consisting of two basic logic-based neurons named AND neuron and OR neuron. Both AND and OR neurons are multivariable nonlinear transformations between unit hypercubes. As illustrated in Fig 3.1, the OR/AND neuron accepts "*n*" inputs, transforms respective inputs AND-wise (AND neuron) and OR-wise (OR neuron) in two separate computing channels, and merges the results through an OR neuron.

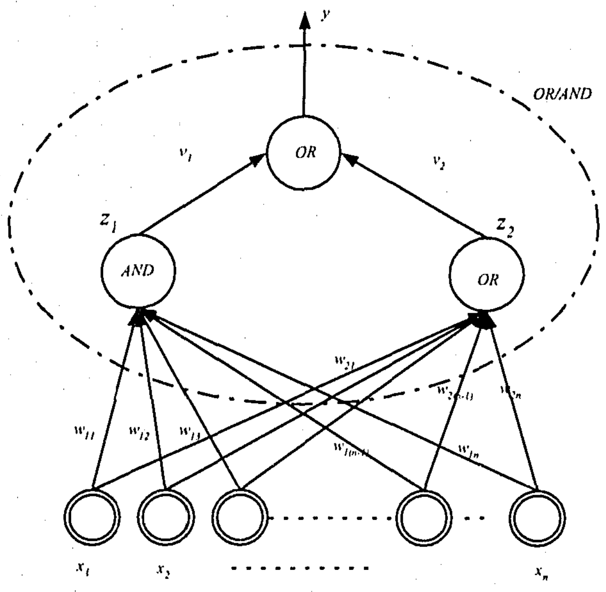


Figure 3.1 - The structure of OR/AND neuron

The AND neuron, as illustrated in Fig 3.2, first individually combines the inputs $X=[x_1, x_2, \ldots, x_n]$ and connections $\mathbf{w}_1=[w_{11}, w_{12}, \ldots, w_{1n}]$ through s-norms, and then aggregates these results AND-wise by applying t-norms.

$$z_1 = \mathrm{AND}(X;\mathbf{w}_1),$$

i.e.,

$$z_1 = \mathop{T}_{i=1}^{n}(x_i s w_{1i})$$

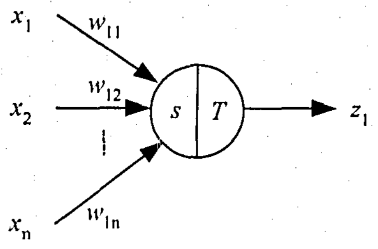


Figure 3.2 - The structure of AND neuron

The OR neuron is dual to the structure of AND neuron, namely

$$z_2 = \mathrm{OR}(X;\mathbf{w}_2),$$

i.e.,

$$z_2 = \mathop{S}_{i=1}^{n}(x_i t w_{2i}),$$

in which, the $X=[x_1, x_2, \ldots, x_n]$ are inputs and $\mathbf{w}_2=[w_{21}, w_{22}, \ldots, w_{2n}]$ are connections, c.f. Figure 3.3.

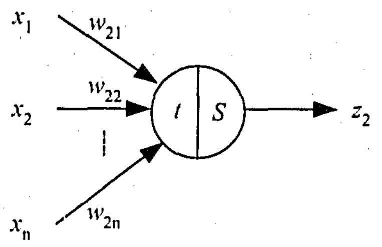


Figure 3.3 - The structure of OR neuron

As depicted in Figure 3.1, the OR/AND neuron combines the output of an AND neuron and the output of an OR neuron via an OR neuron with the connections v=[v_1, v_2] in the final output layer. Therefore, the mapping described by OR/AND neuron is,

$$y = \text{OR/AND}(X; \mathbf{w}, \mathbf{v}), \text{ in which } \mathbf{w}=[\mathbf{w}_1, \mathbf{w}_2]$$

Rewriting the previous expression in coordinate-wise manner, we obtain:

$$y = (z_1 t v_1) s (z_2 t v_2)$$

In extreme cases, if v_1=1 and v_2=0, the OR/AND neuron is the same as the pure AND neuron. And if v_1=0 and v_2=1, the OR/AND neuron functions as a pure OR neuron.

The *s*-norms and *t*-norms applied in the AND and OR neurons are triangular norms representing the logic operations on fuzzy sets. For example, if the above *t* and *s*-norms are realized as product and probabilistic-sum operators governed by expressions,

$$atb = ab \text{ ; } a,\ b \in [0,1];$$

$$asb = 1-(1-a)(1-b) = a+b-ab \text{ ; } a,b \in [0,1],$$

the input-output mapping represented by OR/AND neuron will be:

$$y = z_1 v_1 + z_2 v_2 - z_1 v_1 z_2 v_2,$$

in which,

$$z_1 = \prod_{i=1}^{n}(x_i + w_{1i} - x_i w_{1i}),\ \ z_2 = 1 - \prod_{i=1}^{n}(1 - x_i w_{2i}).$$

If the above t and s norms are realized as min and max operators,

$$atb = \min(a,b)\,;\ a,\ b \in [0,1]$$

$$asb = \max(a,b)\,;\ a,\ b \in [0,1],$$

the input-output mapping of OR/AND neuron will be:

$$y = \max(\min(z_1, v_1), \min(z_2, v_2)),$$

where

$$z_1 = \min_{i=1}^{n}(\max(x_i, w_{1i})),\ \ z_2 = \max_{i=1}^{n}(\min(x_i, w_{2i})).$$

An important property of OR/AND neuron is the role of connections. As concluded in [2], "Because of the boundary conditions of triangular norms, the higher values for connections in OR neuron emphasize that the corresponding inputs exert a stronger influence on the neuron's output." On the other hand, the higher values of connections in AND neuron emphasize that the corresponding inputs exert a lesser influence on the neuron's output. In particular, if the weight of AND neuron w_{1i} is close to 1, the influence of x_1 is almost negligible, and if the weight w_{2i} of OR neuron is close to 0, the influence of x_2 is almost negligible as well. This property in some extent provides OR/AND the capability stated as being required in the beginning of this chapter, that is, the connections in the AND/OR neuron are importance indices representing the influence of inputs on the output. A detailed example of such importance indices is put forward in the numeric experiment below.

Despite the well defined semantic of OR/AND neuron, a serious problem arises from the neuron's structure. That is, the OR/AND only realizes an **in** mapping between the unit hypercubes. Or, in other words, the output of OR/AND neuron cannot cover all the values in the interval of [0, 1]. Specifically, the values of $y = (z_1 t v_1) s (z_2 t v_2)$ are included only in the interval of $[0, v_1 s v_2]$. This is the natural result of boundary conditions of *t* and *s*-norms:

$$z_1 t v_1 \le 1 t v_1 = v_1 , \quad z_2 t v_2 \le 1 t v_2 = v_2 ;$$

therefore,

$$y = (z_1 t v_1) s (z_2 t v_2) \le v_1 s v_2 .$$

To alleviate the abovementioned limitation and enhance the computational capability of OR/AND neuron, a nonlinear sigmoid element is added to the output layer of OR/AND neuron [64]. The augmented OR/AND neuron is represented in Figure 3.4.

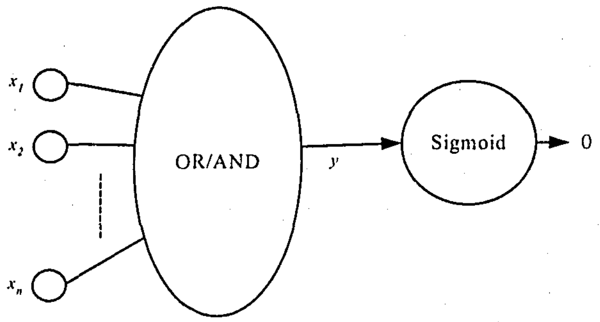


Figure 3.4 - The augmented OR/AND neuron with a nonlinear processing element

The sigmoid element placed in the output layer is a monotonic transformation $\Psi : [0,1] \rightarrow [0,1]$,

$$o = \Psi(y) = \frac{1}{1 + \exp(-(y - m)\sigma)} ,$$

in which the m and σ are two tunable parameters. By adjusting these two parameters, the OR/AND neuron's output in the interval of $[0, v_1 s v_2]$ can be extended to interval [0, 1].

3.1.1.1 Learning/Optimization with Gradient Descending

Assume there are N datasets available to OR/AND neuron's learning: $[x_1(r),...,x_n(r),o_r]$, r=1, ..., N, in which o_r is target and $\hat{o}_r$ is OR/AND neuron's output with respect to inputs $[x_1(r),...,x_n(r)]$.The learning algorithm of the proposed augmented OR/AND neuron adjusts its connections **w**, **v** through gradient descending [65] to minimize the performance index Q, which is the difference between the expected targets and the neuron's real outputs,

$$Q=\sum_{r=1}^{N}(o_r-\hat{o}_r)^2$$

More specifically, the learning algorithm updates OR/AND neuron's connections $\mathbf{w}_1=[w_{11}, w_{12}, \ldots, w_{1n}]$, $\mathbf{w}_2=[w_{21}, w_{22}, \ldots, w_{2n}]$ and $\mathbf{v}=[v_1, v_2]$ through gradient descending as:

$$w_{1i}=w_{1i}-\frac{1}{2}\alpha\frac{\partial Q}{\partial w_{1i}},\quad w_{2i}=w_{2i}-\frac{1}{2}\alpha\frac{\partial Q}{\partial w_{2i}}$$

$$v_1=v_1-\frac{1}{2}\alpha\frac{\partial Q}{\partial v_1},\quad v_2=v_2-\frac{1}{2}\alpha\frac{\partial Q}{\partial v_2},$$

$$m=m-\frac{1}{2}\alpha\frac{\partial Q}{\partial m},\quad \sigma=\sigma-\frac{1}{2}\alpha\frac{\partial Q}{\partial \sigma},$$

where $\alpha\in[0, 1]$ is the learning rate. The initial weights values can be set by experts or managers, or be randomly generated.

3.1.1.2 Numerical Experiment

The example is a multi-objective decision-making problem provided by [64]. It determines a global preference of several methods to make the same product. Five different methods are evaluated with respect to four criteria, namely, reliability, portability, price, and reusability.

The final datasets we obtain are listed in Table 3.1:

Inputs				Output
1.00	0.24	0.3	0.57	0.70
0.53	1.00	0.09	0.13	0.50
0.30	0.20	1.00	0.34	1.00
0.15	0.46	0.55	0.19	0.20
0.13	0.55	0.17	1.00	0.36

Table 3.1 - Datasets obtained in a decision-making problem

The datasets are selected as the training data for an augmented OR/AND neuron. Because of the small number of the training data or the simplicity of the approximation problem, choosing a small learning rate $\alpha = 0.01$ and a small number of iteration as 300 is enough to get a acceptable approximation rate. The resulting values of performance index Q in the successive epochs are visualized in Figure 3.5. The connections in the OR/AND neuron, say v, w_1, w_2 and the parameters of the sigmoid function (m, σ) are randomly initialized.

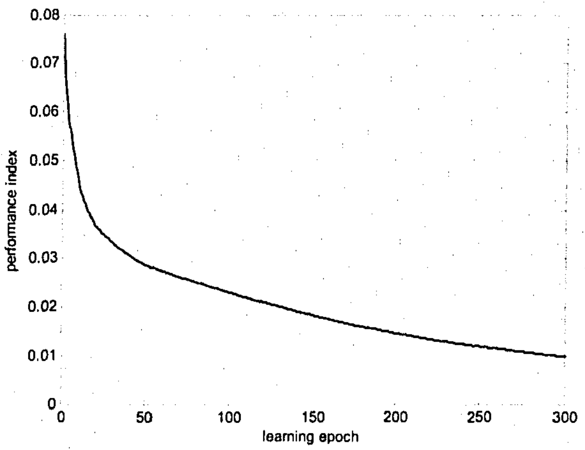


Figure 3.5 - The value of performance index Q in successive learning epochs

The m-σ space in Figure 3.6 depicts the changing of parameters of the sigmoid element in the output layer of the augmented OR/AND neuron. The resulting m=0.81 and σ=8.62.

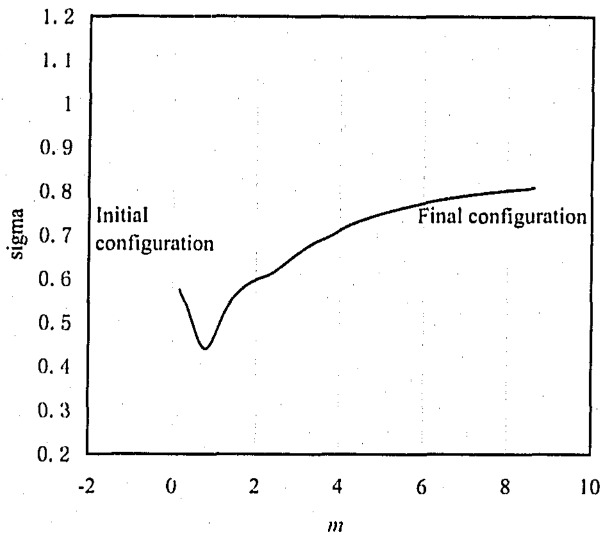


Figure 3.6 - m-σ space during the learning of OR/AND neuron

Finally, the resulting connections of the OR neuron in the output layer are

v=[0 1],

and the resulting connections of AND and OR neuron in the hidden layer are:

w_1=[0.88 0.36 0.31 0.10] (In AND neuron)

w_2=[0.79 0.57 1 0.47] (In OR neuron)

Figure 3.7 demonstrates that the resulting OR/AND neuron approximates the training data very well.

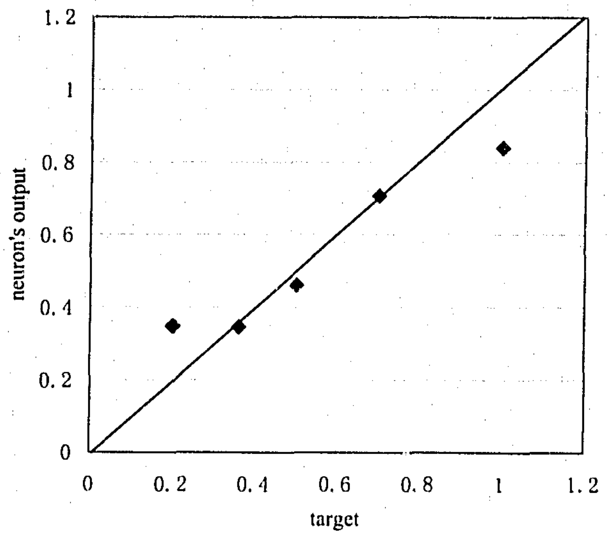


Figure 3.7 - Targets vs. OR/AND neuron's outputs

Because of v=[0 1] and the OR neuron's property, the OR neuron in the hidden layer has much more evident impact on the output. In fact, because the v_2=1, the OR/AND neuron becomes a pure OR neural. Then, based on the hidden OR neuron's connections w_2=[0.79 0.57 1 0.47], different criteria or inputs show distinct impact on the output. More specifically, the third criterion and the first criterion demonstrate more impact on the output of the hidden OR neuron and in the long run, exert more influence on the OR/AND neuron's output. It is worth mentioning that the conclusion on the importance of the third and first criterion here tallies with the one inferred in [64], although the connections that finally result are clearly different.

