POWER SYSTEM ANALYSIS AND LOAD MANAGEMENT USING SOFT COMPUTING SCHEMES

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

by

NNAMDI IKECHI NWULU

In Partial Fulfillment of the Requirements for the Degree of Master of Science

in

Electrical and Electronics Engineering

NICOSIA 2011

Nnamdi Ikechi Nwulu: POWER SYSTEM ANALYSIS AND LOAD MANAGEMENT USING SOFT COMPUTING SCHEMES

Approval of Director of Graduate School of Applied Sciences

Prof. Dr. İlkay SALİHOĞLU

We certify that this thesis is satisfactory for the award of the degree of Master of Science in Electrical and Electronics Engineering

Examining Committee in Charge:

DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: **Nnamdi Ikechi Nwulu**

Signature:

Date: **June 15, 2011**

ABSTRACT

The increasing complex nature of today's power systems coupled with the strain the massive population growth is exerting on the system has been a major burden for power system operators and engineers. Typical reactions to these trends include system capacity expansion and load management. However load management has an added advantage over system capacity expansion due to its lower monetary cost and environmentally friendly nature. Amidst the load management measures adopted in recent years, demand management contracts have become a valued measure necessary for power system operations. In this thesis soft computing schemes or techniques are deployed for power system analysis and also for determining demand management contract values. Amidst the many soft computing schemes available, Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are utilized in this thesis. The major attraction or advantage soft computing schemes possess is their real time applicability coupled with fewer computational procedures and short processing times. Thus a soft computing scheme designed and trained on a specific power system would be able to give accurate values in real time even with a change in the power system. In this thesis, different soft computing architectures and topologies are designed and developed and these schemes are tested on various test power systems. Game Theory's mechanism design serves as the benchmark or "target" for our supervised soft computing models. For one of the experimental cases, a comparison of soft computing schemes is also drawn with a traditional least square regression method and obtained results show that soft computing methods are able to project demand management contracts values with relative accuracy and minimal computational expenses.

Keywords: Power System, Artificial Neural Networks, Support Vector Machines, Load Management, Least Square Regression Method, Soft Computing, Game Theory.

Dedicated to my family who have been with me through it all . . .

ACKNOWLEDGMENTS

I would like to begin by thanking the Almighty God who has been my help and the source of my strength through out the duration of my studies and for always helping me scale through even though I hardly leap high enough.

I am also indebted to my supervisor Assoc. Prof. Dr. Murat Fahrioglu for his total support and belief in me in the course of this work. His humane and kind disposition assisted in no small measure to the successful completion of this thesis.

I am grateful to Assist. Prof. Dr. Husseyin Sevay and Assist. Prof. Dr. Ali Serener for the never failing support, encouragement and assistance especially with the Latex end of the thesis and would also like to thank Prof. Dr. Adnan Khashman who introduced me to the world of soft computing,

I would like to thank 'Emengu' for the joyous times we spent and Vitalis and Simeon for their support.

Lastly and by no means the least i would want to thank my family for their unwavering belief in me and for their financial, moral and other kinds of support in ways impossible to enumerate.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Networks
SVM	Support Vector Machines
PSO	Particle Swarm Optimization
GA	Genetic Algorithms
ISO	Independent System Operator
IEEE	Institute of Electrical and Electronics Engineers
DSM	Demand Side Management
DG	Distributed Generation
OPF	Optimal Power Flow
LMP	Locational Marginal Price
LBMP	Locational Based Marginal Price

CHAPTER 1

INTRODUCTION

1.1 Overview

The ever increasing demand for electrical energy has put an upward pressure on energy serving utilities to seek for ways to increase energy supply. The major disadvantage of increasing energy generation capacity however lies in the fact that it usually involves massive capital investment with a commensurate increase in environmental hazards. It has therefore become imperative in recent years for electrical utilities to determine ways of satisfying consumers energy demand and doing so in an environmentally friendly fashion. Whilst satisfying the increasing demand levels, energy serving utilities or the Independent System Operator (ISO) also have to seek for ways of preserving and improving system security and easing transmission line bottlenecks. One of such ways is through demand management programs. Demand side management or demand management programs attempt to control and curtail a customers demand for electrical energy. Typically these programs are applied in time of power system stress or when the security of the power system is being threatened. Application ranges of demand management programs are either system wide or at specific problem prone spots on the electrical grid. Influencing and controlling a customers demand for electricity can be achieved either by peak clipping, valley filling, load shifting, strategic conservation, strategic load growth and flexible load shape (Gellings, 1985). In recent years demand management contracts have been introduced. A contract is defined as an agreement between utility and customer wherein the customer agrees to willingly shed load and in return receive monetary compensation. It should be noted that the monetary benefit might be in form of cash payments or reduced electricity tariffs and the customer might be willing to shed all of his load or have a limit to load shed. A crucial requirement for the successful implementation of demand management contracts is voluntary customer participation and to obtain voluntary customer participation, demand management contracts utilize some form of incentives or enticement hence they are also known as incentive compatible contracts (IEC).

Furthermore to efficiently design incentive compatible contracts it is necessary for the utility to accurately estimate customers outage costs as the incentive offered by the utility as a matter of necessity needs to be greater than the cost of interruption whilst simultaneously remaining profitable for the utility. This is a difficult task and Game theory's mechanism design has hitherto been used in designing optimal demand management contracts. There is still however a major need for a system that designs optimal demand management contracts with minimal computational complexity and faster processing times. Furthermore real time applicability is a major issue as there is the need for a system that can accurately project optimal contract values in real time as power system parameters change. This thesis posits that soft computing schemes can solve some of the inherent problems in present day demand management contract formulations and presents two prominent soft computing schemes : Artificial Neural Networks (ANN) and Support Vector Machines (SVM) for the design of optimal demand management contracts. In this thesis different topologies and configurations of ANN and SVM models are investigated and proposed. The objectives and contributions of this thesis can be summarized as shown in the next section.

1.2 Contribution

- The design and implementation of an Artificial Neural Network (ANN) based system for determining optimal demand management contract values. The developed system is capable of real time determination of optimal demand management contract values (real time processing) and has been tested on different IEEE test power systems.
- The design and implementation of a Support Vector Machine (SVM) based model for projecting optimal demand management contract values. The SVM system is also capable of real time processing with minimal computational and time overheads and is also implemented on an IEEE test power system.
- The investigation of different novel ANN and SVM topologies/ architectures that employ parallel processing in their design and implementation.
- The suggestion of 9 soft computing compatible input attributes for determining optimal demand management contract values. The input attributes suggested are representative of both engineering and economic factors since demand management programs have

both technical and economic considerations. Factoring both technical and economic indices into contract formulation makes for more accurate contract values

- Employing a novel output binary coding approach for the neural networks output. Instead of training the neural network with exact contract values, we employ binary coding for the neural model's output. Coding the output into binary values increases the synaptic weights at the neural network's output layer, thus improving the networks learning. The measure also has an added advantage as it adds flexibility to the neural model.

1.3 Thesis Overview

The remaining chapters of this thesis are organized as follows:

- *Chapter 2* reviews the latest research in demand management programs in general with a specific focus on the few designed with soft computing tools.
- *Chapter 3* reviews and describes the two soft computing schemes used in this thesis. The algorithms and their computational representations are presented.
- *Chapter 4* introduces optimal demand management contract design using Game theory's mechanism design which serves as the target or teacher for both developed soft computing schemes.
- *Chapter 5* presents the design of optimal demand management contracts using Artificial Neural Networks (ANN) and Support Vector Machines (SVM). An investigation is also provided into differing topologies and architectures. Also a comparison is provided with a traditional regression approach.
- *Chapter 6* concludes the thesis.
- *Chapter 7* presents the probable future development of the thesis

CHAPTER 2

REVIEW OF DEMAND MANAGEMENT PROGRAMS

2.1 Overview

Demand management programs attempt to influence and control a customers demand for electricity and can be achieved either by peak clipping, valley filling, load shifting, strategic conservation, strategic load growth and flexible load shape (Gellings, 1985). In this chapter a review is presented of prior demand management programs.

2.2 Demand Management Programs

Demand management programs are also known as demand response programs and have two major variants. There are incentive based programs and time based programs. Majority of the published works in the literature focus and utilize incentive based programs and for obvious reasons too. This is because in order to attract voluntary customer participation incentive based programs are preferred over time based programs. There exists a plethora of carefully designed demand management programs and schemes both in the academia and in industry and a brief literature review of some schemes is briefly provided :

2.2.1 Demand Management Programs Using Conventional Schemes

In (Aalami et al., 2010)the authors design two incentive based demand management programs. The designed programs are Interruptible/Curtail-able Services (I/C) and capacity market programs (CAP). For the design of these programs, the principle used is the price elasticity of demand and customer benefit function. The authors also incorporate penalties into the program formulation for customers who fail to respond to requests for load reduction. To indicate the suitability of the designed model, performance tests with encouraging results were performed on the Iranian power system. In (Lee and Yik, 2002) another incentive based demand management program is designed. Of particular interest to the authors of this work is the incentive (rebate) offered. Their designed rebate system is built

first by developing a performance curve. The performance curve actually models the relationship between the cost effectiveness and long-term benefits of different energy efficient demand management measures for commercial buildings in Hong Kong. A fundamental premise of the proposed rebate is that the adoption of extra measures by the customer should lead to a higher incentive or rebate which would lead to a diminished marginal rate of return. In a recent work (Paulus and Borggrefe, 2011) a comparison was made between different energy intensive industrial processes in Germany and the degree to which demand management programs for these industrial processes can provide tertiary reserve capacity. The economic and technical benefits of these industry process specific demand management programs are investigated to year 2030, a period Germany is expected to embrace renewable based electricity markets. Simulation results obtained indicate that demand management programs tailored to specific energy intensive industrial processes can provide approximately 50% of capacity reserves for the positive tertiary balancing market in year 2020. In another recent work (Moura and de Almeida, 2010) a study was conducted to determine the role demand management programs can play to ease the grid integration of wind power in Portugal. Due to the unpredictability and intermittent nature of wind power, its availability cannot be constantly guaranteed. Therefore if wind power is to be integrated into the grid, electrical utilities have to devise ways to efficiently match wind power availability to customer demand via demand management programs. In this work, the authors drew a comparison between the adoption of demand management measures and business as usual (BAU) measures in Portugal. Results of the study found out that applying demand management measures in Portugal can reduce the peak load demand by 17.4% in 2020. Recently in (Saffre and Gedge, 2010) a simulation was made on the feasibility of an efficient DSM strategy for smart grids. In this work the authors basically investigated the computing requirements for setting up such a system and the potential benefits accrueable. Furthermore they attempt to find the balance between efficiency and communication intensity in the network. It was discovered that DSM can be applied on a large scale in smart grid based system with manageable computational and communication overheads. In another recent work (Imbert et al., 2010) DSM amidst other system management approaches was applied to the Alpes Maritimes geographical region in Southern France. The authors also sought to determine the particular input data with the most effect on results. In other words input

sensitivity is computed. Two methods were used to compute input sensitivity: Monte Carlo analysis and percentage variation of base values. These two methods, it was discovered, obtained the optimum level of data necessary for efficient and accurate outputs. In (Fahrioglu et al., 2009)a system was designed where Distributed Generation (DG) complements demand management schemes. Economic analysis obtained in (Fahrioglu et al., 2009) indicate that utilities need not restrict themselves solely to demand management programs but can complement existing demand management schemes with distributed generation.

2.2.2 Demand Management Programs Using Soft Computing Tools

Soft computing schemes like Artificial Neural Networks (ANN) have also been used to design demand management programs. An example is in (Atwa et al., 2007) where the authors design a DSM strategy using an Elman artificial neural network that shifts the peak of the average residential electrical water heater power demand profile from periods of high demand to off peak periods. This strategy is structured by grouping water heaters in close proximity together into blocks and creating individual neural networks for each block. Conventional neural networks are ill suited for this type of problem since in this case the patterns vary over time; therefore the authors propose a dynamic neural network because of its temporal processing capabilities. Thus they make use of an Elman neural network. Simulation results show that each household would save $0.173259 per day per house if the network is deployed. Another example is in (Ravi et al., 2008) where a DSM strategy utilizing neural networks was proposed for an industry in India. Amidst the many DSM techniques proposed like End use equipment control, Load priority technique, peak clipping, valley filling and differential tariffs, Load priority technique was the DSM strategy settled upon The results obtained indicate that applying DSM strategies resulted in a reduction of electricity demand by 47.44kVA. A host of other approaches have also been used to design demand management programs. These include Particle Swarm Optimization (PSO) applied in Taiwan (Chen et al., 2009), Game theory (Fahrioglu and Alvarado, 2000) Genetic Algorithms (GA) and Monte Carlo stochastic simulation in (Wang, 2010), System dynamics in (Yang et al., 2006) and Market clearing based options (Zhang et al., 2005) to mention but a few.
It is obvious in light of prior works that demand management programs are useful for electric utilities and consumers alike. Both electric utilities and customers stand to benefit from the

adoption of demand management programs. Despite their demonstrated success, a major factor still hindering the widespread application of demand management programs is the complexity of many demand management contract schemes. Many advanced and computationally expensive schemes have been used to design demand management contracts like System dynamics in (Yang et al., 2006),Market clearing based options (Zhang et al., 2005), Monte carlo analysis (Zhang et al., 2005), (Imbert et al., 2010) that hinder them from being deployed in real time. There is the need for systems that can be deployed in real time without a compromise on accuracy. Furthermore although soft computing tools like ANN (Ravi et al., 2008). PSO (Chen et al., 2009) ,GA (Wang, 2010) and others have been used in designing various demand management programs, there has not been any attempt to develop demand management contract formulations using soft computing tools and in this thesis we present an application of various soft computing platforms for this task.

2.3 Summary

In this chapter, a brief review of demand management programs is provided and their practical applications in the literature. It is obvious that demand management programs are a practical and useful tool for any power system as it is beneficial both to the utility and customers and also has obvious environmental advantages. In the next chapter soft computing schemes are introduced and the process of data manipulation is described in detail.

CHAPTER 3

SOFT COMPUTING METHODS

3.1 Overview

Soft computing techniques are becoming increasingly applied in a wide range of fields and research areas. In this chapter a brief introduction to soft computing schemes is provided. The two soft computing schemes utilized in this work : Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are introduced and a detailed presentation of their computational procedures is provided.

3.2 Fundamentals of Soft Computing

The most concise definition of soft computing is that provided in (Li et al., 1998) which states that : ”Every computing process that purposely includes imprecision into the calculation on one or more levels and allows this imprecision either to change (decrease) the granularity of the problem, or to ”soften” the goal of optimization at some stage, is defined as to belonging to the field of soft computing”

The viewpoint that we will consider here (and which we will adopt in future) is another way of defining soft computing, whereby it is considered to be the antithesis of what we might call hard computing. Soft computing could therefore be seen as a series of techniques and methods so that real practical situations could be dealt with in the same way as humans deal with them, i.e. on the basis of intelligence, common sense, consideration of analogies, approaches, etc. In this sense, soft computing is a family of problem-resolution methods headed by approximate reasoning and functional and optimization approximation methods, including search methods. Soft computing is therefore the theoretical basis for the area of intelligent systems and it is evident that the difference between the area of artificial intelligence and that of intelligent systems is that the first is based on hard computing and the second on soft computing.

Soft computing is a research area in the computer science field which utilizes inexact solutions

to computationally difficult problems. The major difference between soft computing tools and hard computing techniques is that hard computing schemes on the one hand attempt to obtain exact solutions and 'full truth' while soft computing schemes thrive in regions of imprecision, uncertainty and 'partial truth'. A further difference between soft computing techniques and regular computing or 'hard computing' is that inductive reasoning is utilized more frequently in soft computing schemes than in hard computing. Generally speaking, soft computing techniques resemble biological processes more closely than traditional techniques Earlier computational approaches could model and precisely analyse only relatively simple systems. More complex systems often remained intractable to conventional mathematical and analytical methods. Soft computing deals with imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. The basic building blocks or components of soft computing are :

- Neural Networks
- Fuzzy Logic
- Evolutionary Computation
- Machine Learning
- Probabilistic Reasoning

A number of other soft computing schemes exist but they do not clearly fall under the afore mentioned blocks. Other major soft computing schemes are :

- Support Vector Machines
- Bayesian Networks
- Wavelets
- Fractals
- Chaos Theory

There is no hard and fast rule that would classify any single technique under soft-computing. However, there are some characteristics of soft-computing techniques which, taken together, serve to sketch the boundaries of the field.

Soft-computing, as opposed to hard computing, is rarely prescriptive in its solution to a problem. Solutions are not programmed for each and every possible situation. Instead, the problem or task at hand is represented in such a way that the state of the system can somehow be measured and compared to some desired state. The quality of the systems state is the basis for adapting the systems parameters, which slowly converge towards the solution.

3.3 Artificial Neural Networks (ANN)

Though no formal consensus exists among scientists about the definition of Artificial Neural Networks (ANN) one can say with certainty that it is modelled after the biological neural network in humans. The biological neural network in its simplest form can be defined as a set of interconnected neurons. Artificial Neural networks are therefore mathematical models that attempt to emulate the human biological neural system's structure and function. An artificial neural network typically consists of the following components:

- Input layer
- Output layer
- Hidden layer(s)
- Synaptic weights

Where each layer consists of a minimum of one neuron. Figure 3.1 shows the basic architecture of an artificial neural network.
A critical component of artificial neural networks is it's learning algorithm. Simply put an ANN's learning algorithm are explicitly defined and logical rules for network training. There exist many learning algorithms for neural networks all depending on the type of learning the network utilizes. Generally the mode of learning for the network determines the algorithm used. Learning in a neural network is of a wide variety and can be broadly summarized into the following classes:

- Supervised Learning: Synonymous to learning with a teacher. Here the network is presented with the desired output and it is the job of the network to find a way to process the given inputs to arrive at the desired output. Another name for supervised learning is error correction learning

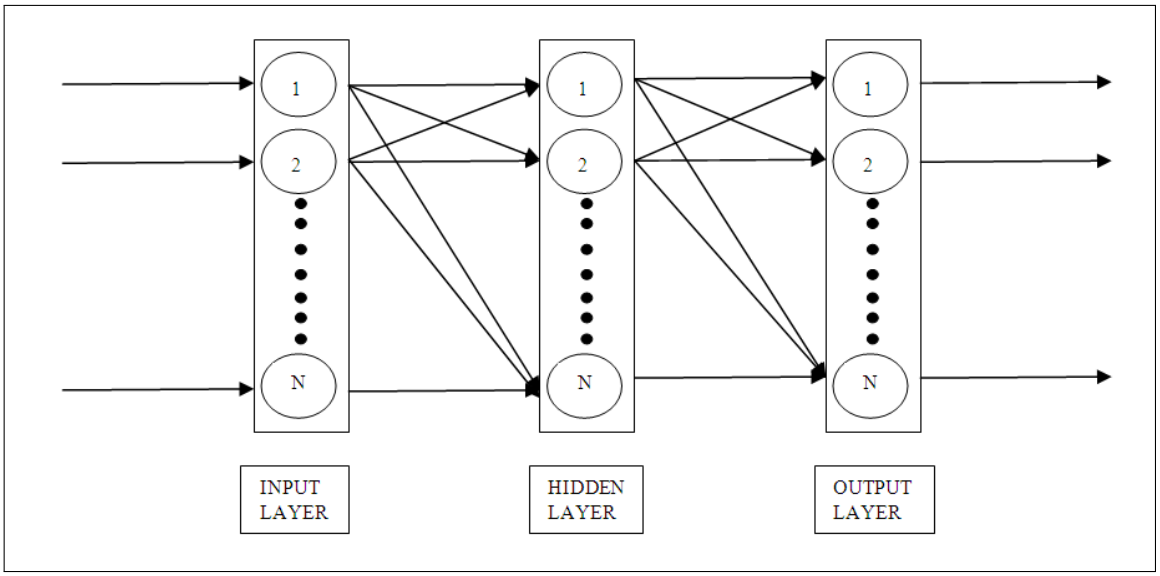

Figure 3.1: A Typical Neural Network Architecture

- Unsupervised Learning: Synonymous to learning without a teacher. The network is only given inputs and is left to generate its outputs.
- Semi- Supervised learning: This is a combination of both supervised and unsupervised learning.

The most popular learning algorithm is the back propagation learning algorithm and it is a supervised learner. It is the learning algorithm applied in this work. There are two sets of computations when applying the back propagation learning algorithm and they are briefly described below:

3.3.1 The Forward Computation

The back propagation algorithm is applied to each individual neuron in the neural network. A description is provided of the computations at each layer:

- INPUT LAYER (i): The input layer is not a processing layer, thus the output at each input layer neuron equals it's input

$$InputLayer'sOutput = O_i equals InputLayer'sInput = I_i. \tag{3.1}$$

- HIDDEN LAYER (h): The total input presented to a neuron at the hidden layer equals

the sum of the product of all outputs of the input layer neurons and their weights.

$$HiddenLayer'sInput = I_h = \sum_i W_{h_i} O_i. \tag{3.2}$$

The output of a neuron at the hidden layer is obtained via the sigmoid function given below:

$$HiddenLayer'sOutput = O_h = \frac{1}{1 + \exp(-I_h)} \tag{3.3}$$

- OUTPUT LAYER (j): The total input presented to a neuron at the output layer equals the sum of the product of all outputs of the hidden layer neurons and their weights.

$$OutputLayer'sInput = I_j = \sum_h W_{j_h} O_h. \tag{3.4}$$

The output of a neuron at the output layer like the hidden layer is similarly obtained via the sigmoid function given below:

$$OutputLayer'sOutput = O_j = \frac{1}{1 + \exp(-I_j)} \tag{3.5}$$

3.3.2 The Error Back Propagation Computation

The error back propagation computations are only applied in the training phase unlike the forward computations that are applied in both the training and testing stages. The error back propagation computation phase like the name implies consists of propagating the calculated error and simultaneously updating the weights. The error equation is given as:

$$E_p = \sum_{j=1}^{n_j} (T_{p_j} - O_{p_j})^2 \tag{3.6}$$

Where E_p is the error for a given pattern.(Subscript p denotes that the value is for a given pattern) and T_{p_j} is the expected output value a neuron should have popularly called the target value and O_{p_j} is the actual value resulting from the feed forward calculations. The error value is a measure of how well the training process performs. The aim of any neural network is to minimize the error value.