3.1.2 GRNN

General Regression Neural Network (GRNN) comes from the statistical regression model named kernel regression. It was first introduced as a new topology of neural network in [66].

3.1.2.1 Statistical Foundation of GRNN

Consider the nonlinear regression model in which y is the dependent scalar variable or output on the independent vector or inputs $\mathbf{X}=[x^1, x^2, \ldots x^m]$. We can formulate the relation between inputs and output as

$$y = f(\mathbf{X}) + e,$$

where e is the noise with zero mean value given any realization of **X**.

Assume the joint continuous probability density function of the random vector variable [**X**, y] is $f_{Xy}(\mathbf{X}, y)$; the conditional probability density function of y on the variable **X** is then given by

$$g(y|\mathbf{X}) = f_{Xy}(\mathbf{X}, y) / \int_{-\infty}^{+\infty} f_{Xy}(\mathbf{X}, y) \qquad (1)$$

In statistical learning theory, it is proved that $f(\mathbf{X})$ is equal to the conditional mean of output y given the inputs **X**, that is,

$$f(\mathbf{X}) = E(y|\mathbf{X}) = \int_{-\infty}^{+\infty} y g(y|\mathbf{X}) dy \qquad (2)$$

Combining the Equation (1) and (2), the input and output mapping, or the regression model, can be formulated as

$$f(\mathbf{X}) = \frac{\int_{-\infty}^{+\infty} y f_{Xy}(\mathbf{X}, y)}{\int_{-\infty}^{+\infty} f_{Xy}(\mathbf{X}, y)} \qquad (3)$$

Till now, the joint density function $f_{Xy}(\mathbf{X}, y)$ is assumed be to be known. However, we have only a sample of observations of **X** and y at most of times. With the available observations as training samples, the $f_{Xy}(\mathbf{X}, y)$ can be estimated through Parzen's method [67].

Given N observations of independent random variable X are $X_1, X_2, \ldots, X_N$, Parzen's density estimate of the probability density function of X is

$$f_X(X) = \frac{1}{NC_X}\sum_{i=1}^{N} K\left(\frac{D(X - X_i)}{h}\right),$$

in which the function K is the kernel function, and h is the smoothing factor. $D(X - X_i)$ indicates the distance between X and X_i. The constant C_X is the normalizing constant to ensure that the density function integrates to unity.

Similarly, the joint density function $f_{Xy}(\mathbf{X}, y)$ can be estimated by the Parzen estimator as

$$f_{Xy}(\mathbf{X}, y) = \frac{1}{nC_X C_y}\sum_{i=1}^{n} K\left(\frac{D(\mathbf{X} - \mathbf{X}_i)}{h}\right) K\left(\frac{D(y - y_i)}{h}\right), \quad (4)$$

in which C_X and C_y are the normalizing constants to ensure that the density function integrates to unity, and $[\mathbf{X}_i\ y_i]$ (i=1,2, ..., n) are the n observations of random variable $[\mathbf{X}\ y]$. Replacing (4) into (3), the mapping between the inputs and output can be estimated as

$$f(\mathbf{X}) = \frac{\int_{-\infty}^{+\infty} y \frac{1}{nC_X C_y}\sum_{i=1}^{n} K\left(\frac{D(\mathbf{X} - \mathbf{X}_i)}{h}\right) K\left(\frac{D(y - y_i)}{h}\right)}{\int_{-\infty}^{+\infty} \frac{1}{nC_X C_y}\sum_{i=1}^{n} K\left(\frac{D(\mathbf{X} - \mathbf{X}_i)}{h}\right) K\left(\frac{D(y - y_i)}{h}\right)}$$

$$= \frac{\frac{1}{nC_X}\sum_{i=1}^{n} K\left(\frac{D(\mathbf{X} - \mathbf{X}_i)}{h}\right) \frac{1}{C_y}\int_{-\infty}^{+\infty} y K\left(\frac{D(y - y_i)}{h}\right)}{\frac{1}{nC_X}\sum_{i=1}^{n} K\left(\frac{D(\mathbf{X} - \mathbf{X}_i)}{h}\right) \frac{1}{C_y}\int_{-\infty}^{+\infty} K\left(\frac{D(y - y_i)}{h}\right)}$$

$$= \frac{\sum_{i=1}^{n} K(\frac{D(\mathbf{X}-\mathbf{X}_i)}{h})\mathrm{y}_i}{\sum_{i=1}^{n} K(\frac{D(\mathbf{X}-\mathbf{X}_i)}{h})} \qquad (5)$$

Applying the most widely used Gaussian function below as the kernel function K in Equation (5),

$$K(\frac{D(\mathbf{X}-\mathbf{X}_i)}{h}) = \exp(\frac{-D(\mathbf{X}-\mathbf{X}_i)}{h})$$

the final estimator of $f(\mathbf{X})$ is:

$$f(\mathbf{X}) = \frac{\sum_{i=1}^{n} \exp(\frac{-D(\mathbf{X}-\mathbf{X}_i)}{h})\mathrm{y}_i}{\sum_{i=1}^{n} \exp(\frac{-D(\mathbf{X}-\mathbf{X}_i)}{h})}. \qquad (6)$$

In conclusion, the estimate of output y = $f(\mathbf{X})$ based on input **X** can be explained as the weighted average of all the observations of y, i.e., $y_1, y_2, \ldots, y_n$.

3.1.2.2 Architecture of GRNN

We can reformulate the mapping represented in Equation (6) into a neural network shown in Figure 3.8. The proposed neural network topology is referred to as General Regression Neural Network (GRNN).

There are four layers in the architecture of GRNN. The first layer is the inputs layer holding m neurons (it is the same as the number of inputs for the system). The input layer simply forwards the inputs to the radial basis layer or kernel layer. The kernel layer possesses the same number of neurons as the number of training examples. The ith neuron in the kernel layer calculates the kernel function $\exp(-D(\mathbf{X}-\mathbf{X}_i)/h)$, based on the training example $\mathbf{X}_i$ and the inputs from the input layer $\mathbf{X}=[x^1, x^2, \ldots, x^m]$. The third layer of GRNN is summation layer that contains only two neurons. The first summation neuron connects all the neurons in the kernel layer with the i^{th} connection weighting y_i. Therefore, it outputs the value of $\sum_{i=1}^{n} \exp(-D(\mathbf{X}-\mathbf{X}_i)/h)\mathrm{y}_i$. The second summation neuron connects all the neurons in the kernel layer with all connections weighting 1. Therefore, the second neuron outputs the value of $\sum_{i=1}^{n} \exp(-D(\mathbf{X}-\mathbf{X}_i)/h)$. Finally, the output layer keeps one neuron that divides the first summation neuron's output by the second summation neuron's output. In other words, the output layer computes the output through Equation (6).

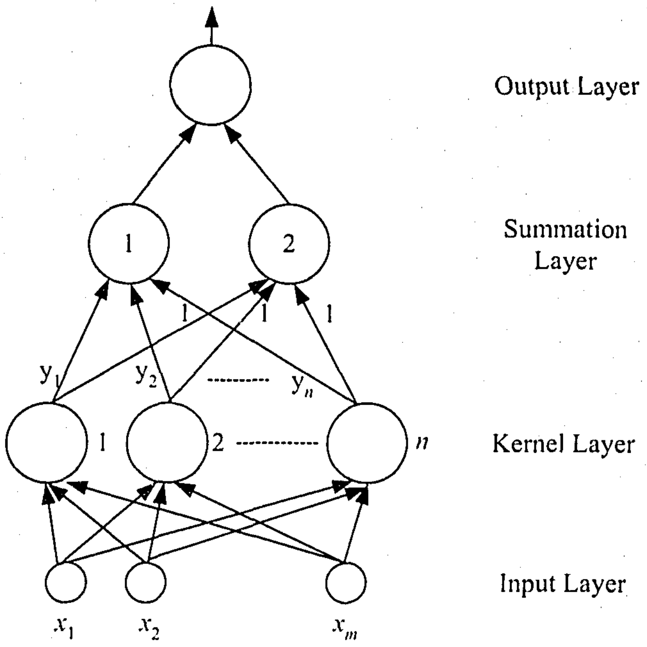


Figure 3.8 - The architecture of GRNN

3.1.2.3 One Pass Learning of GRNN

It is noticeable that there are no adjustable connections in the proposed architecture of GRNN. Given n training datasets, a GRNN holding n hidden neurons in the kernel layer is learned in one pass. The only tunable parameter of GRNN is the smoothing factor h in the kernel function.

As concluded in [66], the smaller the smoothing factor h, the better the estimated function $f(\mathbf{X})$ fits the training data. In extreme cases, as the smoothing factor goes to 0, the estimated function will fully fit the training data. Take the difference between the output of training data and the GRNN's output, based on the corresponding inputs in training data as the performance index,

$$Q = \sum_{i=1}^{n} (y_i - f(\mathbf{X}_i))^2, \quad ([\mathbf{X}_i\ y_i]\ (i=1,2, \ldots, n) \text{ are the } n \text{ training data })$$

The previous conclusion means that the performance index Q will converge to 0 as the smoothing factor goes to 0. However, although the small smoothing factor leads to a small Q or a good approximation of training data, the generalization capability of GRNN may be deteriorated, i.e., the over-fitting problem may emerge. Therefore, to balance the approximation rate and generalization capability, the value of smoothing factor h should be carefully selected.

If we take the GRNNs with different smoothing factors as different models for the same system, the selection of best smoothing factor is changed to a model selection problem. This problem can be solved by traditional model selection methods [68], such as hold-out testing. In particular, the simplest hold-out testing method is dividing the original training data into two datasets: training data and validation data. The GRNN is trained in one pass based on the training data, and the optimal smoothing factor selected is the one that makes the resulting GRNN possess the best performance index Q on the validation data. [66] has proposed a more general hold-out testing: a leave-one-out cross validation method to estimate the best smoothing factor. In this method, if the smoothing factor is h and the n training datasets are $[\mathbf{X}_i\ y_i]$ $(i=1,2, \ldots, n)$, the performance index $Q(h)$ based on leave-one-out cross validation is defined as

$$Q(h) = \sum_{i=1}^{n} (y_i - f_i(\mathbf{X}_i))^2, \quad (7)$$

In Equation (7), f_i is the mapping represented by the GRNN built through smoothing factor h and the training datasets, leaving the i^{th} dataset out. The best smoothing factor h is then selected as the one that minimizes the performance index $Q(h)$.

3.1.2.4 Numeric Experiment

To demonstrate the approximation capability of GRNN, the previously discussed one-pass learning and leave-one-out cross validation is applied to the benchmark problem of M-G time series prediction [32].

The M-G time series is a chaotic time series generated by the Mackey-Glass (MG) time-delay differential equation as below,

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

Assuming $x(0) = 1.2$, $\tau = 17$ and $x(t) = 0$ as $t<0$, we obtain the time series values at integer points through the fourth-order Runge-Kutta method. The resulting time series is plotted in Figure 3.9:

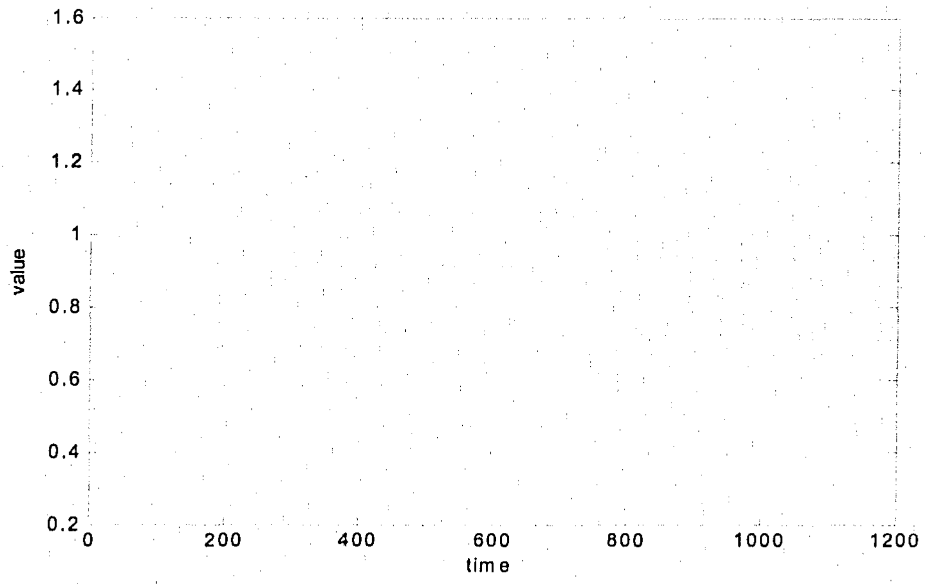


Figure 3.9 - M-G time series

In the M-G time series prediction problem, we hope to derive $x(t+6)$ from $x(t-18)$,

$x(t\text{-}12)$, $x(t\text{-}6)$ and $x(t)$, i.e., we assume

$$x(t+6)=f(x(t-18), x(t-12), x(t-6), x(t)) \qquad (8)$$

Taking inputs $\mathbf{X}=[x(t-18), x(t-12), x(t-6), x(t)]$ and output $y=x(t+6)$, we generate 100 datasets based on t=118 to 218 as the training datasets. The following 100 datasets based on t=219 to 318 are generated as the testing data.