There is another important parameter known as the error signal Δ. The error signal for the

output layer Δ_j is defined as

$$\Delta_j = (T_j - O_j)O_j(1 - O_j) \tag{3.7}$$

While the error signal for the hidden layer Δ_h is defined as:

$$\Delta_h = O_h(1 - O_h)\sum_{j=0}^{n_j} W_{j_h}\Delta_j \tag{3.8}$$

The Δ is a very important parameter necessary for weight updates. Other necessary parameters are the learning coefficient (η) which is the degree of the networks learning ability and the momentum factor (α) which determines the speed of learning. These three parameters are highly essential for network learning and weight updates. The equations for updating network weights are given below: The weights are given random initial values before weight adjustment begins. Weights adjustment starts at the output layer and propagates backwards back to the input layer.

- HIDDEN LAYER WEIGHT UPDATE :

$$W_{h_i}(new) = W_{h_i}(old) + \eta\Delta_h O_i + \alpha[\delta W_{h_i}(old)] \tag{3.9}$$

  Where $\delta W_{h_i}(old)$ is the previous weight change

- OUTPUT LAYER WEIGHT UPDATE : The output layer weights W_{j_h} are updated using the following equation:

$$W_{j_h}(new) = W_{j_h}(old) + \eta\Delta_j O_h + \alpha[\delta W_{j_h}(old)] \tag{3.10}$$

  Where $\delta W_{j_h}(old)$ is the previous weight change

3.3.3 Basic Learning Procedure Using ANN

- Employ data pre processing via normalization and appropriate output coding (In this thesis all features are normalized to values between 0 - 1 using a novel approach in (Khashman, 2009) and (Khashman, 2010))

- Consider a suitable ANN algorithm and activation function

- Perform network training and error minimization while experimenting with various network parameters . Iterations are performed till goal error is met or maximum number of permissible iterations are performed.
- Test the trained ANN model on the testing dataset

Artificial neural networks are usually applied to non-linear problems and have been used in classification,regression,optimization and pattern recognition tasks. In power systems they have been applied with a high degree of success in areas like power electronics and drives (Bhattacharya and Chakraborty, 2011) ,(Kinhal et al., 2011) detecting and locating power quality disturbance (Weili and Wei, 2009), (Liao et al., 2010) , short term load forecasting (Chogumaira et al., 2010),(Chu et al., 2011) , recognising fault transients (Perera and Rajapakse, 2011), and Autoreclosure systems in transmission lines (Zahlay et al., 2011) to mention but a few. We present a novel application of neural networks to the design of optimal demand management contracts. In this work we use a single neural network model and also a double neural network model explained in subsequent chapters. Both neural network demand management contract design systems are based on the simple and yet efficient back propagation learning algorithm which gives instantaneous and accurate range of contract values. There are two major advantages of neural networks for problems of this kind. First is its suitability for real time deployment as a neural network trained on a particular power system would be able to give spontaneous demand management contract values as system parameters change. With present demand management contract schemes, once the system topology changes new contract values have to be computed. This is not the case with our neural network based system. Once a neural network is trained on a particular power system, it has learnt the inherent dynamics present in that system and thus it would be able to project instantaneous contract values even with a change in system topology. Secondly deploying neural networks for this task has the added advantage of minimizing computational complexity and reducing time costs. In (Saffre and Gedge, 2010) it was discovered that applying demand management programs also involves a significant amount of computation and communication overheads which might hamper the optimality of most programmes. There is thus the need for efficient demand management programs with high optimality and accuracy and also with minimal computational and time costs. Neural network based systems can provide this. Since the learning algorithm applied in this work is the back propagation

learning algorithm which is a supervised learner, there is need for a “teacher“ or target to benchmark the results obtained and we use game theory from mechanism design for this. Mechanism design (Fudenberg and Tirole, 1991) allows the utility with no information about its consumers decide how much to buy from its customers and the right price. The mechanism makes sure that the utility maximizes its benefit and at the same time ensures customers compensation attracts them to participate voluntary. In the next section, support vector machines another soft computing scheme is introduced.

3.4 Support Vector Machines (SVM)

Support Vector Machines (SVM) is one of the newest branches in soft computing. SVMs are supervised learners founded upon the principle of statistical learning. Unlike Neural Networks and other supervised learning tools, Support Vectors have the advantage of reducing the problems of over-fitting or local minima prevalent in Neural Networks. This is because learning in SVM is based on the structural risk minimization principle whereas in neural networks learning is based on the empirical risk minimization principle (Vapnik, 2000). SVM for the case of non linearly separable data works by non-linearly mapping the inner product of a feature space to the original space with the aid a kernel. When training in SVM, the solution of SVM is unique globally, and it is only dependent on a small subset of training data points which are referred to as support vectors. SVM is capable of learning in high-dimensional spaces with a small number of training examples and has high generalization ability.
There are two types of datasets where SVM classifiers can be successfully applied (Forsyth and Ponce, 2003).

- Linearly Separable Datasets
- Non Linearly Separable Datasets

For the second type of datasets (Non Linearly Separable Datasets), the ”kernel trick” is very often used. Kernels are useful for mapping vector instances in a set unto a higher dimensional space. This is useful for cases when the original data instance hyperplane doesn’t provide good classification results and thus requires a decision boundary with a more complex geometry (Forsyth and Ponce, 2003).

The computations for SVM (Forsyth and Ponce, 2003) are given below:

Suppose that we have a training set of N examples

$$[(x_1, y_1).....(x_N, y_N)] \tag{3.11}$$

where y_1 is either 1 or -1

In a linearly separable problem, there are values of w and b such that

$$y_i(w.x_i + b) > 0 \tag{3.12}$$

where w and b represent a hyperplane.

This can be formatted as a dual optimization problem where the aim is to maximize

$$\sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i(y_i y_j x_i.x_j)\alpha_j \tag{3.13}$$

subject to

$$\alpha_i \geq 0 \tag{3.14}$$

and

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.15}$$

where α_i is a Lagrangian multiplier introduced to ease the maximization problem. It is possible to determine w and it is given as

$$w = \sum_{1}^{N} \alpha_i y_i x_i \tag{3.16}$$

Any point x_i where α_i is non zero gives the following relation.

$$y_i(w.x_i + b) = 1 \tag{3.17}$$

and we can then determine the value of b.

New data points are classified as

$$f(x) = sign(w.x + b) \tag{3.18}$$

which becomes

$$sign((\sum_{1}^{N} \alpha_i y_i x.x_i) + b) \tag{3.19}$$

and can be rewritten as

$$sign(\sum_{1}^{N} (\alpha_i y_i x.x_i + b)) \tag{3.20}$$

Figure 3.2 shows an example of three different classifiers. It is obvious that classifier (line H3) doesnt separate the two classes, classifier (line H1) does, but the margin is not optimal. Classifier (line H2) presents the best results as it is at the maximum margin or exactly equidistant from the two patterns.

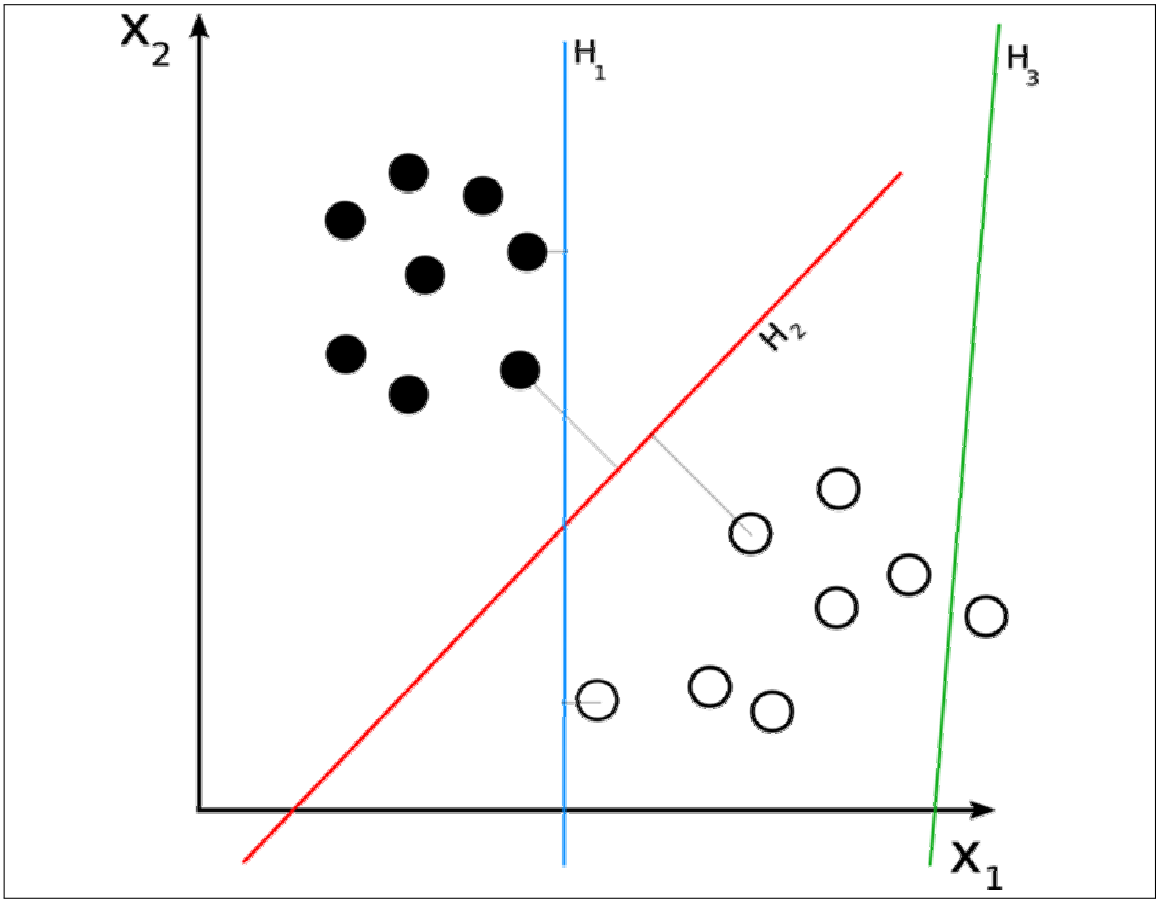

Figure 3.2: Examples of different kinds of classifiers on a linearly separable dataset. (Wikipedia, 2010)

Equation (3.20) provides an expression for a linearly separable dataset. However there might be a dataset that is not linearly separable and thus we need to map the feature vectors into a

new space and look for hyperplanes in the new space. Now suppose from equation (3.20) we introduce a map for x and x_i where $\phi(x)$=x and $\phi(x_i)$=x_i. Equation (3.20) becomes

$$sign(\sum_{1}^{N}(\alpha_i y_i \phi(x).\phi(x_i) + b)) \tag{3.21}$$

Lets assume that there is some function $k(x, y)$ positive for the paired x, y. It is possible to equate function $k(x, y)$ to $\phi(x).\phi(y)$. Thus instead of determining ϕ we determine an easier $k(x, y)$ and replace ϕ.

The optimisation problem then becomes

$$\sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i (y_i y_j k(x_i, x_j)) \alpha_j \tag{3.22}$$

subject to

$$\alpha_i \geq 0 \tag{3.23}$$

and

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.24}$$

and the decision classification function (classifier) is

$$sign(\sum_{1}^{N}(\alpha_i y_i k(x, x_i) + b)) \tag{3.25}$$

$k(x, y)$ is known as the kernel and it is only required that $k(x, y)$ be greater than zero for all values of x and y.

Figure 3.3 shows a graphical representation of kernel utilization and how it maps vector instances in a set unto a higher dimensional space There are four basic kernel types presently in use with SVM and they are given below:

- Linear Kernel
- Polynomial Kernel
- Radial Basis Function Kernel (RBF)
- Sigmoid Kernel

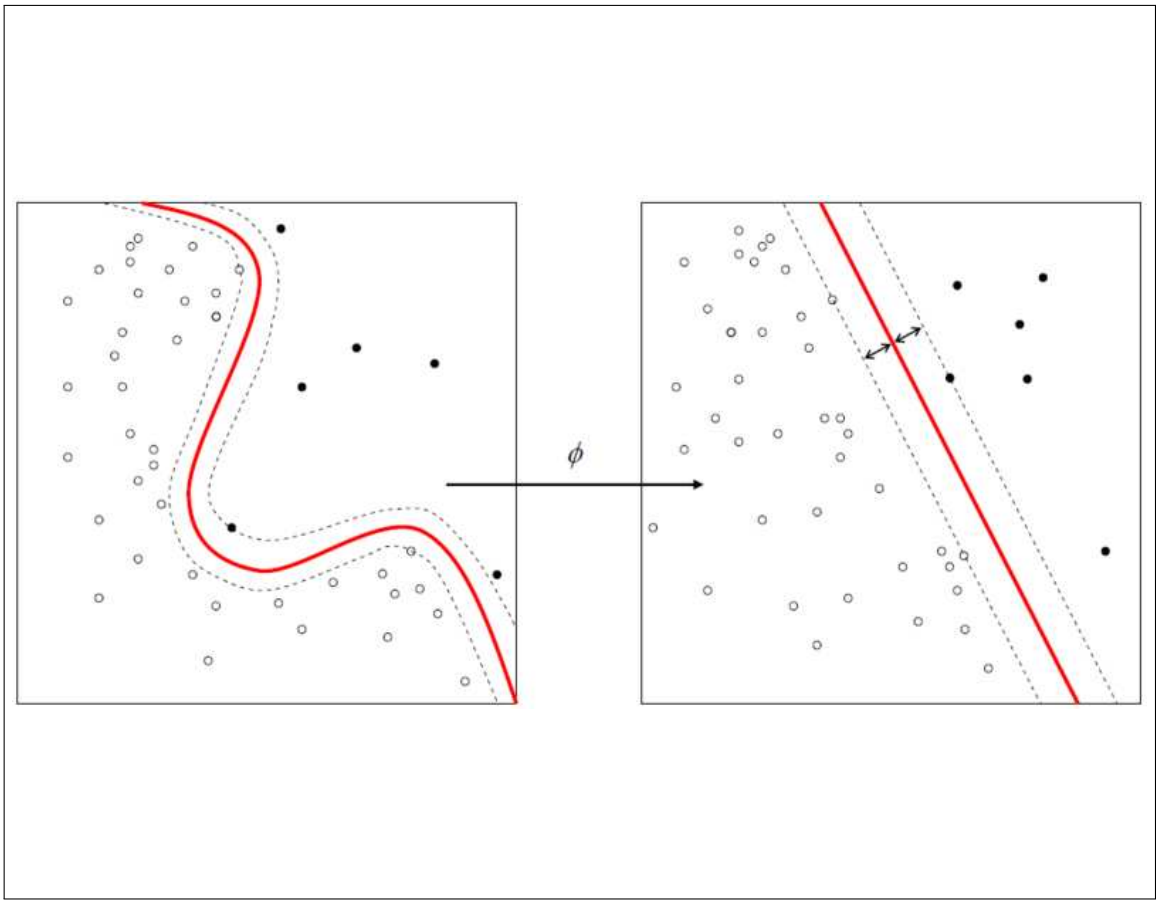

Figure 3.3: Graphical representation of Kernel Utilization for Non linearly separable data. (Wikipedia, 2010)

Any time an SVM model is developed in this thesis, it is the RBF Kernel used and the LIBSVM package (Chang and Lin, 2001) is used for implementation of SVM learning.
The equation for the RBF kernel is given by:

$$K(x, y) = exp(-\gamma||x - y||^2), \gamma > 0 \tag{3.26}$$

The major reason why we use the RBF kernel is because it has fewer numerical difficulties, possesses less hyper-parameters than other kernels and its ability to handle cases when the relationship between class labels and attributes is highly non-linear. To control generalization capability of SVM, the RBF kernel has two parameters: Gamma (γ) and C (cost parameter of the error term). Both C and γ should be greater than zero.
It should be noted that ANN also has two similar parameters that control generalization and also should be greater than zero. They have been defined earlier as (η) which is the degree of the networks learning ability and the momentum factor (α) which determines the speed of

learning.

In order to search for suitable parameters (C and γ) for our RBF kernel we perform a parameter search using cross validation specifically the v-fold cross validation method. The cross-validation procedure is a technique used to avoid the over fitting problem. In v-fold cross-validation, we first divide the training set into v subsets all with equal size. Sequentially one subset is tested using the SVM classifier trained on the remaining (v-1) subsets. Cross-validation accuracy is the percentage of data which are correctly classified. The parameters which produce the best cross validation accuracy are saved and then used to train the SVM learner. The saved model is then used on the out of sample data (testing set). Throughout the remainder of this work v=5.

3.4.1 Basic Learning Procedure Using SVM

- Pre-process your data by scaling (In this thesis all features are scaled to values between 0 - 1 using a novel approach in (Khashman, 2009) and (Khashman, 2010))
- Consider a suitable kernel (either RBF, sigmoid, polynomial or linear)
- Obtain the best parameters by cross validation (Best C and γ)
- Test the trained SVM model on the testing dataset

3.5 Summary

In this chapter soft computing as a major computational tool was introduced and its use in solving computationally difficult problems not suited for traditional or hard computing was highlighted. Moreover a detailed presentation of the theory and procedural implementation of two prominent soft computing schemes : Artificial Neural Networks and Support Vector Machines was described. Finally different reasons were adduced for applying soft computing schemes to power system operations. In the next section Game theory's mechanism design which serves as the teacher for the developed soft computing schemes is introduced and explained in detail.

CHAPTER 4

DEMAND MANAGEMENT CONTRACT FORMULATIONS

4.1 Overview

Game Theory's mechanism design serves as the target or teacher to the proposed soft computing demand management schemes. This means that results obtained from the different soft computing schemes are bench marked against results obtained from game theory in order to determine accuracy rates of the developed systems. In this chapter a brief review of demand management contract formulations using game theory is provided (reproduced with permission from (Fahrioglu and Alvarado, 2000)) and the general non linear nature of demand management contracts is presented. This chapter essentially reviews the formulations derived in (Fahrioglu and Alvarado, 2000) which is necessary to provide a backdrop for the soft computing formulations presented in subsequent chapters. It should be noted that the non linearity of demand management contracts make it suitable for soft computing tools.

4.2 Introduction to Non Linear Pricing

We begin with the assumption that the valuation of electricity by a consumer follows a declining marginal benefit and that the declining marginal benefit is a function of energy consumed (denoted by q). The marginal benefit can therefore be represented by the following equation :

$$b(q) = \theta(b_0 - sq) \tag{4.1}$$

Where θ can be said to be a parametric quantity depending on the customer and is scaled to the $0:1$ interval i.e. $0 < \theta < 1$. It can also be defined as the customer type. Furthermore b_0 represents the value of the first unit of electricity consumed and s represents how the marginal value of additional electricity consumed declines. The marginal benefit function for $b_0 = 1$, $s = 1$ and two values of θ is shown in Figure 4.1

The integral of this marginal benefit gives us the total benefit B. The total benefit for the

marginal benefit function described above is given by the following equation:

$$B(\theta, q) = \theta b_0 q - \frac{1}{2} s\theta q^2 \tag{4.2}$$

and the total benefit curves for each of the two customer types is given in Figure 4.2.

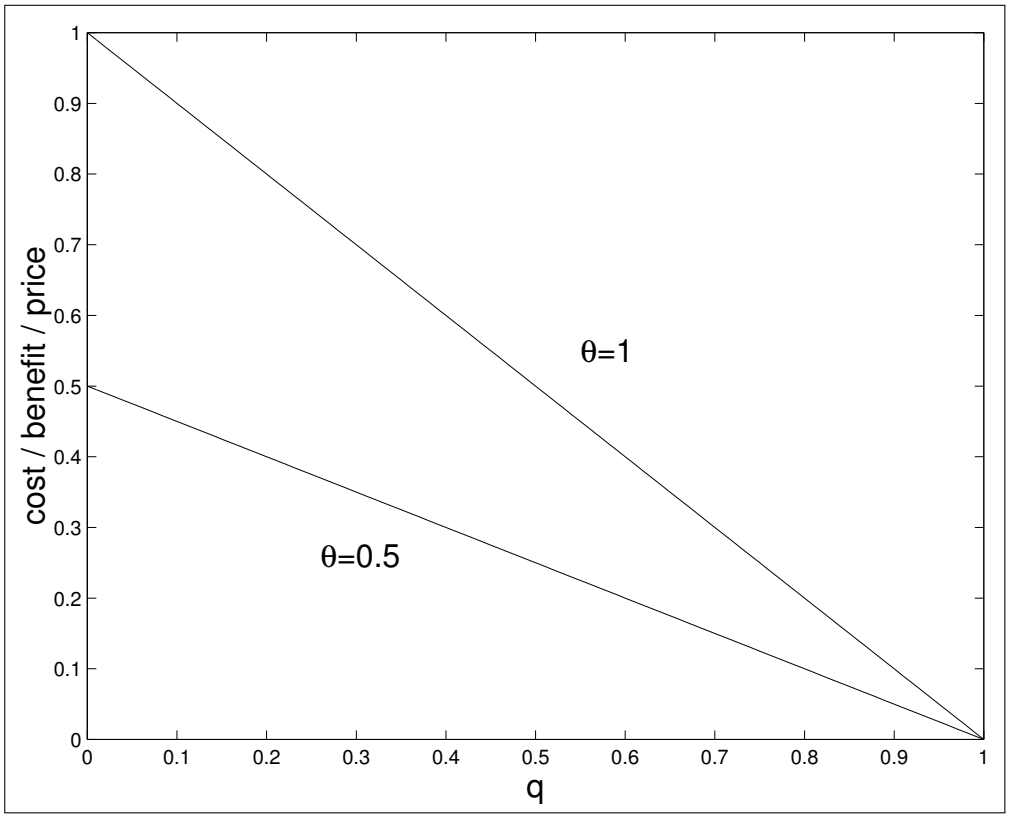

Figure 4.1: Marginal benefit for two customer types.

We assume that c is the cost (per hour) of producing electricity under a particular set of conditions. Furthermore we assume that the electrical utility considers only two kinds of customers it is desirous of selling electrical power to : A small customer it desires to sell quantity $\underline{q}$ and a large customer it is willing to sell quantity $\bar{q}$. $\underline{q}$ and $\bar{q}$ at this stage is yet to be determined and ($\underline{q} < \bar{q}$). The production cost for $\underline{q}$ is $c\underline{q}$ and likewise the production cost for $\bar{q}$ is $c\bar{q}$. Figure 4.2 shows the straight line defining these production costs. To sell at a profit, the electrical utility has to select price/quantity points on or above this line. If C_1 is the chosen price for quantity $\underline{q}$ and C_2 is the selected selling price for quantity $\bar{q}$ (shown in Figure 4.2). It is obvious that the utility would only be able to sell to the small customer if $B(\underline{\theta}, \underline{q}) \geq C_1$ and would also only be able to sell to the large customer if $B(\bar{\theta}, \bar{q}) \geq C_2$. This is visible from the figure and this condition is known as the *rationality constraint*.