Using the Euclidian distance as the distance function D in Equation (6), we apply the GRNN on the training datasets and select the best smoothing factor h in the range of [0.01, 0.1] through the hold-one-out cross validation. Figure 3.10 demonstrates the best smoothing factor in range [0.01, 0.1] is 0.03:

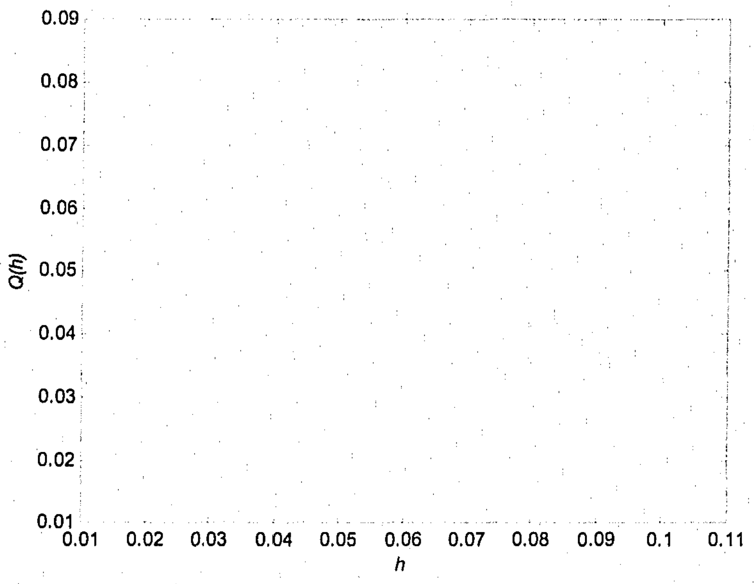
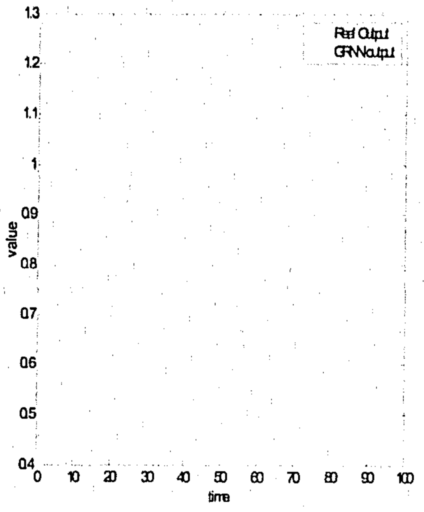
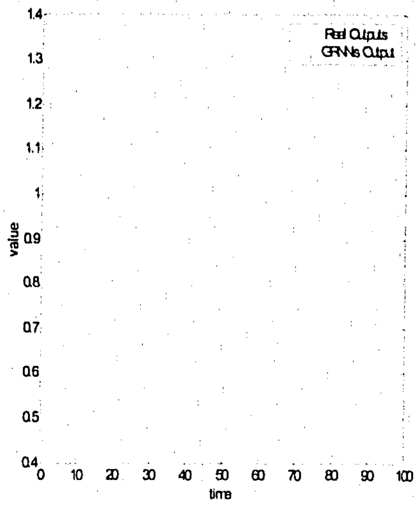


Figure 3.10 - Performance index versus the smoothing factor h

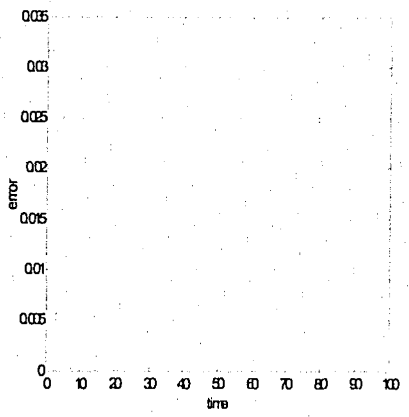


(a) on training data

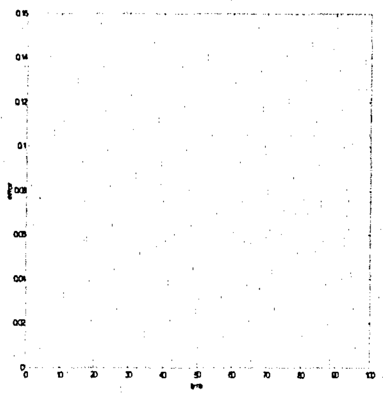


(b) on testing data

Figure 3.11 - GRNN's outputs vs Real Ouputs



(a) approximation error



(b) testing error

Figure 3.12 - The approximation error and testing error between real outputs and GRNN's outputs

Based on this smoothing factor, the difference between the resulting GRNN's outputs and the training data outputs is illustrated in Figure 3.11 (a). The error between the predict value and the real value is plotted in Figure 3.12 (a). From Figure 3.11 (a) and Figure 3.12

(a), it is easy to see that the proposed GRNN approximates the mapping *f* between the inputs and output in equation (8) very well. Figure 3.11 (b) and Figure 3.12 (b) demonstrate that the proposed GRNN model's capability of prediction on testing data is also acceptable.

3.1.3 GRNN with Adaptive Kernel Shape and Its GA based Learning

Although GRNN provides a powerful tool to approximate multivariable functions or mappings, the one and only adjustable parameter limits its capability. More specifically, "GRNN cannot ignore irrelevant inputs without major modifications to the basic algorithm. So GRNN is not likely to be the top choice if you have more than 5 or 6 nonredundant inputs."[69] Another more important problem is that it does not present any index to describe the different inputs' impacts on output.

To solve these limitations of GRNN, Specht and Romsdahl [70] introduced the adaptive GRNN. This is GRNN with the adaptive shape of a kernel. Recall the applied kernel function till now:

$$K(\frac{D(\mathbf{X}-\mathbf{X}_i)}{h}) = \exp(\frac{-D(\mathbf{X}-\mathbf{X}_i)}{h})$$

The shape of the kernel function is fully controlled by the smoothing factor h. The larger the smoothing factor, the smoother the shape of kernel function will be. If additional parameters are implanted in this function, the shape of the kernel function will be more elastic and the GRNN will possess a more flexible approximation capability. In particular, for a system with m inputs, m local smoothing factors are included in the distance computation function D, i.e.,

$$D(X-X_i) = \sum_{j=1}^{m} \sigma_j (x_j - x_{ij})^2 , \qquad (9)$$

In Equation (9), X=[x_1,...,x_n], X_i=[x_{i1},...,x_{in}] and σ_j (j=1,..., m) are referred to as local smoothing factors. To avoid confusion, the smoothing factor h is now called a global smoothing factor.

The introduction of local smoothing factors gives GRNN the capability to consider the different impacts of various inputs and solves the problem of "cannot ignore irrelevant inputs" [69], providing GRNN stronger approximation capability. In fact, such local smoothing factors are widely used in most of kernel based regression methods [71-72], of which GRNN is typical. In Equation (9), because different dimension of input is multiplied by σ_j (j=1,..., m) separately, the smaller sigma value emphasizes the lesser impact or importance of the corresponding input variable on the distance value D (the smaller value of $\sigma_j(x_j - x_{ij})^2$), and furthermore, the lesser importance to the kernel function's value and the final output of GRNN. The adaptive GRNN's local smoothing factors, therefore, can be taken as the importance indices of inputs. The smaller value of the local smoothing factor indicates a smaller influence of the corresponding input on the output. In extreme cases, if the local smoothing factor is close to 0, the corresponding input is negligible. In other words, the corresponding input can be removed from the input feature set, just as concluded in [70] "Adaptation of kernel shapes provides... a feature selection capability and improved accuracy".

Additional tunable parameters, such as local smoothing factors changed only the computation of kernel function in the second layer of GRNN; they do not change the architecture and connections of GRNN. The GRNN continually holds the one-pass learning procedure. However, the determination of both global and local smoothing factors is a more difficult problem. Specht [70] has suggested a simple adapting procedure that perturbs each smoothing factor a small amount and accepts the perturbation that most improves the performance index. The perturbation is continued until some specific criteria are satisfied. This method is a random search approach. Masters [73] has proposed a gradient-based method, in which the partial derivatives of performance index are derived and the best smoothing factors are searched via gradient descending.

This subsection proposes a new approach, which applies the real-coded genetic algorithm (RCGA) to find optimal global and local smoothing factors of adaptive GRNN.

First, to solve the problem using a genetic algorithm, the searching target is defined. As previously discussed, the GRNN's performance based on leave-one-out cross validation can be selected as the target to be minimized. However, the genetic algorithm holds a population of individuals and will evolve a number of generations; in each generation each individual must calculate the performance index based on leave-one-out cross validation – e.g., Equation (7) – once, which will be a huge computing burden for the algorithm. In addition, when the number of training datasets is large, the calculation of Equation (7) itself is already a processor time-consuming job. Therefore, the proposed approach discards the leave-one-out cross validation to select the best smoothing factors in the genetic algorithm; simple hold-out testing is applied. That is, the original training data is divided into two datasets: the training data and the validation data. The GRNN is then trained in one pass, based on the training data. Finally, the genetic algorithm is applied to find the smoothing factors minimizing the performance index Q on the validation data. More specifically, if there are n_1 training data [$\mathbf{T}_i$ yt_i] (i=1,2, ..., n_1) and n_2 validation data [$\mathbf{X}_i$ y_i] (i=1,2, ..., n_2), the GRNN will be built based on training data [$\mathbf{T}_i$ yt_i] and the GA will be applied to find smoothing factors that minimize Q, expressed as

$$Q = \sum_{i=1}^{n_2} (y_i - f(\mathbf{X}_i))^2, \quad (10)$$

(In Equation (10), f is the mapping function generated by GRNN based on n_1 training data).

Details of the real-coded genetic algorithm for the adaptive GRNN is organized according to the key steps of GA discussed in Chapter 1

Representation

Assume there are m inputs in the system, so that there are m local smoothing factors and one smoothing factor to be optimized. In the real coded genetic algorithm, each individual is represented by a vector of real numbers. In this problem, the real vector with length of m+1 is applied. The first entry in the vector is global smoothing factor h, and the next m entries are local smoothing factors σ_j (j=1,..., m), as shown in Figure 3.13:

h	σ_1	σ_2	...	σ_m

Figure 3.13 - The real-coded genetic individual

Initialization

For a given size of population, say p, we randomly generate p individuals, which are real vectors with the length of m+1, and each entry in the vector is between the upper bound and lower bound of smoothing factors.

Evaluation

Because the target is to minimize the performance index Q defined in equation (10) and the GA always gives the individuals with higher fitness values greater chances to survive to the next generation, 1/(1+Q) is selected as the fitness function to evaluate each individual.

Selection

The proposed algorithm uses a roulette-wheel selection procedure and always keeps the best individual to the next generation. This means that at generation g, for a population of n individuals with fitness values $f_1, f_2, \ldots, f_n$, the individuals' probabilities of being chosen for the next generation are p_1, p_2, ..., p_n, where $p_i = f_i / \sum_{j=1}^{n} f_j$ (i=1, 2, ..., n). We first compute $q_i = \sum_{j=1}^{i} p_j$ (i=1, 2, ..., n), then perform the following step n times: generate a random number x, which is uniformly distributed in the unit interval and if $q_{i-1} < x \le q_i$, (i=1, 2, ..., n; $q_0 = 0$), select the ith individual at generation g into the next generation g+1.

Variation

Two variation operators applied in this algorithm are crossover and mutation.

1. Crossover operator:

This algorithm simply applies the one-point crossover. That is, given two parent individuals, such as $x=x_1x_2\ldots x_kx_{k+1}\ldots x_L$ and $y=y_1y_2\ldots y_ky_{k+1}\ldots y_L$, we generate two offspring $x'=x_1x_2\ldots x_ky_{k+1}\ldots y_L$ and $y'=y_1y_2\ldots y_kx_{k+1}\ldots x_L$; where k is a randomly selected position. Figure 3.14 depicts this operator:

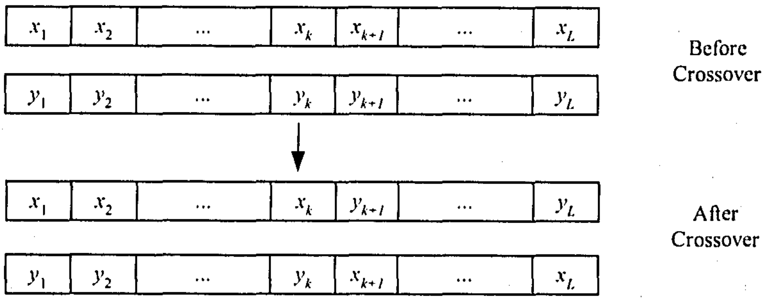


Figure 3.14 - The one-point crossover

2. Mutation operator:

The mutation operator is realized by the Gaussian mutator, which picks a new value based on a Gaussian distribution around the current value. This means that for the entry/gene x^i to be mutated in a individual/chromosome (a vector of real numbers), the mutation gene $x^{i\prime}=x^i+N(0,1)$, where $N(0,1)$ is value generated from a Gausssian distribution with zero mean and standard deviation 1. Of course, if $x^{i\prime}$ is out of the gene's boundary, it will be truncated to interval [lower bound, upper bound].

Stop criteria

The maximum number of generations is set as the stop criteria.

3.2 OR/AND Neuron vs. GRNN on Construction Data

With the introduction of local smoothing factors into GRNN, GRNN has not only achieved improved accuracy but also gained importance indices for each input's impact on output. Then between OR/AND neuron and GRNN, both of which satisfied the requirements mentioned at the beginning of this chapter, which one should be selected as the modeling technique to build a construction performance model? This section compares the performances of the two techniques on real data collected in a construction site. The results demonstrate that the adaptive GRNN achieves greater accuracy and is the best choice as the basic modeling technique.

3.2.1 Description of Data

The data studied in this thesis were collected from a pipe module fabrication yard in an industrial construction project in Edmonton, Alberta, during the period of April 2003 to May 2004 [79]. The "labor productivity in hydro testing (HT productivity)" is taken as the construction performance indicator considered in the construction activity. Table 3.1 tabulates the 7 factors impacting "HT productivity". All the influence factors are determined by a group of experts who manage the construction project [74] and they are assumed to be independent. The short descriptions of all the influence factors are also listed in the same table.

The datasets, including values of both the influence factors and the construction performance indicator, namely "HT productivity", were extracted from the constructor's Information Management System, for a period of 169 working days. Therefore, we get a historical data containing 169 datasets in which each dataset includes 7 inputs (impacting factors) and 1 output (the performance indicator).

Before OR/AND neuron and GRNN are applied on the historical data to approximate the mapping between inputs and output, data preprocessing is required for the following reasons: First, OR/AND neuron is a nonlinear transformation between unit hypercubes, therefore, the inputs and output should be normalized into the unit interval [0, 1]. Secondly, since the GRNN is a typical neural network, it is recommended to first normalize inputs and output before the feeding of the data [73]. Therefore, normalizing functions listed in Table 3.3 are applied to the collected datasets, and the OR/AND and GRNN are then compared by their performance on the normalized data.

Factor/ Variable			Description
1	WKL	Work Load	No. of pipe modules in progress
2	EQA	Equipment availability	No. of cranes available
3	MAV	Manpower availability	No. of pipefitters available
4	TEM	Mean Temperature	The mean temperature of the air in degrees Celsius.
5	PRE	Total precipitation	The sum of the total rainfall and the water equivalent of the total snowfall
6	RWK	Rework	Pipe fabrication rework (work force hours spent on repairs)
7	QAC	Quality Assurance/ Quality Control input	No. of hours spent on QA/QC work.

Table 3.2 - Factors that affect labor productivity in Hydro-testing [74]

Variable	Normalizing function (NF)*	Parameters of NF *
WKL	Sigmoid	0.5
EQA	Sigmoid	-0.5
MAV	Sigmoid	-0.5
TEM	1-Gaussian	(0, 5)
PRE	Sigmoid	0.5
RWK	Sigmoid	0.5
QAC	1-Gaussian	(0, 10)
HT	Sigmoid	-0.5

Table 3.3 - Normalizing functions for the variables

* There are two type of normalizing factions. The Sigmoid function is $y=1/(1+e^{-\sigma x})$ and the only parameter in it is σ. The 1-Gaussian function is $y=1-e^{-(x-c)^2/2\sigma^2}$, where ($c$, σ) are the two parameters.