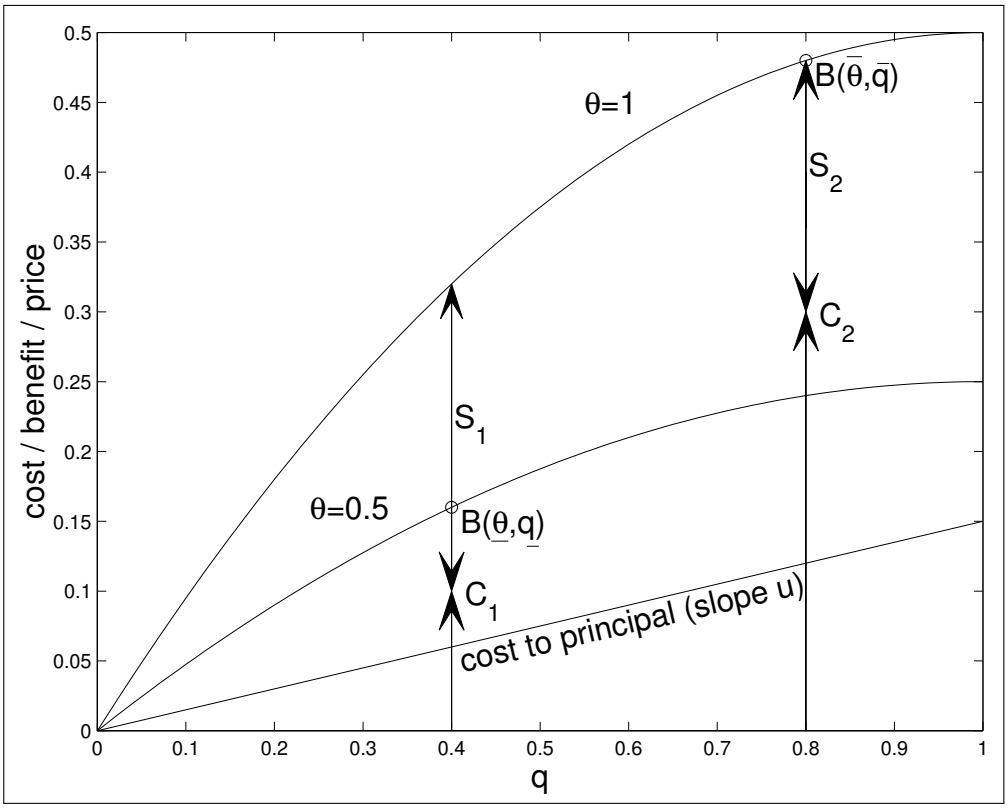

Figure 4.2: Total benefit, cost to producer and consumption levels for two customer types.

There is also another constraint although not as obvious as the first: Assuming the utility always charges prices that are close to $B(\bar{\theta}, q)$, the small consumer would thus be unable to use power as offering him power would mean the utility would operate at a loss. Lets assume that there is a minimum of one price/quantity offering that equals or is below curve $B(\underline{\theta}, q)$ (but above cq). This price condition is actually met with the $(C_1, \underline{q})$ offering. Suppose the large consumer chose a little amount of consumption ($\underline{q}$) the segment S_1 represents its total benefit as shown in Figure 4.2. Conversely, if the consumer were to choose the large amount ($\bar{q}$), its net benefit is represented by segment S_2. It is therefore logical in the light of the above to assume that if the large customer's benefit is greater when it consumes less (i.e. if $S_1 > S_2$), it is actually going to consume less. However this scenario is not favourable to the utility as it leads to highly sub optimal conditions. Pricing should therefore be structured in a way such that $S_1 \leq S_2$ for the larger customer. This condition can be termed the *incentive compatibility* condition. Figure 4.2 shows an instance that violates this condition which in turn tempts the customer/consumer to be dishonest or "lie". It can be proved mathematically that the conditions for optimality demand that the rationality condition determine the lower consumption/price point whilst the incentive compatibility condition determine the upper

price point.

In the event that there was only the large customer in the system, optimality is achieved when $\frac{dB(\bar{\theta}, q)}{dq} = c$ and in the event that there was only the small customer optimality is also achieved when $\frac{dB(\underline{\theta}, q)}{dq} = c$. Mechanism design is the preferred tool used to design optimal pricing structures whenever there is a cluster of different customers and there is uncertainty in the cluster.

4.2.1 Demand Management Contract Design Using Game Theory

It is obvious that the cost of curtailing power to a customer is dependent on the customer and the amount curtailed. Let's assume that the cost $c(\theta, x)$ of curtailing x MW from a customer of type θ is:

$$c(\theta, x) = K_1 x^2 + K_2 x - K_2 x\theta. \tag{4.3}$$

Where θ denotes the customer type (a continuous variable). The $-K_2 x\theta$ term enables different values of θ to give different values of $\partial c/\partial x$ (Customer's marginal cost). It should be pointed out that as θ increases the marginal cost decreases meaning that θ is used to differentiate customers willingness to shed load from the "least willing" to the "most willing". It is obvious from equation (4.3) that the customer with the lowest value of θ has the highest marginal cost which implies the lowest marginal benefit. Thus θ serves as a yardstick for ranking the *willingness* of customer's to curtail load.

For the sake of our analysis, we assume that the values of K_1 and K_2 are $1/2$ and 1 respectively. It should be noted that this assumption does not significantly affect our analysis as it is just equivalent to scaling. It is obvious that curtailment is valued differently by customers of different types. The customers interruption cost is given by equation (4.3), however the utility does not know the value of θ in this equation.

There are two other possibilities about the probability distribution of θ (denoted by $f(\theta)$) They are:

- The whole mix of customer types can be represented by varying θ from 0 to 1. It can also be assumed that the values of θ are random and uniformly distributed in the $[0, 1]$ range, meaning that there is an even chance that the customer will assume one of the customer types.

- θ values are discrete and these values have probabilities that are presumed. The simplest representation of this assumption is θ having two discrete values.

θ's value is only known to the customer and not the utility. Moreover the probability distributions of θ are subjective and it is not necessary for the utility to know the exact values of the probabilities of θ. The utility then has to derive an incentive function $y(x)$ in view of her estimate of customer types. The incentive function obtained details how much incentive the utility is prepared to pay willing customers for specific amounts of load curtailed.
The role of the customers is to examine the incentive function offered by the utility and then choose on their own volition the amount of load they are willing to curtail. For customers to embrace demand management programs willingly they have to see a net positive benefit accruing to them from program participation. A customers benefit function is given as:

$$V_1(\theta, x, y) = y - \frac{1}{2}x^2 - x + \theta x. \tag{4.4}$$

A net positive benefit implies that $V_1 \geq 0$ and this is imperative to ensure willing customer participation.
When the power system is stressed, it can become costly for the utility to deliver electric power to certain customers. The electric utility can therefore calculate how much it would cost to *not* deliver electric power to a certain location. This value represented as λ is termed the value of "power interruptibility" and it can be determined from optimal power flow procedures (Huneault and Galiana, 1991; Dommel and Tinney, 1968). A further important use of λ is enabling the utility determine its own benefit function when a specific customer curtails power. The utility's benefit function is given as:

$$V_2(\theta, x, \lambda) = \lambda x(\theta) - y(\theta) \tag{4.5}$$

λ is the (\$/MW) value of power not delivered to a location/customer. The utility's aim is to maximize its benefit function

$$\max_{x,y} \int_0^1 [\lambda x(\theta) - y(\theta)] f(\theta) d\theta \tag{4.6}$$

such that,

$$y(\theta) - \frac{1}{2}x^2(\theta) - x(\theta) + \theta x(\theta) \geq 0 \tag{4.7}$$

$$\begin{aligned} y(\theta) - \frac{1}{2}x^2(\theta) - x(\theta) + \theta x(\theta) \geq \\ y(\hat{\theta}) - \frac{1}{2}x^2(\hat{\theta}) - x(\hat{\theta}) + \theta x(\hat{\theta}) \end{aligned} \tag{4.8}$$

where $\hat{\theta}$ is an incorrectly reported customer type. The utility's benefit function is maximized in such a way that the customers are bound to choose their true type. There are two constraints inherent in this maximization problem. There is the *individual rationality constraint* (constraint (4.7)) which ensures that every customer is "tempted" to participate and the *incentive compatibility constraint* (constraint (4.8)) which compels customers to report their actual θ. Mechanism design's revelation principle in reference (Fudenberg and Tirole, 1991) provides a platform for solving this maximization problem and it gives the following results:

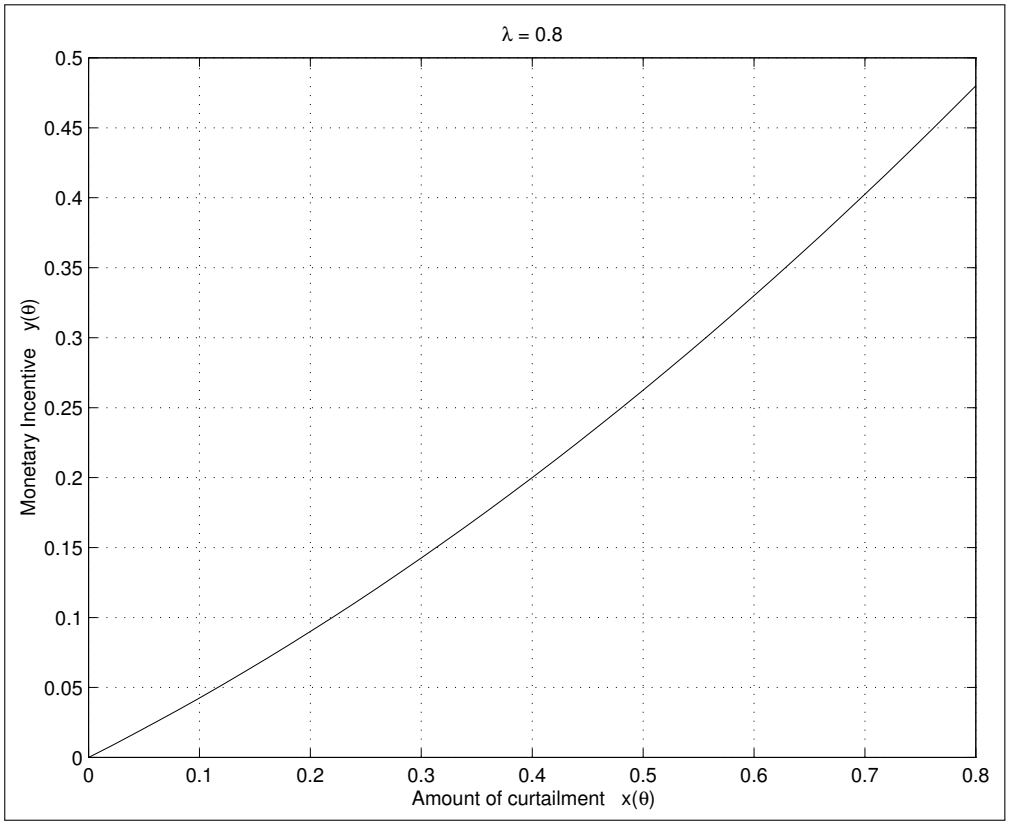

Figure 4.3: Designed Contracts

$$x(\theta) = \begin{cases} 0 & \text{if } 0 \leq \theta < 1 - \frac{\lambda}{2} \\ 2\theta + \lambda - 2 & \text{if } 1 - \frac{\lambda}{2} \leq \theta \leq 1 \end{cases} \tag{4.9}$$

$$y(\theta) = \begin{cases} 0 & \text{if } 0 \leq \theta < 1 - \frac{\lambda}{2} \\ \theta^2 - 2\theta + 2\theta\lambda & \\ +\frac{3}{4}\lambda^2 - 2\lambda + 1 & \text{if } 1 - \frac{\lambda}{2} \leq \theta \leq 1 \end{cases} \tag{4.10}$$

Equations (4.9) and (4.10) provides a definition of the demand management contracts to be offered to customers. A graph of the demand management contract values is shown in figure 4.3 which shows the utility's offered incentive as a function of the customer's load curtailed for a given value of λ. Figure 4.4 shows how the incentive offered varies as λ varies. As λ

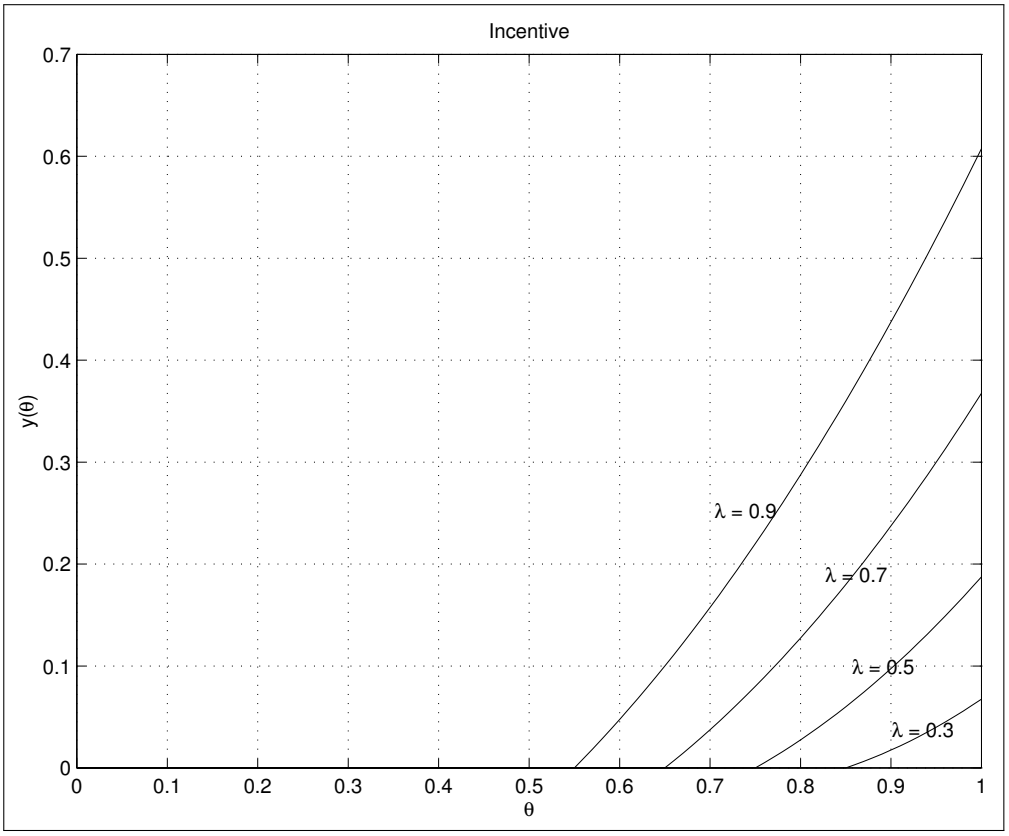

Figure 4.4: Normalized incentive function vs θ.

changes, it affects the number of participating customers. It is obvious from figure 4.4 that the incentive offered to customers is highly dependent on λ and this means that the utility would really seek to entice customers who have high λ's and thus are at costly locations to participate in the demand management programs. θ's role on the other hand represents the customer type which further determines their interruption costs.

4.2.2 Power System Sensitivity Analysis

To determine the value of interruptible power for the electric utility, sensitivity analysis is used. In (Greene et al., 1997), the sensitivity of the loading margin of a power system with respect to arbitrary parameters is computed. If we assume that loads are the parameters, one can compute the sensitivity of the loading margin with respect to individual loads. Let :

$$f(x, \gamma, p) = 0 \tag{4.11}$$

where γ is the vector of real and reactive load powers, x is the vector of state variables and p is the vector of loads. Suppose a unit vector k specifies a pattern of load increase, the left eigenvector w can be determined by the point of collapse method (Canizares and Alvarado, 1993). We define the sensitivity of the loading margin to a load change as:

$$\frac{\Delta L}{\Delta p} = L_p = \frac{-\omega f_p}{\omega f_\gamma k} \tag{4.12}$$

After computing the sensitivity of the loading margin to a load change, the loads in the system can be ranked. If we denote the loading margin of the system as L, equation (4.12) enables us to determine a relationship between the change in individual loads (Δp_1, Δp_2, etc) to change in security margin which is given as:

$$\Delta L = L_{p_1}\Delta p_1 + L_{p_2}\Delta p_2 + ... + L_{p_m}\Delta p_m \tag{4.13}$$

where m is the number of loads we are interested in. It is obvious from equation (4.13) that the load with the highest sensitivity would aid the most in increasing the system loading margin. The electric utility can use the dollar per kW values obtained from the designed contracts and the sensitivities from the sensitivity analysis to determine the cost of increasing system security given as:

$$\frac{\Delta L}{\Delta \$} = \frac{\Delta L}{\Delta p}\frac{\Delta p}{\Delta \$} \tag{4.14}$$

where $\Delta\$$ denotes the amount the utility will spend.

The importance of equation (4.14) is that it aids us in determining the cost of increasing the loading margin when one of the loads participating in the demand management scheme is curtailed.

4.3 Summary

Demand management contracts are essentially non linear in nature which make them suitable for soft computing schemes. However in the initial design of soft computing tools there is the need for a teacher or a target to benchmark results highlighting the importance of bench mark procedures. In this chapter an introduction was provided of game theory's mechanism design which acts as the benchmark for soft computing tools. In the next chapter the proposed ANN and SVM demand management contract soft computing schemes are presented.

CHAPTER 5

DEMAND MANAGEMENT CONTRACT FORMULATION USING SOFT COMPUTING SCHEMES

5.1 Overview

In this chapter, we lay out the formulations of determining optimal demand management contracts using soft computing techniques. Two different neural network models are implemented and tested on the IEEE 9 bus test power system and the IEEE 14 bus test power system. The double SVM model is tested on the IEEE 14 bus test power system and the experimental procedures and obtained results are presented. Finally a comparison of the results is drawn with a traditional regression approach.

5.2 Input Attributes for Soft Computing Models

Any soft computing models performance is very dependent on its input attributes, so care must be taken when choosing these values. In our work we chose the same values which served as input to the mechanism design and we also chose other key power system inputs values. Part of the novelty of this thesis is determining useful power system parameters that do not necessarily need to be fed to the game theory model , but would still deliver optimal and accurate contract values. The following were our final soft computing input attributes both for ANN and SVM:

- Amount of power curtailed (A1): Represented by x MW. At each of the nodes we assume that the customer is willing to curtail either all of their load or a fraction of their load.
- The value of power interruptibility (A2): Denoted by (λ). This is usually obtained from existing power flow routines. It represents the cost of not delivering power to a particular location and it is in U.S. dollars ($).
- Customer Type (A3) : Denoted by (θ) This is usually normalized between 0 and 1 and

serves as a differentiator of customer willingness to shed load. Most willing ((θ)=1) and least willing ((θ)=0).

- Cost of curtailing (A4): Denoted by $c(\theta, x)$. We assume a quadratic cost function.
- Bus voltage magnitude (A5): This is the voltage magnitude in per unit at each load node.
- Bus voltage phase angle (A6): This is the phase angle at each load node.
- Reactive Power (A7): The reactive power at each load node.
- Probability Function (A8): We make use of a probability function thats inversely proportional to the amount of power curtailed, the reasoning being that the probability of a customer curtailing a lot of power is low and vice versa.
- Case Number (A9): The total number of cases we have to determine contracts for. It is the product of the number of scenarios and the number of real loads in the power system.

5.3 Contract Formulations With Single ANN Model

The procedures for a contract formulations using a single ANN model were designed and tested in two experiments where experiment 1 involved the IEEE 9 bus test power system and experiment 2 involved the larger IEEE 14 bus test power system .

5.3.1 Experiment 1

Experiment 1 [1], involved the 9 bus example which has 3 generators and 3 loads as shown in Figure 5.1

The Locational Marginal Price (λ) is obtained from optimal power flow routines using MATPOWER (Zimmerman et al., 2009). A simulation is made of different scenarios and we obtain contract values using Game Theory for each scenario. The simulated scenarios are scenarios that any electricity utility might encounter in daily operations. For the purpose of this experiment, the simulated scenarios are given below.

[1]Experiment 1 was published in the 10th International Conference on Clean Energy (ICCE), September 15-17, 2010. Famagusta, North Cyprus as "Designing Optimal Demand Management Contracts Using An Artificial Neural Network". An expanded journal version was published in the International Review of Modelling and Simulations (IREMOS), Vol.4 No.1, February 2011 as "Power System Demand Management Contract Design : A Comparison between Game Theory and Artificial Neural Networks".