The performance index considered in this comparison is the mean square error (MSE):

$$Q=\sum_{i=1}^{169}(y_i-\hat{y}_i)^2/169,$$

in which, y_i is the output of the i^{th} datasets and $\hat{y}_i$ is the predicted output of the OR/AND neuron or the GRNN.

It should be noted that the most helpful technique for normalizing the data is most likely to be fuzzy information granulation based on fuzzy sets/ linguistic terms. The application of linguistic terms will not only normalize the datasets, but also enhance the model's transparency. More details will be discussed in Chapter 4.

3.2.2 Comparison Results

At first, for the augmented OR/AND neuron, with sigmoid function attached in the output layer, is tested on the collected 169 datasets. The initial and resulting weights of connections in AND and OR neurons are listed in Table 3.4. The initial connection weights of the OR neuron in the output layer are v=[1 1] and the resulting ones are v=[1 0.51]. The initial and resulting parameter pair in the sigmoid function are [m =0.5 σ =1] and [m =0.51 σ =0.91].

Variable	Initial weights (OR neuron)	Initial weights (AND neuron)	Resulting weights (OR neuron)	Resulting weights (AND neuron)
WKL	0.9381	0.0618	1.0000	1.0000
EQA	0.5384	0.4615	0.9376	1.0000
MAV	0.8363	0.1636	1.0000	0.1760
TEM	0.8377	0.1622	0.0000	0.0000
PRE	0.8286	0.1713	0.5641	0.9872
RWK	0.0000	1.0000	1.0000	0.9739
QAC	0.1838	0.8161	0.0000	1.0000

Table 3.4 - Initial and resulting connection weights of OR/AND neuron

The performance index according to the successive iterations of learning is shown in

Figure 3.15. The final performance index Q=7.86*10^{-3}.

The OR/AND neuron's outputs vs. real targets or outputs in the historical data are plotted in Figure 3.16. It can be figured out that the OR/AND neuron cannot approximate the mapping between the inputs and output very well, despite the sigmoid function being appended in the output layer. Because the OR/AND neuron model cannot approximate the input-output mapping very well, the importance indices derived from this model is no longer reliable any more.

As a comparison, the adaptive GRNN learned through the genetic algorithm is also applied on the 169 datasets. The 169 datasets are divided into two datasets in which the training data has 135 data patterns and the validation data has 34 data patterns. It is imperative to mention that the setting of parameters of a genetic algorithm is not trivial and may take some trial and error in order to achieve acceptable results for a number of problems. However, the experiments prove that this problem is not sensitive to the parameter setting of the genetic algorithm. Therefore, the chosen parameters are these simplest ones in Figure 3.17. After 100 generations of evolution, the resulting local smoothing factors are [26.538 45.6343 18.0074 42.6319 72.439 0.222879 4.06774], and the global smoothing factor is 0.85. Here the global smoothing factor is a small value that means the surface of input-output mapping is steep [66].

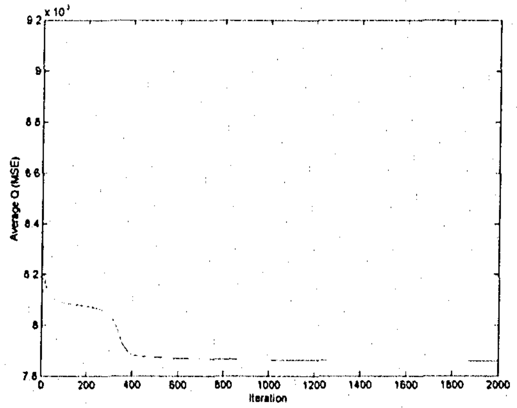


Figure 3.15 - Performance index in successive learning epochs

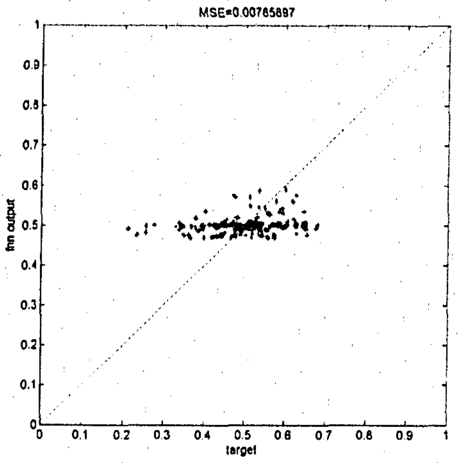


Figure 3.16 - OR/AND neuron's outputs vs. real targets

Probability of Crossover: 0.9
Probability of Mutation: 0.01
Size of Population: 50
Maximum generation: 100

Figure 3.17 – The setting of parameters of GA

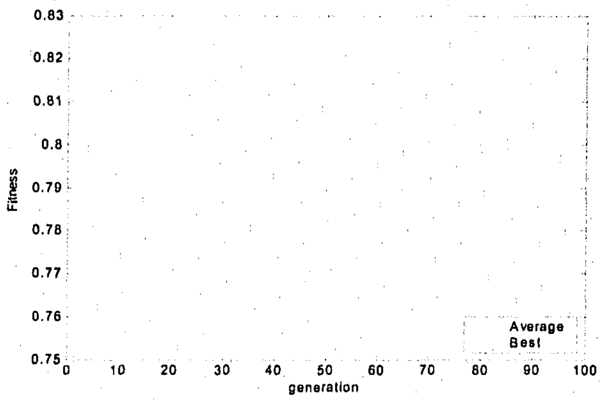


Figure 3.18 - Best and Average fitness of individuals in successive generations

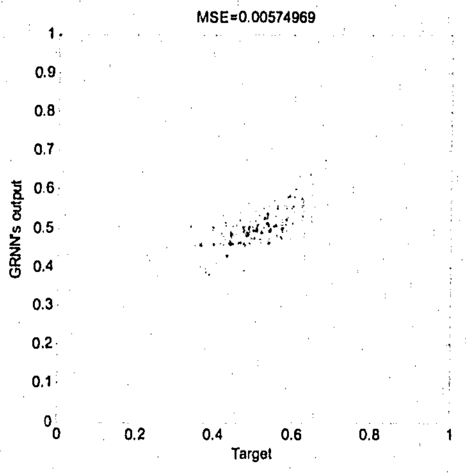


Figure 3.19 - GRNN's output vs. real targets

Figure 3.18 shows the best and average fitness of individuals in successive generations of evolution. The resulting GRNN's outputs, compared with the real output of the system (the targets), are demonstrated in Figure 3.19. The final performance index is Q=5.75*10^{-3}.

Based on local smoothing factors: [26.53 45.63 18.00 42.63 72.43 0.22 4.06], it is easy to conclude that the 5th factor – "Total precipitation" has a critical impact on the construction performance -"HT productivity" because its local smoothing factor holds a distinct higher value than others. On the contrary, the 6th factor – "Rework" has little impact on the construction performance due to the small value of its local smoothing factor. These conclusions are helpful insights for managers who are inspecting the performance problems related to "HT productivity".

After studying the GRNN's performance on the real construction data (Q=5.75*10^{-3}), and the OR/AND neuron's performance on the same data (Q=7.86*10^{-3}), the apparent conclusion is that the GRNN can approximate real construction data more accurately than the OR/AND neuron. Although both OR/AND neuron and GRNN provide the importance indices of inputs on the output, the model demonstrating greater accuracy should be regarded as more reliable. Therefore, the decision is to select the adaptive GRNN learned via the genetic algorithm as the technique to build construction performance model. Chapter 4 will apply the proposed GRNN on the construction data granulated through the membership functions (representing the linguistic terms), which are automatically generated from a strengthened fuzzy c-means algorithm.

3.3 Conclusion

This chapter discussed two Computational Intelligence based models. The OR/AND neuron is a fuzzy neural network model and the adaptive GRNN is a general neural network model. Although both models offer the importance indices of inputs on the output, adaptive GRNN learned through a real-coded genetic algorithm achieves much greater approximation accuracy when applied on real data collected from a construction site. In conclusion, according to the collected construction data, the adaptive GRNN is selected as the basic technique for the task of construction performance modeling.

Chapter 4
Enhancing Transparency of the Model via Information Granulation

In Chapter 3, the GRNN is selected as the basic technique for construction performance modeling. However, the influence factors and resulting construction performance in the proposed model are still described as numeric variables. For example, we still consider the influence factors such as "Mean Temperature" and "Crew Size" and their importance to the resulting construction performance indicator, for example, "Labor Productivity". All these factors and indicators are numeric variables.

Linguistic descriptions are in fact, found almost everywhere in real construction situations. Typical linguistic terms used in construction projects are "High Crew Skill Level", "Poor Tool Condition", and so on. Furthermore, we can transform a numeric input into several memberships of its linguistic terms. For example, if we generate three linguistic terms to portray the variable "Crew Size", i.e., "Small Crew Size", "Normal Crew Size" and "Large Crew Size", the numeric value of "Crew Size" is then transformed into 3 memberships of the linguistic terms of "Small Crew Size", "Normal Crew Size" and "Large Crew Size". This transformation procedure results in more concrete influence factors and construction performance indicators, and greatly enhances the transparency of the construction performance model. For instance, it is more concrete and transparent to consider the influence of "High Mean Temperature" and "Small Crew size" on "Low Productivity" but the influence of "Mean Temperature" and "Crew Size" on "Labor Productivity".

The above transformation is a process of information granulation in which the information granules take the form of semantically meaningful fuzzy sets, namely linguistic terms, distributed fully over each variable's universe of discourse. However, building decent membership functions for the linguistic terms is not an easy task. Although determining membership functions based on expert knowledge is possible, it needs more time and experts' understanding of the concept of membership functions. The more desirable method is generating membership functions automatically from historical data – in particular, building membership functions through clustering of historical data.

This Chapter proposes a membership function elicitation method based on a fuzzy clustering algorithm. This method takes the clustering problem as an encoding problem. Derived from the principle of minimum information loss in decoding, it suggests a performance index known as representation error to constrain the encoding procedure, i.e., the fuzzy clustering algorithm. A detailed description of the proposed method is given in Section 4.1. The new method is tested the real construction data in Section 4.2. The resulting fuzzy sets are compared with expert defined fuzzy sets in the same section. Finally, the generated membership functions are utilized in the information granulation of collected data, and the granulated data is fed to the adaptive GRNN learned through the genetic algorithm. The improved performance and the enhanced transparency of GRNN on the granulated data are demonstrated in Section 4.3.

4.1 FCM based Membership Function Construction

4.1.1 One-Dimensional FCM with Decoding Consideration

Given a number of historical examples of a numeric variable $[x_1, x_2, \ldots, x_N]$ and the anticipated number of clusters – c, the Fuzzy C-means (FCM) [75] algorithm partitions these examples into c clusters by minimizing the objective function Q:

$$Q = \sum_{k=1}^{c} \sum_{j=1}^{N} u_{j,k}^{m} (x_j - o_k)^2 .$$

In this objective function, Q represents the sum of the distance of individual data to the cluster centers $[o_1, o_2, \ldots, o_c]$ (The cluster centers are also called prototypes). Here $u_{j,k}$

represents the membership of data x_j belonging to cluster k. The parameter m stands for the fuzzification factor that is a real number greater than 1.

The FCM iteratively updates the prototypes $[o_1, o_2, \ldots, o_c]$ and memberships $u_{j,k}$ through the equations below [75] :

$$o_k = \frac{\sum_{i=1}^{N} u_{i,k}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{i,k}^{m}},$$

$$u_{i,k} = \frac{1}{\sum_{j=1}^{c} \left(\frac{x_i - c_k}{x_i - c_j} \right)^{\frac{2}{m-1}}}.$$

This iteration will stop when

$$\max_{i,k}\left(\left| u_{i,k}^{ite+1} - u_{i,k}^{ite} \right| \right) < e,$$

where e is a termination criterion, and ite are the iteration steps. This procedure helps to find values of prototypes and memberships that achieve a saddle point or local minimum of objective function Q.

It is worth mentioning that, although the fuzzification factor m has a great impacting on the resulting clusters and memberships [76], its value is usually selected as 2 for simplicity. If the FCM clustering is regarded as an encoding procedure, the corresponding decoding can be introduced. In order to minimize the information loss from encoding to decoding, an applicable fuzzification factor can be selected.

With the FCM algorithm, the original data x_i can be regarded as being encoded into c memberships $[u_{i,1}, u_{i,2}, \ldots, u_{i,c}]$belonging to the c prototypes $[o_1, o_2, \ldots, o_c]$. Therefore, to reconstruct the data x_i from the memberships, the weighted average of prototypes is computed as the decoded value of x.

$$\hat{x}_i = \frac{\sum_{k=1}^{c} u_{i,k}^m \cdot o_k}{\sum_{k=1}^{c} u_{i,k}^m}.$$

In the previous equation, $u_{i,k}^m$ actually represents the belongingness of x_i to clusters k with consideration of the fuzzification factor m.

In order to minimize the information losses from encoding to decoding, the representation error V, or the difference between all the original data and the decoded data, should be minimized [76]

$$V = \sum_{i=1}^{N} (x_i - \hat{x}_i)^2$$

Therefore, the best fuzzification factor m is selected as the one that leads to the smallest representation error.

4.1.2 One-Dimensional FCM based Membership Function Elicitation

For a system with n input variables $[x_1, x_2, \ldots, x_n]$ and one output variable y, there are N historical data $[\mathbf{X}_i, y_i]$ (i=1, 2, ..., N), where $\mathbf{X}_i$ is the n dimensional input vector $[x_{i1}, x_{i2}, \ldots, x_{in}]$ and y_i is the output. The first step to generate membership functions from historical data is to apply FCM to each dimension of the historical datasets. That is, for input variable x_i, apply one-dimensional FCM to the historical data $(x_{i1}, x_{i2}, \ldots, x_{iN})$ to generate c clusters. The resulting prototypes are marked as $[o_{i1}, o_{i2}, \ldots, o_{ic}]$.