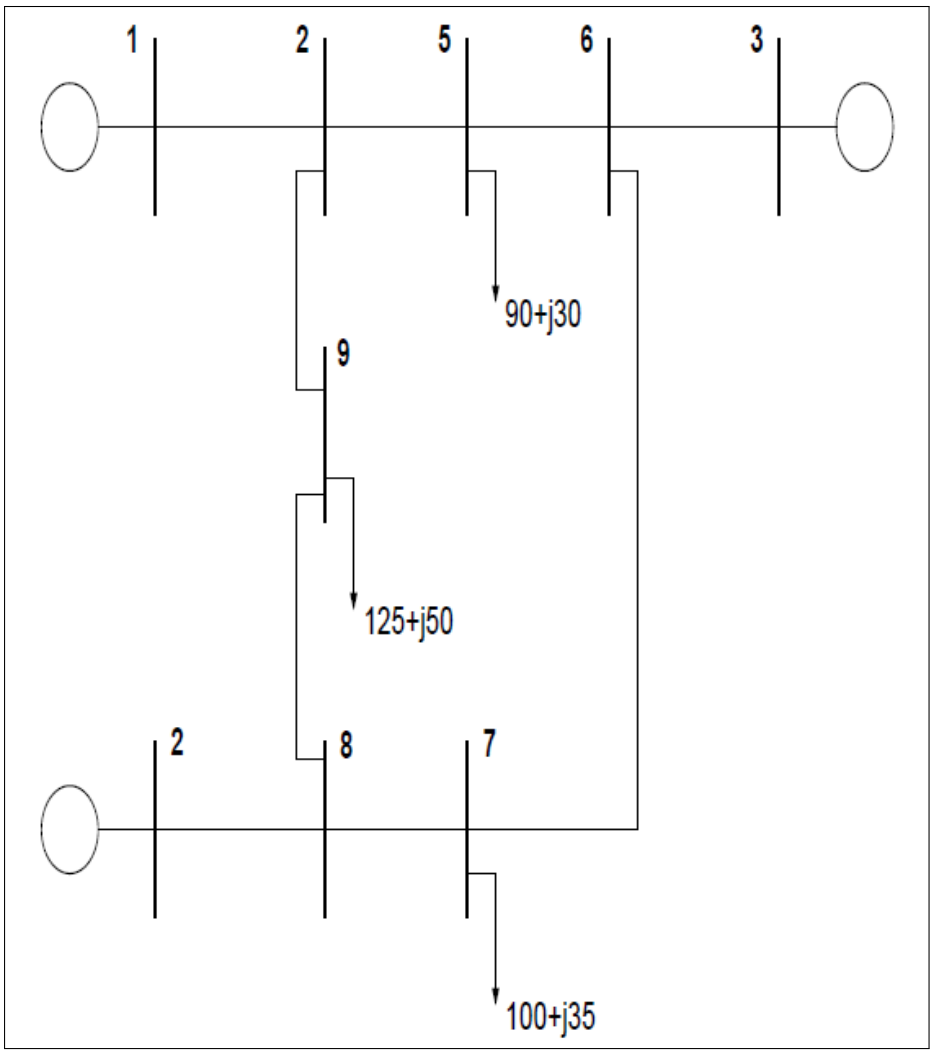

Figure 5.1: The IEEE 9 Bus Test System

Simulated Scenarios:

- All the generators and loads are connected
- All loads are connected and each generator is switched off
- All generators are connected and each of the loads is set to zero
- All generators are connected and each of the loads are curtailed to $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, $\frac{1}{5}$, $\frac{1}{6}$, $\frac{1}{7}$, $\frac{1}{8}$ and $\frac{1}{9}$ of their maximum capacity.

The range of the input attributes for experiment 1 are given below :

- A1 : In this experiment, the values range from 0 to 125MW.
- A2 : In this experiment, the values range from $15.558 to $41.087.
- A3 : In this experiment, the values are normalized between 0 and 1.
- A4 : In this experiment, the values range from 0 to $5134.
- A5: In this experiment, the values range from 1.031 to 1.097 per unit.
- A6: In this experiment, the values range from -9.099 to 7.031 degrees.
- A7: In this experiment, the values range from 30 to 50 MVAr.
- A8: In this experiment, the values range from 0.04 to 0.06
- A9 : In this experiment, there are 84 (28 x 3) cases

In the end there are a total of 84 different unique customer cases and we obtain contract values x (θ) which is the load curtailed and y (θ) which is the incentive paid for each load in the network (the assumption is that each load represents a customer).

Data Pre - Processing

In this phase, we basically perform two crucial tasks. Firstly the input attributes are normalized between 0 and 1 and secondly; we employ a suitable output coding (binary coding) for the contract values. The normalization technique we employ finds the largest number in each attribute and divides all the values in that attribute by the highest number. Table 5.1 shows the maximum values for the input attributes.

Table 5.1: Maximum Value For Each Input Attribute; Used To Normalize The Input Data Prior To Feeding Into The Neural Network

Input Attribute	A1	A2	A3	A4	A5	A6	A7	A8	A9
Maximum Value	125	41.0871	0.97	5134	1.097	7.031	50	0.06	84

Table 5.2 shows examples of the neural network input values prior to normalization; listing the first 10 cases.

Table 5.2: Examples Of The Pre- Normalization Input Attributes And Corresponding Contract Values For The First 10 Cases

A1	A2	A3	A4	A5	A6	A7	A8	A9	[KW]	[$]
90	24.99	0.10	173.8	1.08	-3.98	30	0.06	1	1000.0	12751.4
100	24.25	0.35	181.2	1.09	-1.19	35	0.05	2	973.8	12118.2
125	24.99	0.97	189.4	1.07	-4.62	50	0.04	3	1050.0	13956.6
90	36.11	0.10	173.8	1.05	0.34	5.8	0.06	4	1467.2	26857.7
100	33.86	0.35	181.2	1.09	7.03	7.5	0.05	5	1377.5	23767.9
125	36.22	0.97	189.4	1.03	-0.22	5.8	0.04	6	1521.9	28749.7
90	40.87	0.10	173.8	1.09	-6.37	5.0	0.06	7	1667.3	34886.7
100	40.67	0.35	181.2	1.08	-8.00	12.7	0.05	8	1664.2	34364.8
125	41.09	0.97	189.4	1.07	-9.09	16.6	0.04	9	1726.6	36880.3
90	34.46	0.10	173.8	1.08	-7.03	-3.9	0.06	10	1397.9	24337.2

The output coding presents some difficulty because the total contract output has to be load curtailed and incentive paid. To overcome this difficulty we use binary output coding. The output coding into binary values assist the neural network model in training, thus instead of training the neural network with two-digit target output value (load curtailed and incentive paid), we use a 26-digit binary code to represent the output result. This method results in using 26 output neurons (one neuron for each binary digit), which consequently increases the synaptic weights at the neural network output layer, thus improving its learning. 18 neurons for the load curtailed and 8 neurons for the incentive paid giving us a total of 26 neurons. Table 5.3 and Table 5.4 show the load curtailed and incentive paid interval with their corresponding binary coding. The two codes are concatenated to give the total number of neurons at the output layer.

Figure 5.2 shows the general topology of the optimal demand management contract single neural network model. The neural network input layer has 9 neurons, according to the number of the input attributes; where each input neuron receives a normalized attribute numerical value. There is one hidden layer containing 14 neurons, which was determined after several experiments involving the adjustment of the number of hidden neurons from one to 50 neurons, in order to assure meaningful learning while keeping the time costs to a

Table 5.3: Output Binary Coding For Load Curtailed

	Load Curtailed (kW)	Output Coding
1	0 - 100	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	101 - 200	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	201 - 300	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4	301 - 400	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	401 - 500	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
6	501 - 600	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
7	601 - 700	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
8	701 - 800	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
9	801 - 900	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
10	901 - 1000	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
11	1001 - 1100	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
12	1101 - 1200	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
13	1201 - 1300	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
14	1301 - 1400	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
15	1401 - 1500	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
16	1501 - 1600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
17	1601 - 1700	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
18	1701 - 1800	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Table 5.4: Output Binary Coding For Incentive Paid

	Incentive Paid ($)	Output Coding
1	0 - 5000	1 0 0 0 0 0 0 0
2	5001 - 10000	0 1 0 0 0 0 0 0
3	10001 - 15000	0 0 1 0 0 0 0 0
4	15001 - 20000	0 0 0 1 0 0 0 0
5	20001 - 25000	0 0 0 0 1 0 0 0
6	25001 - 30000	0 0 0 0 0 1 0 0
7	30001 - 35000	0 0 0 0 0 0 1 0
8	35001 - 40000	0 0 0 0 0 0 0 1

minimum. The output layer has 26 neurons according to the 26 intervals corresponding to 18 neurons for load curtailed and 8 neurons for incentive paid. During the learning phase, the learning coefficient, and the momentum rate were adjusted during various experiments; achieving an error value of 0.00092 which was considered as sufficient for this application. The final training parameters for this neural model were: learning coefficient (0.00648) and momentum rate (0.614). The initial weights of the neural network were randomly generated with values between -0.35 and +0.35.

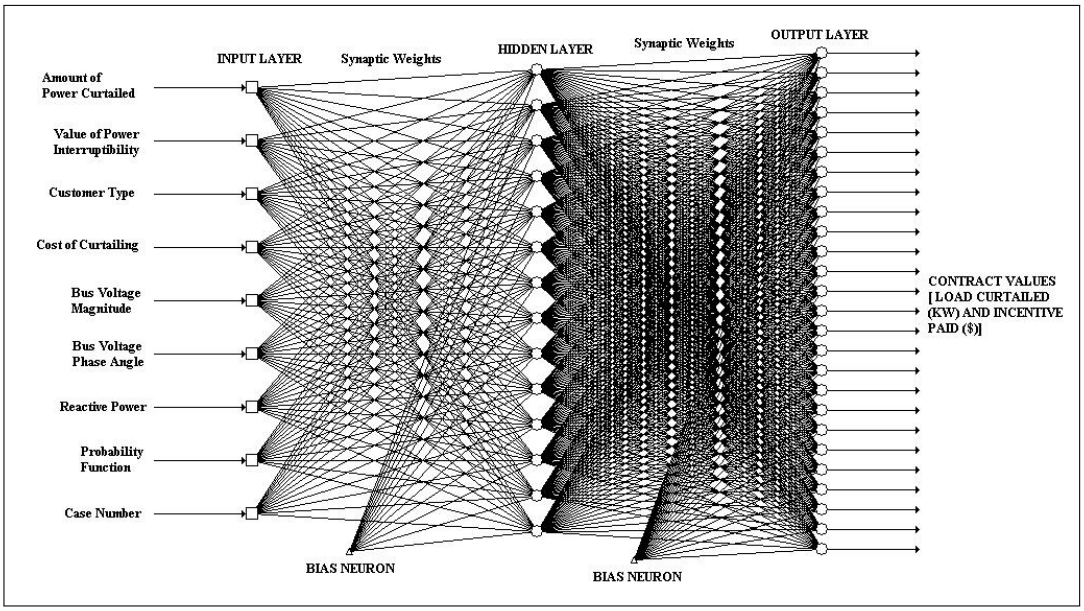

Figure 5.2: The Single Demand Management Contract Artificial Neural Network Architecture

Experimental Results and Implementation

The results of implementing the neural network model were obtained using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and MATLAB v 7.9.0 (R2009b). The neural network learned and converged after 7000 iterations and within 239.22 seconds, whereas the running time for the neural network after training and using one forward pass was $0.7\text{x}10^{-4}$ seconds. Table 5.5 lists the final parameters of the successfully trained neural network, and the accuracy rates. Figure 5.3 shows the error graph for the designed neural network

5.3.2 Experiment 2

Experiment 2 [2] involves the 14 bus example which has 5 generators and 11 loads shown in Figure 5.4

The methodology and procedures adopted in this simulation is similar to that adhered to in experiment 1, however it is a different power system and because the number of loads are more than in experiment 1 we have fewer number of simulated scenarios

Simulated Scenarios:

- All the generators and loads are connected

[2]Experiment 2 was published in the 10th International Conference on Environmental and Electrical Engineering (EEEIC), May 8-11, 2011. Rome, Italy. as "A Neural Network Model for Optimal Demand Management Contract Design".