For simplicity, having the prototypes $[o_{i1}, o_{i2}, \ldots, o_{ic}]$, the simplest fuzzy partition of input variable x_i's universe of discourse is defining triangular membership functions for fuzzy sets/granules centered at prototypes. In addition, if we use triangular membership functions, the group of fuzzy sets defined over the universe of discourse is a frame of cognition of the input variable x_i because it retains the typical properties of a frame of cognition, such as distinguishability and complementarity [78]. The resulting c membership functions for variable x_i are labeled as $F_{i1}, F_{i2}, \ldots, F_{ic}$, in which, F_{i1} is a function, such as:

$$y = \begin{cases} 1, \ if\ x \le o_{i1}; \\ (o_{i2} - x)/(o_{i2} - o_{i1}), \ if\ o_{i1} < x < o_{i2}; \\ 0, \ if\ x \ge o_{i2}; \end{cases}$$

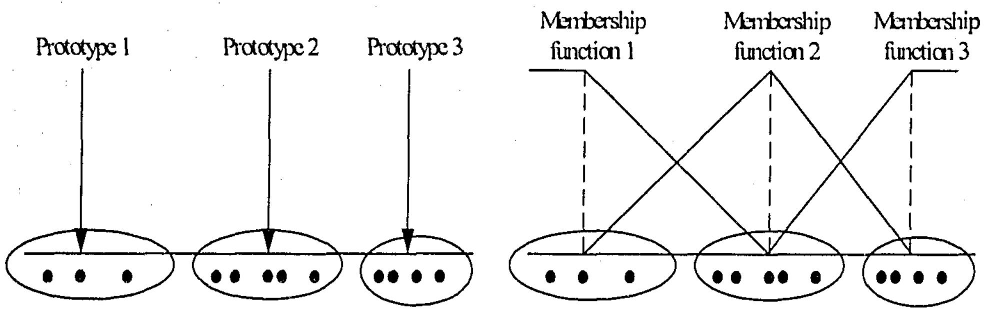
F_{ij} (j=2, 3..., c-1) is a triangle membership function, such as:

$$y = \begin{cases} 0, \ if\ x \le o_{i(j-1)}\ or\ x \ge o_{i(j+1)} \\ (x - o_{i(j-1)})/(o_{ij} - o_{i(j-1)}), \ if\ o_{i(j-1)} < x < o_{ij} \\ (o_{i(j+1)} - x)/(o_{i(j+1)} - o_{ij}), \ if\ o_{ij} < x < o_{i(j+1)} \end{cases}$$

And F_{ic} is a function like:

$$y = \begin{cases} 1, \ if\ x \ge o_{ic} \\ (x - o_{i(c-1)})/(o_{ic} - o_{i(c-1)}), \ if\ o_{i(c-1)} < x < o_{ic}; \\ 0, \ if\ x \le o_{i(c-1)} \end{cases}$$

As an illustration, if the datasets are grouped into 3 clusters, as depicted in Figure 4.1 (a), the generated 3 membership functions are as shown in Figure 4.1 (b).



(a) Data clustered in 3 groups (b) The 3 generated membership functions

Figure 4.1 - An example of clustering based membership function generation

4.2 Membership Functions Generated from Construction Data

This section discusses the application of the proposed membership function elicitation method to real construction data. The collected construction data is the same as that used in last chapter: there are seven input variables or influence factors, and one output variable – construction performance. The number of samples used is 169. More detailed descriptions of each variable can be found in Section 3.3.

To describe Variable 1 – Work Load – the used number of linguistic terms is from 2 to 4. (For example, if 3 linguistic terms is applied, they could be denoted as "Low Work Load", "Average Work Load" and "High Work Load".) Consequently, the number of clusters in FCM is set from 2 to 4.

Computing the representation error *V* and FCM's objective function *Q* based on different values of fuzzification factor *m* gives the results shown in Figure 4.2.

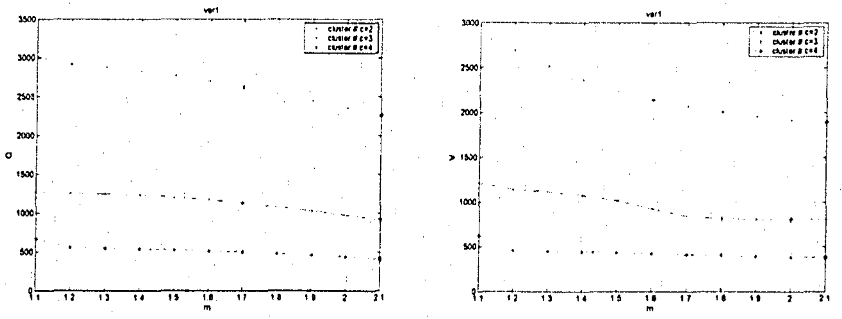


Figure 4.2 - Representation error *V* and FCM's objective function *Q* based on different values of fuzzification factor *m* for Variable 1 - Work Load

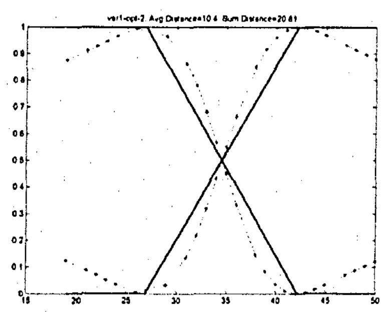
From plots in Figure 4.2, the optimized fuzzification factor *m* is found to be 2.1 in the 2 clusters case; similarly, *m*=2 in the 3 clusters case and *m*=2.1 in the 4 clusters case.

Based on the optimized fuzzification factors, the FCM is performed and the membership functions are plotted in Figure 4.3. The memberships of each dataset belonging to every cluster are plotted in the same figure. Two Hamming distances between the membership values for each dataset in each membership function, and the memberships for each dataset in each cluster, are calculated and denoted at the top of Figure 4.3, in which,

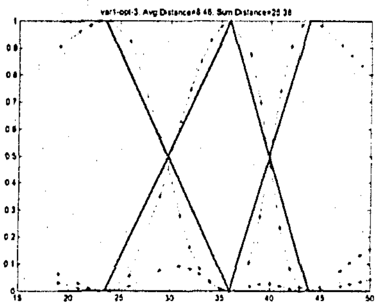
$$\text{Sum Distance}=\sum_{j=1}^{c}\sum_{i=1}^{169}\left|u_{i,j}-F_j(x_i)\right|$$

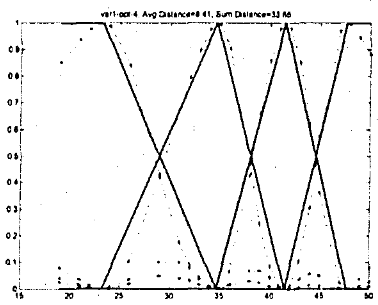
$$\text{Avg Distance}=\frac{1}{c}\sum_{j=1}^{c}\sum_{i=1}^{169}\left|u_{i,j}-F_j(x_i)\right|$$

In the two equations, x_1, x_2, ..., x_{169} are the 169 historical datasets and c is the number of clusters. F_j (j=1, 2, ..., c) are generated membership functions; $u_{i,j}$ is the degree of membership of data x_i in j^{th} cluster. The Avg Sum Distance is regarded as the average approximation error of the defined membership functions to the memberships generated by the FCM algorithm.



(a) Two membership functions case





(b) Three membership functions case (c) Four membership functions case

Figure 4.3 - FCM generated membership functions for Variable 1 - Work Load

For Variable 2 – Equipment Availability – the best fuzzification factors can be deduced from Figure 4.4 (those that achieve the minima of representation error *V*). The final FCM generated membership functions are illustrated in Figure 4.5.

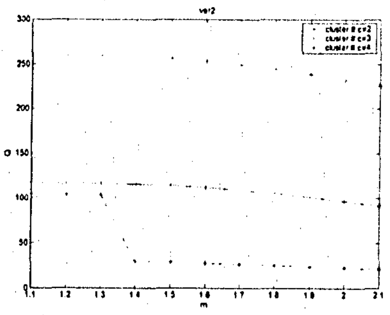
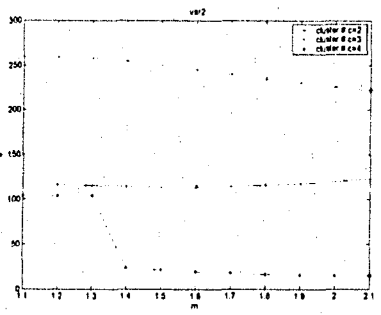
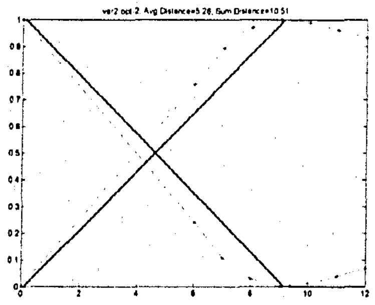
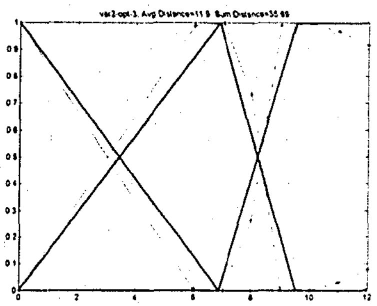
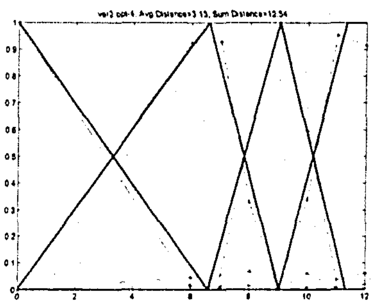



Figure 4.4 - Representation error *V* and FCM's objective function *Q* based on different values of fuzzification factor *m* for Variable 2 - Equipment Availability



(a) Two membership functions case

(b) Three membership functions case (c) Four membership functions case

Figure 4.5 - FCM generated membership functions for Variable 2 - Equipment Availability

For Variable 3 – Manpower Availability – the best fuzzification factors can be deduced from Figure 4.6 (those that achieve the minima of representation error V). The final FCM generated membership functions are illustrated in Figure 4.7.

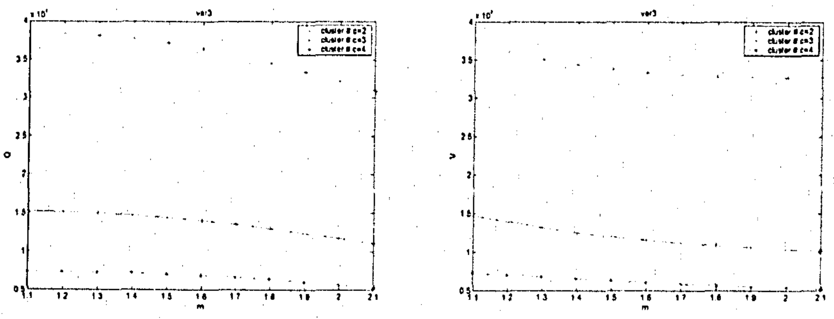
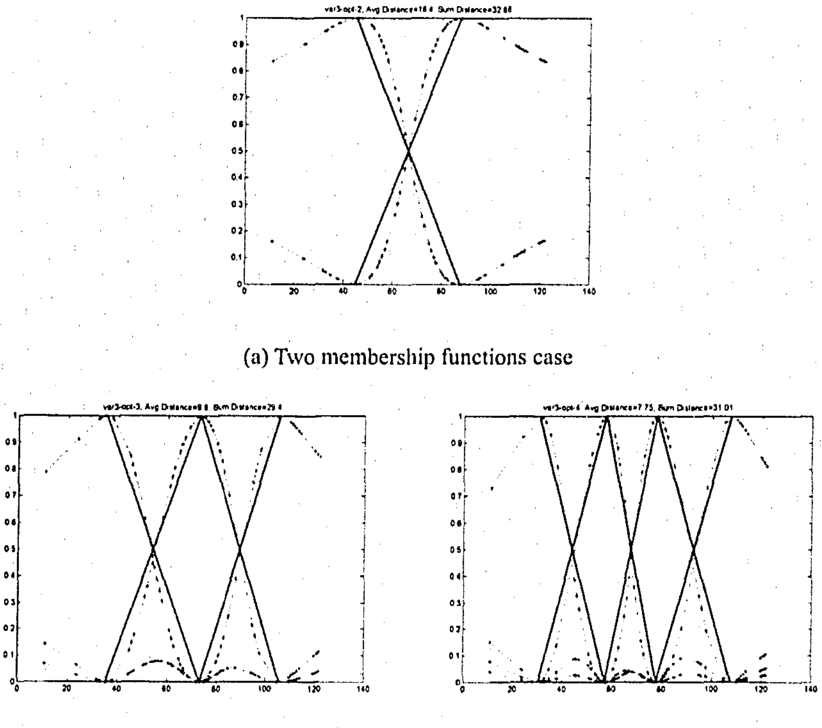


Figure 4.6 - Representation error V and FCM's objective function Q based on different values of fuzzification factor m for Variable 3 - Manpower Availability



(a) Two membership functions case

(b) Three membership functions case (c) Four membership functions case

Figure 4.7 - FCM generated membership functions for Variable 3 - Manpower Availability

For Variable 4 – Mean Temperature – the best fuzzification factors can be deduced from Figure 4.8 (those that achieve the minima of representation error V). The final FCM generated membership functions are illustrated in Figure 4.9.

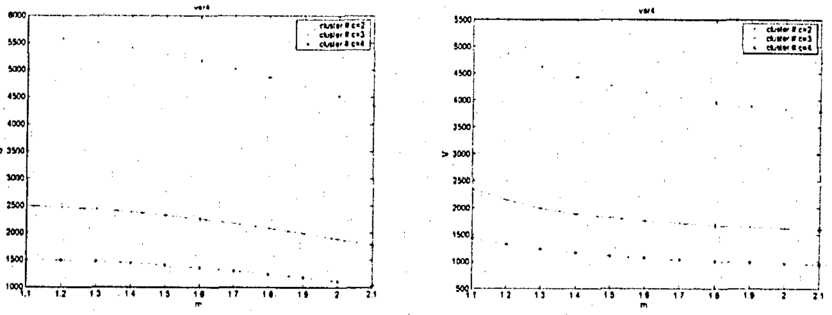
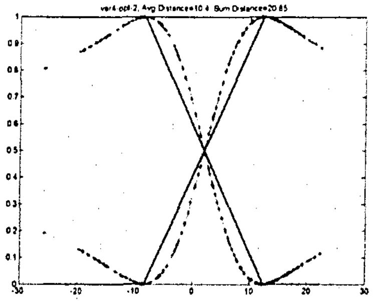
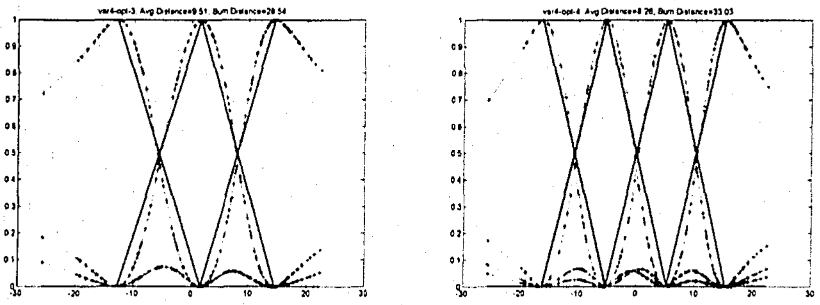


Figure 4.8 - Representation error V and FCM's objective function Q based on different values of fuzzification factor m for Variable 4 - Mean Temperature



(a) Two membership functions case



(b) Three membership functions case (c) Four membership functions case

Figure 4.9 - FCM generated membership functions for Variable 4 - Mean Temperature

For Variable 5 – Total Precipitation – the best fuzzification factors can be seen from Figure 4.10 (those that achieve the minima of representation error *V*). The final FCM generated membership functions are illustrated in Figure 4.11.

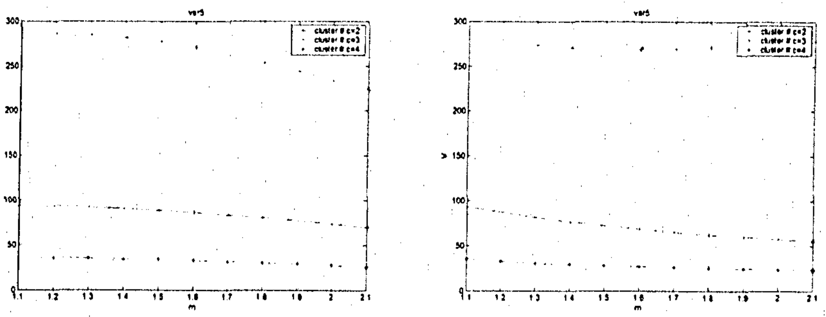
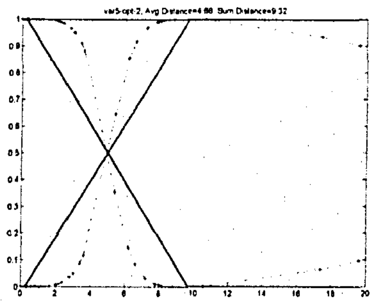
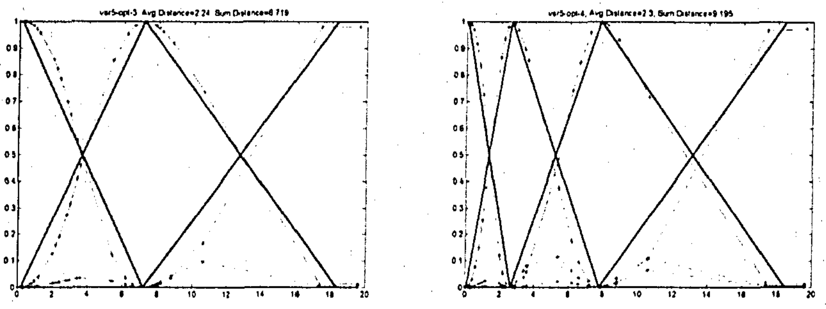


Figure 4.10 - Representation error *V* and FCM's objective function *Q* based on different values of fuzzification factor *m* for Variable 5 - Total Precipitation



(a) Two membership functions case



(b) Three membership functions case (c) Four membership functions case

Figure 4.11 - FCM generated membership functions for Variable 5 - Total Precipitation

For Variable 6 – Rework – the best fuzzification factors can be determined from Figure 4.12 (those that achieve the minima of representation error V). The final FCM generated membership functions are illustrated in Figure 4.13.

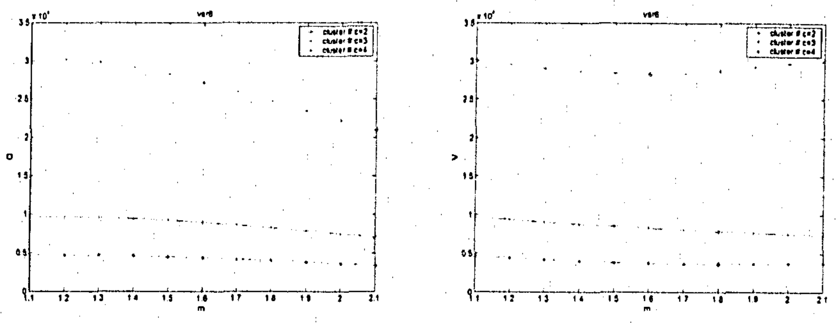
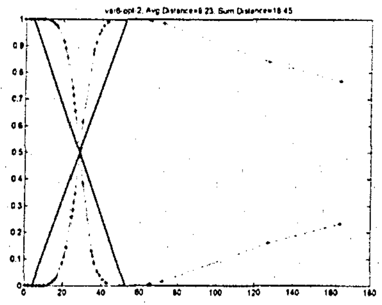
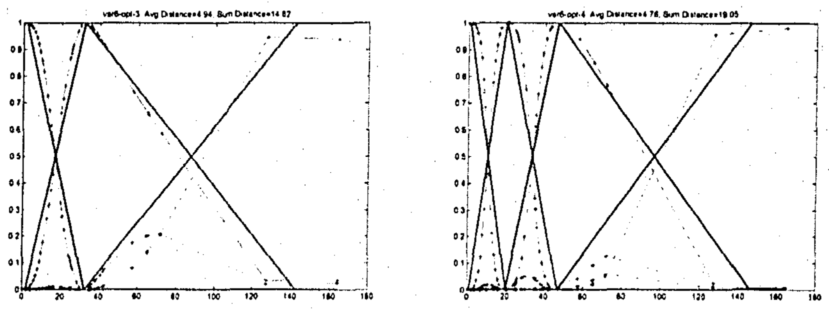


Figure 4.12 - Representation error V and FCM's objective function Q based on different values of fuzzification factor m for Variable 6 - Rework



(a) Two membership functions case



(b) Three membership functions case (c) Four membership functions case

Figure 4.13 - FCM generated membership functions for Variable 6 - Rework

For variable 7 – Quality Control Input – the best fuzzification factors can be deduced from Figure 4.14 (those that achieve the minima of representation error V). The final FCM generated membership functions are illustrated in Figure 4.15.

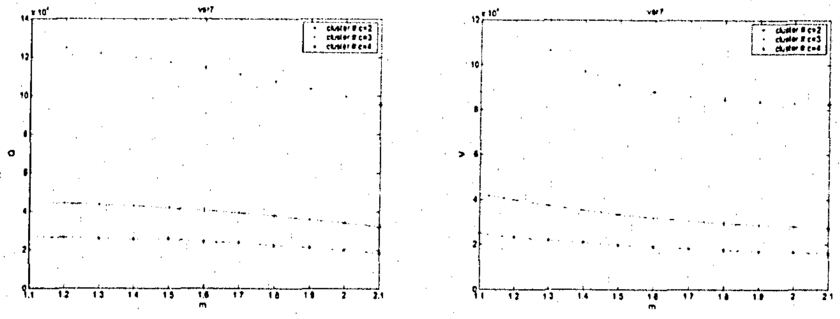
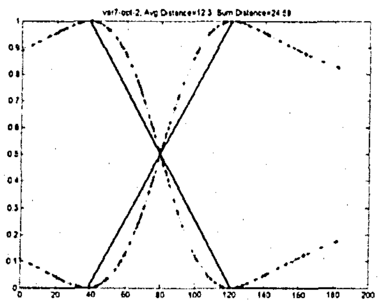
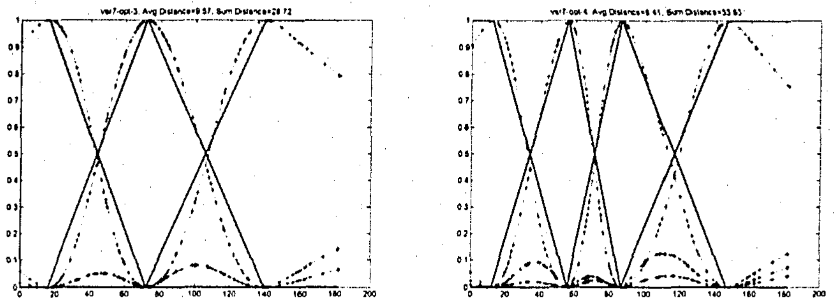


Figure 4.14 - Representation error V and FCM's objective function Q based on different values of fuzzification factor m for Variable 7 - Quality Control Input



(a) Two membership functions case



(b) Three membership functions case (c) Four membership functions case

Figure 4.15 - Membership functions generated based on FCM for Variable 7 - Quality Control Input

Finally for Variable 8, or the output variable – HT Labor productivity – the best fuzzification factors can be determined from Figure 4.16 (those that achieve the minima of representation error *V*). The final FCM generated membership functions are illustrated in Figure 4.17.

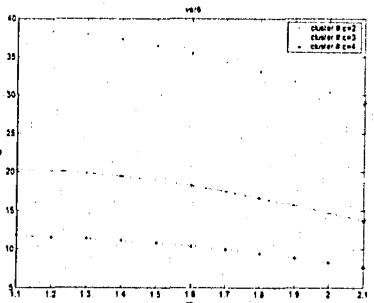
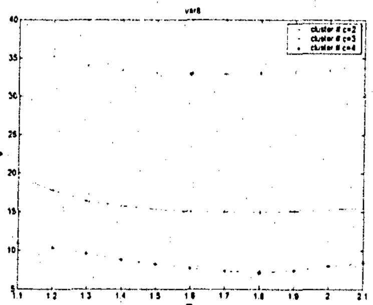
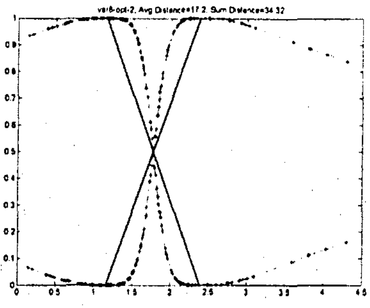
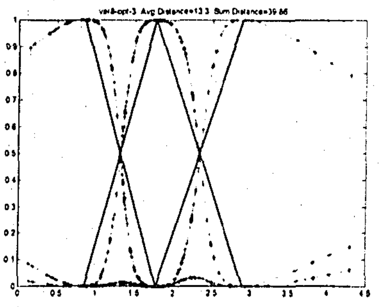
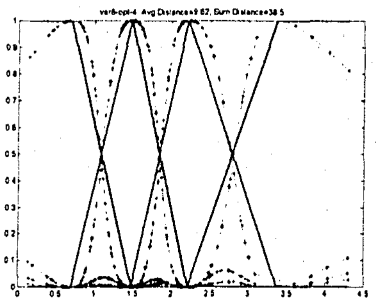



Figure 4.16 - Representation error *V* and FCM's objective function *Q* based on different values of fuzzification factor *m* for output variable - HT Labor productivity



(a) Two membership functions case

(b) Three membership functions case (c) Four membership functions case

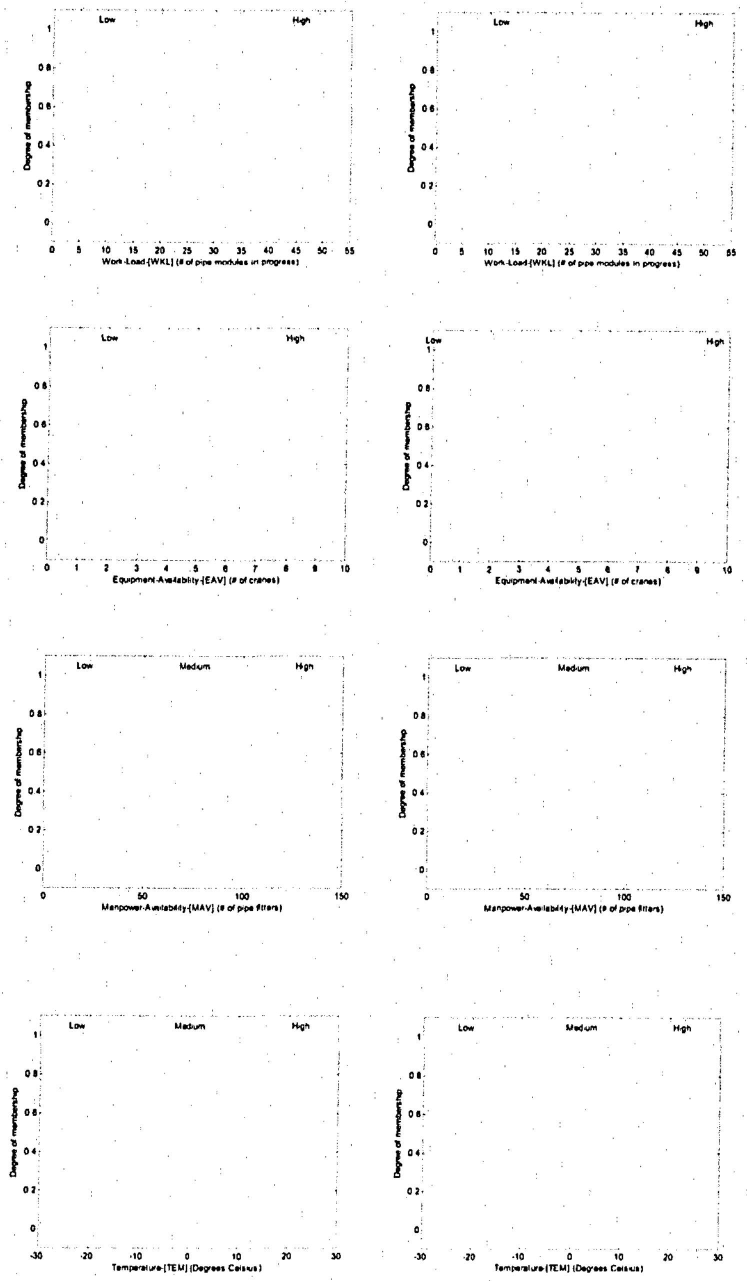
Figure 4.17 - FCM generated membership functions for output variable - HT Labor productivity

Based on the average approximation errors (the Avg Distance) noted on the figures about generated membership functions, the general conclusion is that the average approximation error will decrease with the increase of the number of membership functions. Or in other words, the defined triangular membership functions will approximate the memberships generated by FCM algorithm better if the number of membership functions or clusters is larger. But the larger number of membership functions means larger number of linguistic terms for each variable, or the larger number of granulated influence factors, to build a parsimony model with decent number of influence factors, this research will ask the experts to define the number linguistic terms or membership functions used to describe the input and output variables. Next section will discuss the FCM generated membership functions' capability to capture the expert knowledge about the linguistic terms

4.3 FCM Generated Membership Functions vs. Expert Defined Membership Functions

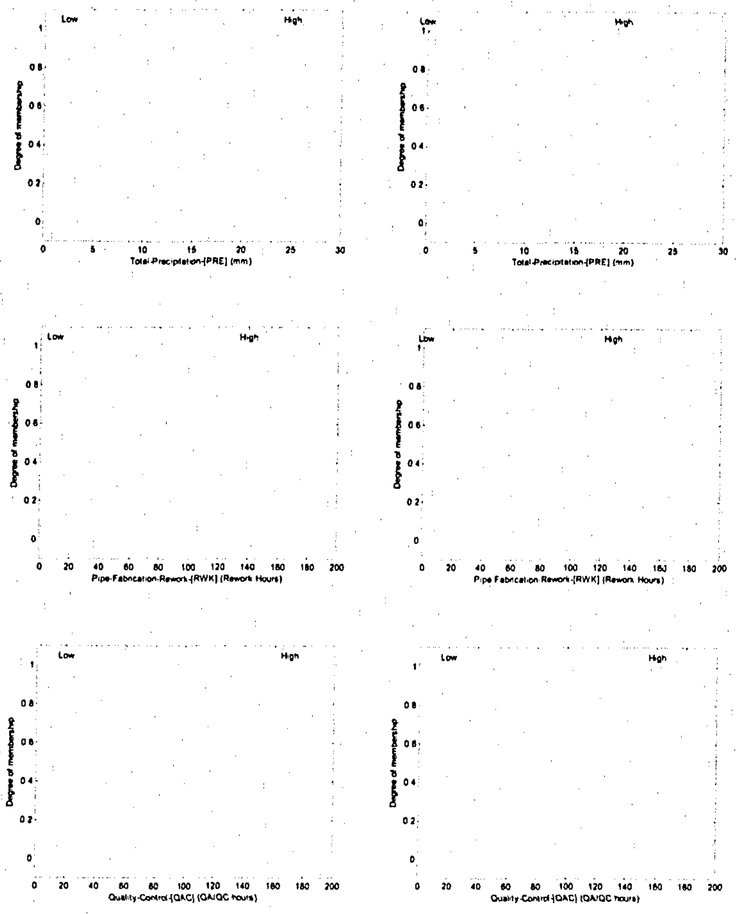
A group of experts has been interviewed to identify linguistic measures of each influence factor considered in the historical data, and the membership functions for each influence factor are directly developed based on their professional experience. That means there is no experimental method applied in the developing procedure. The experts just defined the shapes and the setting of all the membership functions directly from their experience. The resulting membership functions are listed in Figure 4.18 (a).

As a comparison, Figure 4.18 (b) exhibits the membership functions generated through FCM algorithm with decoding consideration. It can be seen that the FCM generated membership functions are very similar to the ones developed by the experts. The comparison suggests that the proposed method is an effective approach for automatically generating membership functions from historical datasets.



(a) Expert defined membership functions (b) FCM generated membership functions

Figure 4.18 - FCM generated MFs vs Expert defined MFs



(a) Expert defined membership functions (b) FCM generated membership functions

Figure 4.18 - FCM generated MFs vs Expert defined MFs (continued)

It is noticeable that, compared with expert defined membership functions, the most visible different membership functions are the ones for Variable 2 - Equipment Availability. After studying Figure 4.19, which plots all 169 data examples of Variable 2, it can be seen that many examples take the same value. This means that only limited values are useful to the clustering algorithm, or in other words, the historical datasets may not contain enough information about the variable. Therefore, the difference between the experts' results and the FCM's results is reasonable. Although different from the experts' results, the FCM's results still tally with human being's intuition on the available data plotted in Figure 4.19: there are two visible clusters: the data taking the value of 0 and the data taking other values.

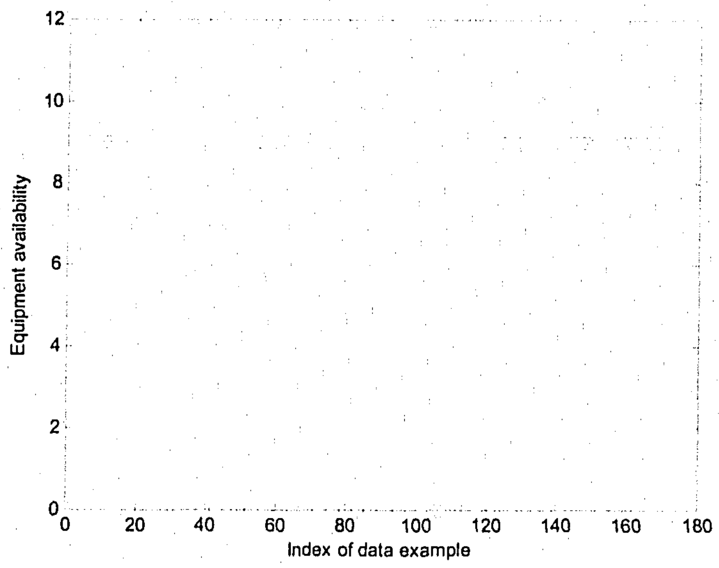


Figure 4.19 - Plot of all 169 data examples of Variable 2 – Equipment Availability

4.4 A More Transparent GRNN Model Based on Granulated Data

Working on the same 169 collected datasets from the real construction site, this section uses the membership functions generated by FCM to define the linguistic terms applied in information granulation. We test the adaptive GRNN on the granulated data.

As shown in Table 4.1, the 7 influence factors are granulated into several linguistic terms. Experts define the number of the linguistic terms based on their experience. The new system will hold 16 inputs instead of 7. For example, the "Work Load" factor is granulated into 2 factors, "Low Work Load" and "High Work Load".

Impacting factors	Linguistic terms
Work Load	Low, High
Equipment Availability	Low, High
Manpower Availability	Low, Medium, High
Mean Temperature	Low, Medium, High
Total Precipitation	Low, High
Rework	Low, High
Quality Assurance/ Quality Control Input	Low, High

Table 4.1 - Linguistic terms of influence factors

To justify the information granulation's capability to improve the performance of the GRNN model, the output or the performance indicator considered in the system is still "labor productivity in hydro testing (HT productivity)". The output or the construction performance indicator is normalized through the Sigmoid function $y=1/(1+e^{-\alpha x})$ as was done in Chapter 3.

The parameters of the genetic algorithm applied to learn the GRNN are set to be the same as the ones used in Chapter 3. The genetic learning of GRNN achieves the final performance index or the mean square error (MSE) $Q= 4.60*10^{-3}$. Compared with the performance achieved in Chapter 3 – $Q=5.75*10^{-3}$, implementing information granulation to influence factors greatly improves the performance of the GRNN model. Figure 4.20 shows the best and average fitness of individuals in successive generations of evolution. The resulting GRNN's outputs, compared with the real outputs of the system (the targets), are demonstrated in Figure 4.21.

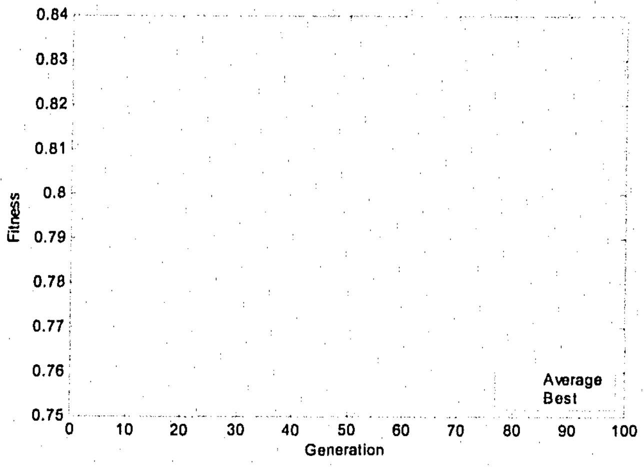


Figure 4.20 - Best and Average fitness of individuals in successive generations

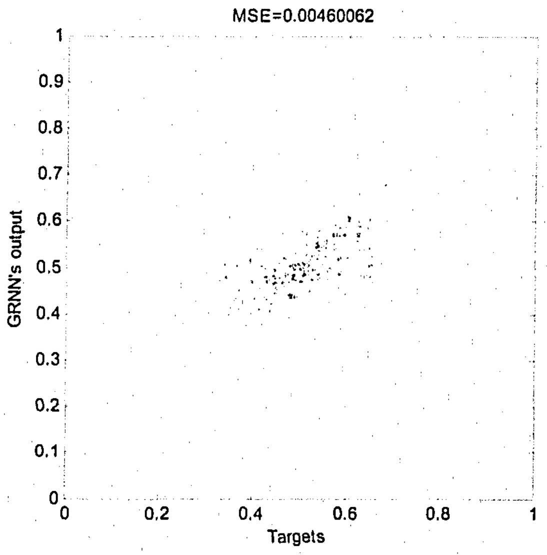


Figure 4.21 - GRNN's output vs. real targets

To obtain more useful insights from the historical construction data, we also granulate the output – "labor productivity in hydro testing (HT)" into three linguistic terms, namely "Low labor productivity", "Medium labor productivity", and "High labor productivity". Then the adaptive GRNN is applied to capture the relationships between the 16 linguistic influence factors and the more concrete linguistic construction performance indicator – "Low labor productivity".

Taking the same setting of the genetic algorithm as the last experiment, the resulting local smoothing factors of GRNN for the 16 linguistic influence factors are [10.89 9.67 23.50 59.44 92.39 22.62 85.74 28.12 0.83 99.64 50.18 19.08 4.05 0.71 65.79 97.44]. The final global smoothing factor is 0.99. It is a small value means the kernel function will be a steep one. Therefore the surface of input-output mapping, which is generated by the combination of these kernel functions, is an elastic surface. The best and average fitness of individuals in successive generations of evolution are shown in Figure 4.22. The outputs of the resulting GRNN, compared with the real outputs in historical data (the targets), are demonstrated in Figure 4.23. The GRNN finally achieves a mean square error $Q = 3.35*10^{-3}$.

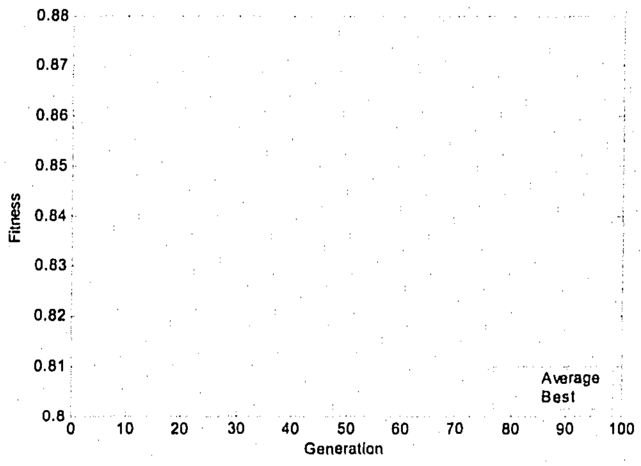


Figure 4.22 - Best and Average fitness of individuals in successive generations (with granulated output)

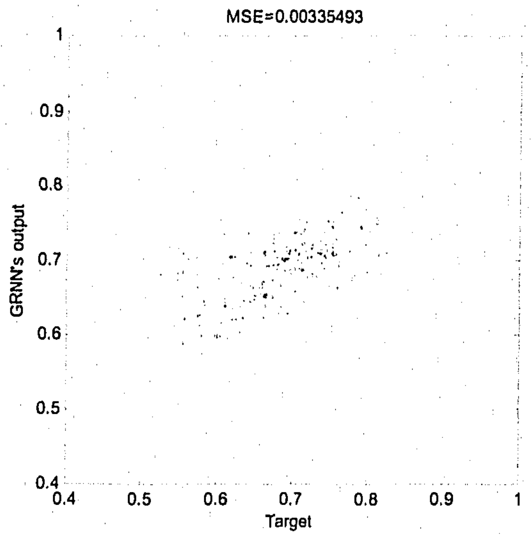


Figure 4.23 - GRNN's output vs. real targets (with granulated output)

Impacting factors		Importance index (Smoothing factor)	Impacting factors		Importance index (Smoothing factor)
Work Load	Low	10.89	Mean temperature	Low	28.12
	High	9.67		Medium	*0.83*
Equipment availability	Low	23.50		High	**99.64**
	High	59.44	Total precipitation	Low	50.18
Manpower availability	Low	**92.39**		High	19.08
	Medium	22.62	Rework	Low	4.05
	High	85.74		High	*0.71*
			Quality control input	Low	65.79
				High	**97.44**

Table 4.2 - Linguistic influence factors and their importance indices

Table 4.2 lists all the local smoothing factors according to separate linguistic influence factors. Because the local smoothing factors or the importance indices for "High Rework" and "Medium mean temperature" are close to 0, these two factors have almost no influence on the output –"Low labor productivity". On the contrary, the importance indices for "High mean temperature", "High quality control input" and "Low manpower availability" are much larger than the ones for other influence factors. Therefore, we can conclude that when the problem of "Low labor productivity" occurs in the studied construction site, factors such as "High mean temperature", "High quality control input", and "Low manpower availability" should be looked at more closely. These transparent insights on construction performance obviously have value to managers who have observed problems in construction performance, and wish to identify and quantify the possible reasons. It is worth mentioning that those insights on construction performance are just for the specific construction site, and they are the results inferred from collected data. Those insights just give project managers some suggestions instead of final decisions.

To test the GRNN model's performance especially the generalization capability on the granulated data, the 169 collected data are randomly split into 120 training data and 49 testing data. Taking the same setting of genetic algorithm as last experiment, the achieved performance index (MSE) for the training data is Q=4.87*10^{-3}. The performance index for testing data is Q=7.12*10^{-3}. The outputs of the resulting GRNN, compared with the real outputs in data, are demonstrated in Figure 4.24. The interesting finding is that the GRNN model's approximation rate or performance index on training data is even worse than its approximation rate on the whole 169 datasets. The reason is that the GRNN learned by the genetic algorithm has already considered the capability of generalization though the hold-out validation. The small number of training data may do not hold enough information about the mapping between the inputs and the output, and to get decent generalization capability, the GA learned GRNN may choose the lower approximation rate.

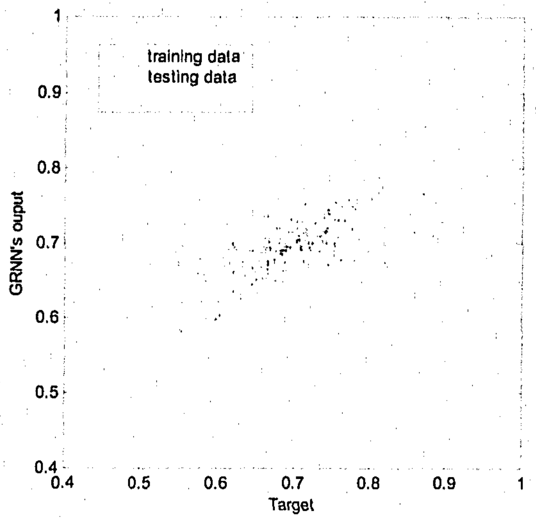


Figure 4.24 - GRNN's output vs. real targets (with training and testing data)

4.5 Conclusion

An information granulation procedure is introduced to transform the numeric value of an influence factor to the memberships of its linguistic terms. A strengthened fuzzy c-means (FCM) algorithm is proposed to determine the membership functions of linguistic terms. Comparison between FCM generated membership functions and the expert defined membership functions shows the effectiveness of the proposed method. Finally, the adaptive GRNN is applied on the data granulated by the membership functions defined by FCM. The resulting model demonstrates both better accuracy and better transparency. It proves that the implanting of information granulation in the construction performance model enhances its transparency.

Chapter 5
Conclusion and Future Work

In construction, when the actual performance diverges greatly from the anticipated performance, managers need to carefully study the possible causes of the discrepancy, in order to prevent it from occurring in the future. Computer tools, which automatically identifies the possible the causes of performance problem, and quantifies the impacts of different causes, would be of considerable help to managers. In the framework of such a diagnostic tool, a transparent construction performance model is required. This model should not only offer important indices of inputs of the model, but should also embed the information granulation process. The objective of this thesis, therefore, is to build a transparent construction performance model that useful in the construction performance diagnosis framework, utilizing the techniques of computational intelligence – including neural networks, fuzzy sets and systems, and genetic algorithms and their hybrids.

Three key contributions are made by this thesis:

1. The adaptive general regression neural network (GRNN), which not only possesses a strong approximation capability but also provides an importance index for each input variable, is proposed as the basic modeling technique. A new genetic algorithm based learning algorithm for the adaptive GRNN is also presented

2. A fuzzy c-means (FCM) clustering based membership function generation method, which applies representation error for its fuzzification factor selection, is proposed as the data information granulation technique to enhance the transparency of the construction performance model. The adaptive GRNN is applied on the granulated data or the linguistic data, and shows improved approximation accuracy. The resulting importance indices offered by GRNN are also more understandable since the importance indices are related to the importance of linguistic terms, such as "Low Crew

Size", instead of to the variables, such as of simple numeric variable such as "Crew Size".

3. The proposed modeling techniques are assessed on real data collected from the construction field, and the results demonstrate the feasibility and superiority of proposed techniques.

With respect to future work, two directions are indicated.

First, because construction problems are highly non-linear and dynamic, one possible direction is to collect more real construction data and to build more accurate construction performance models – for example, we can build different models for different construction seasons.

A second possible future work is to apply the proposed modeling techniques, including the genetic learned adaptive GRNN and the FCM based information granulation, to other real problems that require not only an accurate approximation model, but also would benefit from the importance indices marking the quantitative impacts of inputs on the output.

Bibliography

[1]. H. A. Bassioni, A. D. F. Price, and T. M. Hassan, Performance Measurement in Construction, *Journal of Management in Engineering*, 20(2): 42-50.

[2]. D. Dumitrescu, B. Lazzerini, and L. Jain, *Evolutionary Computation*, Boca Raton, FL, CRC Press, 2000.

[3]. E. Fiesler and R. Beale (eds.), *Handbook of Neural Computation*, New York, NY, Institute of Physics and Oxford University Press, 1996.

[4]. E. Ruspini, P. Bonissone, and W. Pedrycz (eds.), *Handbook of Fuzzy Computation*, New York, NY, Institute of Physics Publishing, 1998.

[5]. W. Pedrycz, *Computational Intelligence: An Introduction*, Boca Raton, FL, CRC Press, 1997.

[6]. IEEE Society of Computational Intelligence, http://ieee-cis.org/scope/

[7]. L. Fausett, *Fundamentals of Neural Networks*, Englewood Cliffs, NJ, Prentice Hall, 1994.

[8]. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Englewood Cliffs, NJ, Prentice Hall, 1999.

[9]. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning Internal Representations by Error Propagation, in D. E. Rumelhart & J. L. McClelland (eds), *Parallel Distributed Processing: Explorations in the microstructure of cognition*, MIT Press, 318-361, 1986.

[10]. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* New York, NY, Addison Wesley, 1989.

[11]. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* New York, NY, Springer-Verlag, 1997.

[12]. M. Mitchell, *An Introduction to Genetic Algorithms*, Cambridge, MA, MIT Press., 1998

[13]. L. Fogel, A. Owens and M. Walsh, *Artificial Intelligence Through Simulated Evolution*, New York, NY, John Wiley, 1966.

[14]. X.Yao, Y. Liu and G. Lin, Evolutionary Programming Made Faster. *IEEE Trans.*

Evolutionary Computation 3(2): 82-102, 1999.

[15]. I. Rechenberg, ,*Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[16]. H.-P. Schwefel, *Numerical optimization of computer models* , Wiley, Chichester, 1981

[17]. J.R. Koza, Genetic *Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, MIT Press, 1992.

[18]. J.R. Koza, *Genetic programming II: Automatic Discovery of Reusable Programs*, Cambridge, MA, MIT Press, 1994.

[19]. J.H. Holland and J.H. Reitman, Cognitive Systems Based in Adaptive Algorithms. In D. Waterman and F. Hayes-Roth (eds.) *Pattern-directed Inference Systems*. New York, NY, Academic Press, 1978.

[20]. P.-L. Lanzi, W. Stolzmann, and S.W. Wilson, *Learning Classifier Systems: From Foundations to Applications*, New York, NY, Springer, 2000.

[21]. Z. Michalewicz and D.B. Fogel, *How to Solve it: Modern Heuristics*, New York, NY, Springer-Verlag 2002.

[22]. F. Herrera, M. Lonzano, and J.L. Verdegay, Tackling Real-coded Genetic Algorithms: Operators and Tools for Behavioral Analysis. *Artificial Intelligence Review*, 12(4), 1998.

[23]. H. Wright, Genetic algorithms for Real Parameter Optimization. In G. J. Rawlins (eds), *Foundations of Genetic Algorithms*, 205-218, San Mateo, CA, Morgan Kaufmann, 1991.

[24]. L.A. Zadeh: Fuzzy sets. *Information and Control* 8: 338-353, 1965.

[25]. W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge, MA, MIT Press, 1998.

[26]. W. Pedrycz, *Fuzzy Sets Engineering*, Boca Raton, FL, CRC Press, 1995.

[27]. D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. New York, NY, Academic Press, 1980.

[28]. B. Kosko, *Neural Networks and Fuzzy Systems*, Singapore, Prentice Hall International Inc., 1992.

[29]. R. Belew, J. McInerney and N. Schraudolph, Evolving Networks: Using the Genetic

Algorithm with Connectionism Learning, in *Proceeding Second Artificial Life Conference*, 511-547, New York, NY, Addison-Wesley, 1991.

[30]. A. J. Jones, Genetic Algorithms and Their Applications to the Design of Neural Networks, *Neural Computing & Applications*, 1: 32-45, 1993.

[31]. X. Yao, Evolving Artificial Neural Networks. *Proc. of the IEEE*, 87(9): 1423-1447, 1999.

[32]. J.S.R. Jang and C.T. Sun, Neuro-Fuzzy Modeling and Control, *Proceedings of the IEEE*, 83(3): 378-406, 1995.

[33]. D. Nauck, and R. Kruse, Neuro-Fuzzy Systems for Function Approximation, *Fuzzy Sets and Systems*, 101: 261-271, 1999.

[34]. T. Feuring, Learning in Fuzzy Neural Networks, in *Proc. IEEE Int. Conf. Neural Networks*, Washington, DC, 1061-1066, 1996.

[35]. J.J. Buckley and Y. Hayashi, Fuzzy Neural Networks: A survey, *Fuzzy Sets and Systems* 66: 1-13, 1994.

[36]. K. McGarry, S. Wermter and J. MacIntyre, Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks, *Neural Computing Surveys* 2: 62-93, 1999.

[37]. F. Hoffmann, G. Pfister, Evolutionary Learning of Fuzzy Control Rule Base for an Autonomous Vehicle, *Proc. Sixth Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'96"*, Granada, Spain, 1235-1240, 1996.

[38]. A. G. Tettamanzi, Evolutionary Algorithms and Fuzzy Logic: A Two-way Integration, *Proc. 2nd Joint Conf. on Inf. Sciences*, 464-467, 1995.

[39]. T. L. Seng, M. Bin Khalid, and R. Yusof, Tuning of a Neuro-fuzzy Controller by Genetic Algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 29(2): 226-236, 1999.

[40]. E.E. Gustafson and W.C. Kessel, Fuzzy Clustering with a Fuzzy Covariance Matrix, In: *IEEE CDC*, 761-766, San Diego, California, 1979.

[41]. W. Pedrycz, Conditional Fuzzy Clustering in the Design of Radial Basis Function Neural Networks, *IEEE Trans. on Neural Networks*, 9: 601- 612, 1998.

[42]. A. Sawhney and A. Mund, Adaptive Probabilistic Neural Network-Based Crane Type Selection System, *Journal of Construction Engineering and Management*, American Society of Civil Engineers, 128(3): 265-273, 2002.

[43]. M.W. Emsley, D.J. Lowe, A.R. Duff, A. Harding, and A. Hickson, Development of Neural Networks to Predict Total Construction Costs. *Construction Management and Economics*, 20(6): 465-472, 2002.

[44]. T. Hegazy and A. Ayed, A Neural Network Model for Parametric Cost Estimation of Highway Projects, *Journal of Construction Engineering and Management*, ASCE, 24(3): 210-218, 1998.

[45]. H. Adeli and A. Karim, Scheduling/Cost Optimization and Neural Dynamics Model for Construction, *Journal of Construction Engineering and Management*, ASCE, 123(4): 450-458, 1997.

[46]. M.-Y. Cheng and C.-H. Ko, Object-Oriented Evolutionary Fuzzy Neural Inference System for Construction Management, *Journal of Construction Engineering and Management*, 129(4) 461-469, 2003.

[47]. R. Sonmez and J. E. Rowings, Construction Labor Productivity Modeling with Neural Networks, *Journal of Construction Engineering and Management*, 124(6): 498-504, 1998.

[48]. A. H. Boussabaine, The Use of Artificial Neural Networks in Construction Management: a Review, *Construction Management and Economics Journal*, 14: 427-436, 1996.

[49]. B. M. Ayyub, Systems Framework for Fuzzy Sets in Civil Engineering, *International Journal of Fuzzy Sets and Systems*, 40(3): 491-508, 1991.

[50]. J. H. M. Tah, and V. Carr, A Proposal for Construction Project Risk Assessment Using Fuzzy Logic. *Construction Management and Economics*, 18: 491-500, 2000.

[51]. K. C. Lam, A.T.P. So, T. S. Hu, T. Ng, R.K.K. Yuen, S.M. Lo, S.O. Cheung, and H. Yang, An Integration of Fuzzy Reasoning Technique and Fuzzy Optimization Method in Construction Project Management Decision-making, *Journal of Construction Management and Economics*, 19(1): 63-76, 2001.

[52]. L. Chao and M. J. Skibniewski, Fuzzy Logic for Evaluating Alternative Construction

Technology, *Journal of Construction Engineering and Management*, ASCE 124(4): 297-304, 1998.

[53]. F. A. Robinson and S. Zhuo, A Fuzzy Expert System for Design Performance Prediction and Evaluation. *Canadian Journal of Civil Engineering*, CSCE, 28(1): 1-25, 2001.

[54]. H. Jang and S. Kim, and Jeffrey S. Russell, Manage Space for Construction Facilities on High-rise Buildings, Construction Research Congress, Winds of Change: Integration and Innovation in Construction, *Proceedings of the Congress*, 925-932, 2003.

[55]. S.-S. Leu, A.-T. Chen and C.-H. Yang, A Fuzzy Optimal Model for Construction Resource Leveling Scheduling, *Canadian Journal of Civil Engineering*, 26: 673-684, 1999.

[56]. Y. Natsuaki, H. Furuta, S. Mukandai, and K. Yasuda, Application of Genetic Algorithm to Bridge Construction Management, *Proceedings of ISUMA - NAFIPS '95 The Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society*, 105-108, 1995

[57]. D. X. M. Zheng and M. M. Kumaraswamy, Applying a Genetic Algorithm-Based Multiobjective Approach for Time-Cost Optimization, *Journal of Construction Engineering and Management*, 130(2): 168-176, 2004

[58]. C. M. Tam, T. K. L. Tong, and W. K. W. Chan, Genetic Algorithm for Optimizing Supply Locations around Tower Crane, *Journal of Construction Engineering and Management*, 127(4), 315-321, 2001

[59]. R. Navon and A. M. McCrea, Selection of Optimal Construction Robot Using Genetic Algorithm, *Journal of Computing in Civil Engineering*, 11(3): 175-183, 1997

[60]. Y. C. Toklu, Application of Genetic Algorithms to Construction Scheduling With or Without Resource Constraints, *Canadian Journal of Civil Engineering* 29(3): 421-429, 2002

[61]. H.R. Thomas, and A.S Sakarcan, Forecasting Labor Productivity Using the Factor Model. *Journal of Construction Engineering and Management*, ASCE, 120(1): 228-239, 1994

[62]. S. R. Sanders and H. R. Thomas, Masonry Productivity Forecasting Model, *Journal of*

Construction Engineering and Management, 119(1): 163-179, 1993

[63]. M. Dissanayake, Automating construction performance diagnosis, *Canadian Civil Engineer*, In press, 2005

[64]. K. Hirota, and W. Pedrycz, OR/AND Neuron in Modeling Fuzzy Connectives, *IEEE Transactions on Fuzzy Systems*, 2: 151-161, 1994

[65]. W. H. Press, S. A. Teukolsky, W. T. VEtterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Second edition, Cambridge, Cambridge University Press, 1992.

[66]. D.F. Specht, A Generalized Regression Neural Network, *IEEE Transactions on Neural Networks*, 2: 568-576, 1991

[67]. E. Parzen, On Estimation of a Probability Density Function and Mode, *Ann Math Statist.*, 33: 1065-1076, 1962

[68]. T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, New York, NY, Springer, 2001.

[69]. FAQ of Neural networks, http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-21.html

[70]. D. F. Specht and H. Romsdahl, Experience with Adaptive Probabilistic Neural Networks and Adaptive General Regression Neural Networks. In *Proceedings of IEEE International Conference on Neural Networks*, Orlando, FL, 1994.

[71]. V. N. Vapnik, *Statistical Learning Theory*, New York, NY, Wiley, 1998.

[72]. P.V. Yee, *Regularized Radial Basis Function Networks: Theory and Applications to Probability Estimation, Classification, and Time Series Prediction*, PHD Thesis, McMaster University, Hamilton, Ontario, 1998

[73]. T. Masters, *Advanced Algorithms for Neural Networks: A C++ Sourcebook*, New York, NY, John Wiley and Sons, 1995

[74]. G.M. Dissanayake, L. Chen, W. Pedrycz, A.R. Fayek, and A.D. Russell, Fuzzy logic Modeling of Causal Relationships-Case Study: Reasoning about Construction Performance, Fuzzy Information, 2004. *Processing NAFIPS '04. IEEE Annual Meeting of the*, 2: 605-610, 2004

[75]. F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy Cluster Analysis*. Chichester,

Wiley, 1999

[76]. W. Pedrycz and J. Valente de Oliveira, An R^n Reconstruction Problem in Linguistic Interfaces, *Proc. of IFSA 2005 - The International Fuzzy Systems Association World Congress*, Beijing, China, July 28-31,2005.

[77]. W. Pedrycz (eds.), *Granular Computing: An Emerging Paradigm*, Heidelberg, Physica-Verlag, 2001

[78]. C. Mencar, *Theory of Information Granulation: Contributions to Interpretability Issues*, PhD Thesis, University of Bari, Bari, Italy, 2004

[79]. G.M. Dissanayake, Private Correspondence, 2004