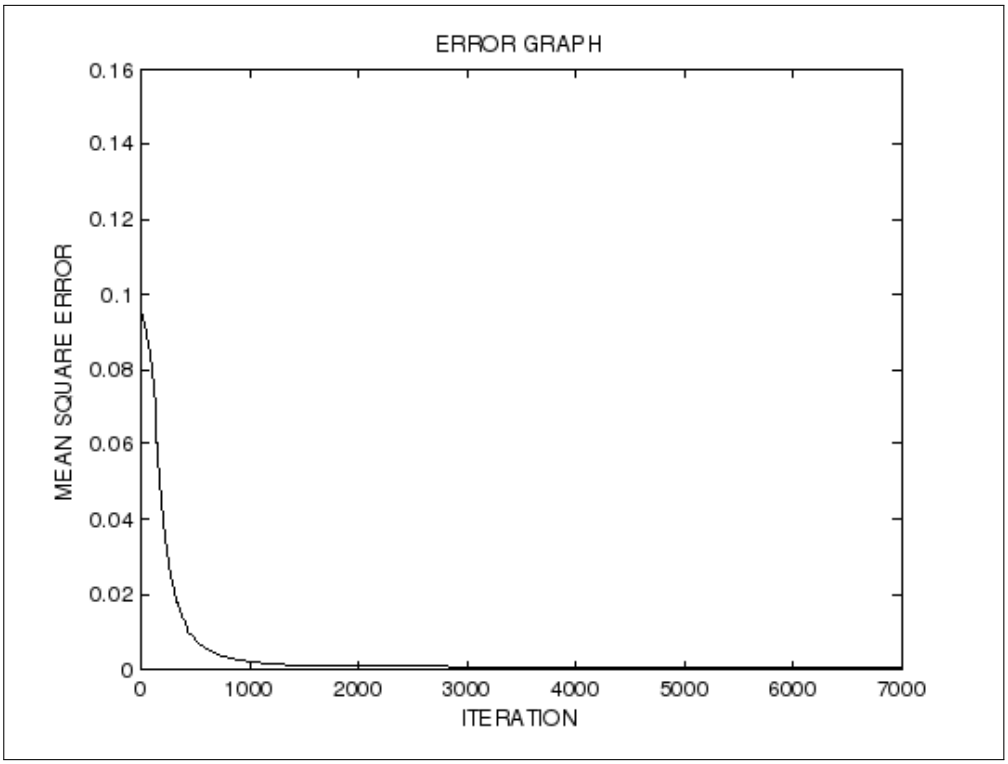

Figure 5.3: Mean Square Error vs Iteration Graph for Experiment 1

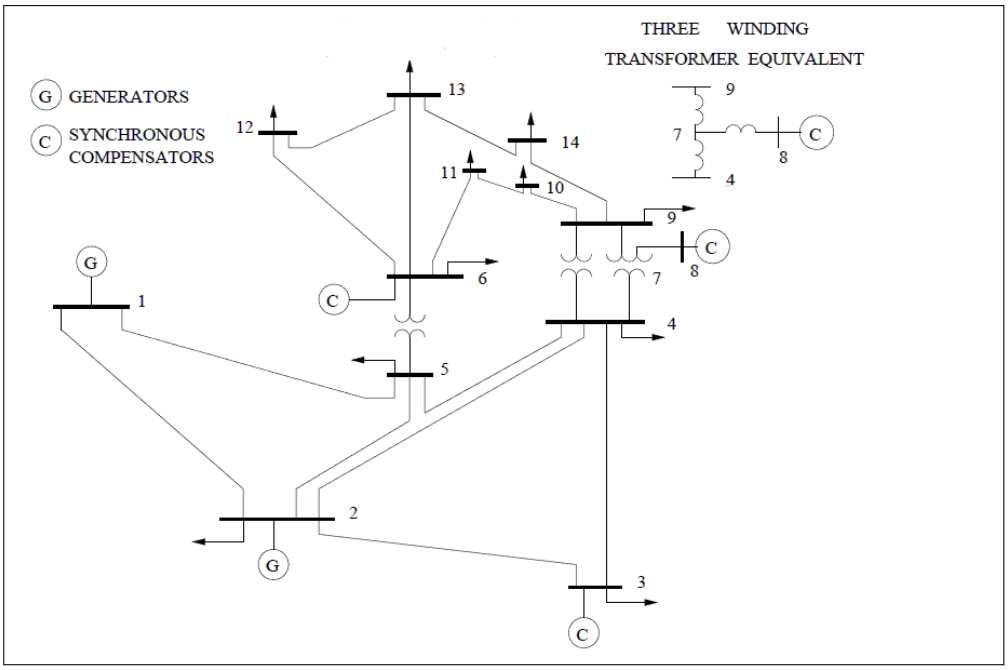

Figure 5.4: The IEEE 14 Bus Test System

Table 5.5: Final Neural Network Parameters for Experiment 1

Parameters	Numerical Values
Number of Input Neurons	9
Number of Hidden Neurons	14
Number of Output Neurons	26
Learning Co-efficient	0.00648
Momentum Rate	0.614
Minimum Required Error	0.00001
Obtained Error	0.00092
Maximum Allowed Iterations	7000
Number of Iterations Performed	7000
Training Time(seconds)[1]	239.22
Testing Time(seconds)[1]	0.00007
Training Dataset Accuracy Rate	91.42%
Testing Dataset Accuracy Rate	85.92%
Overall Dataset Accuracy Rate	88.67%

[1] Using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and MATLAB v $7.9.0(R2009b)$.

- All loads are connected and each generator is switched off
- All generators are connected and each of the loads is set to zero
- All generators are connected and each of the loads are curtailed to $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of their maximum capacity.

Thus there are 506 different unique customer cases and like the prior experiment contract values x (θ) which is the load curtailed and y (θ) which is the incentive paid for each load in the network (the assumption is that each load represents a customer) are obtained.
The range of the input attributes for experiment 2 are given below :

- A1 : In this experiment, the values range from 0 to 94.2MW.
- A2 : In this experiment, the values range from $37.34 to $42.71
- A3 : In this experiment, the values are normalized between 0 and 1.
- A4 : In this experiment, the values range from 0 to $158.22
- A5: In this experiment, the values range from 0.98 to 1.06 per unit.
- A6: In this experiment, the values range from -14.89 to 1.35 degrees.
- A7: In this experiment, the values range from -3.9 to 19 MVAr.

- A8: In this experiment, the values range from 0.02 to 0.07
- A9 : In this experiment, there are 506 cases

Experiment 2 proves the validity of our soft computing learning procedure as the single neural network model again delivered robust results even though it was a larger power system with more weights at the output layer. (Due to the 32 neurons at the output layer). Table 5.6 gives the final parameters of the trained neural model.

Table 5.6: Final Neural Network Parameters for Experiment 2

Parameters	Numerical Values
Number of Input Neurons	9
Number of Hidden Neurons	25
Number of Output Neurons	32
Learning Co-efficient	0.00517
Momentum Rate	0.73
Minimum Required Error	0.00001
Obtained Error	0.0021
Maximum Allowed Iterations	14000
Number of Iterations Performed	14000
Training Time(seconds)[1]	372.03
Testing Time(seconds)[1]	0.00008
Training Dataset Accuracy Rate	94.25%
Testing Dataset Accuracy Rate	87.14%
Overall Dataset Accuracy Rate	90.695%

[1] Using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and MATLAB v $7.9.0(R2009b)$.

5.4 Contract Formulations With Double ANN Model

The test power system in Experiment 2 is the same one utilized in this section. The difference being that it is a different neural network topology used here. We make use of a parallel processing algorithm and hence obtain the double neural network model. Instead of having a single neural network for load curtailed and incentive paid, we build two ANN models and feed them simultaneously with the input attributes. While one network is trained to determine the load curtailed, the other determines incentive paid.
Simulated Scenarios: The scenarios are similar to Experiment 2 and they are itemized below:

- All the generators and loads are connected
- All loads are connected and each generator is switched off

- All generators are connected and each of the loads is set to zero
- All generators are connected and each of the loads are curtailed to $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ of their maximum capacity.

The input attributes to the double ANN model is again the same as that in Experiment 2. We therefore have a total of 506 cases, we used training to testing ratio of 50%:50%. We choose not to use a higher training to testing ratio so that the neural network is not exposed to more training data than testing data. Therefore 253 cases are used for training the network, while the remaining 253 cases are used for testing. As stated before prior to using these observations with the neural network model, all values were normalized to numerical values between 0 and 1. The data pre-processing; i.e. attribute normalization as well as output data coding, have been explained before but will be revised again in the following section.

5.4.1 Data Pre - Processing

In this phase, we basically perform two crucial tasks. Firstly the input attributes are normalized between 0 and 1 and secondly; we employ a suitable output coding (binary coding) for the contract values. The normalization technique we employ finds the largest number in each attribute and divides all the values in that attribute by the highest number. Table 5.7 shows the maximum values for the input attributes.

Table 5.7: Maximum Value For Each Input Attribute; Used To Normalize The Input Data Prior To Feeding Into The Neural Network

Input Attribute	A1	A2	A3	A4	A5	A6	A7	A8	A9
Maximum Value	94.2	42.71	0.97	158.22	1.06	1.35	19	0.07	506

Table 5.8 shows examples of the neural network input values prior to normalization; listing the first 15 cases.

Since there are two neural network models, each model has its own output interval calibrated differently and hence the corresponding output binary coding differs. This arrangement ensures that there is parallel processing going on of the input attributes howbeit with different results. While the output interval for load curtailed is calibrated to 29 intervals, the interval for incentive paid is 23 intervals. Table 5.9 and Table 5.10 show the load curtailed and incentive paid interval with their corresponding binary coding.

Table 5.8: Examples Of The Pre- Normalization Input Attributes And Corresponding Contract Values For The First 15 Cases

A1	A2	A3	A4	A5	A6	A7	A8	A9	[KW]	[$]
3.5	40.26	0.10	11.17	1.05	-13.09	1.8	0.07	1	1095.9	23488.1
6.1	40.38	0.12	19.20	1.04	-13.53	1.6	0.06	2	1110.8	24082.4
7.6	39.66	0.14	23.73	1.01	-7.43	1.6	0.05	3	1091.3	23307.9
9.0	40.32	0.15	27.89	1.04	-13.23	5.8	0.04	4	1108.4	23986.6
11.2	39.73	0.17	34.30	1.06	-12.69	7.5	0.03	5	1091.3	23310.4
13.5	40.58	0.19	40.83	1.04	-13.58	5.8	0.03	6	1124.9	24650.9
14.9	41.19	0.21	44.72	1.02	-14.27	5.0	0.03	7	1128.4	24787.7
21.7	38.36	0.27	62.66	1.04	-4.02	12.7	0.03	8	1054.7	21916.0
29.5	40.17	0.35	81.34	1.04	-12.99	16.6	0.03	9	1110.4	24055.7
47.8	40.19	0.52	117.21	1.01	-8.66	-3.9	0.02	10	1107.4	23939.9
94.2	40.58	0.97	158.03	1.02	-9.93	19.0	0.02	11	1211.7	27990.0
3.5	41.62	0.10	11.17	1.04	0.70	1.8	0.07	12	1139.7	25262.6
6.1	41.82	0.12	19.20	1.04	0.46	1.6	0.06	13	1153.9	25849.1
7.6	41.34	0.14	23.73	1.02	-0.17	1.6	0.05	14	1141.6	25339.2
9.0	41.80	0.15	27.89	1.04	0.31	5.8	0.04	15	1152.8	25801.6

5.4.2 Neural Network Arbitration

During this phase we use a supervised neural network that is based on the back propagation learning algorithm due to its implementation simplicity and the availability of sufficient input attributes for training and testing this supervised learner. Figure 5.5 shows the general topology of the optimal demand management contract neural network model. It can be observed that the new architecture allows for parallel and simultaneous flow of information thereby increasing efficiency. We term the neural network for load curtailed NN1 and that for incentive paid NN 2 The neural network input layer has 9 neurons, according to the number of the input attributes; where each input neuron receives a normalized attribute numerical value. There is one hidden layer containing 25 neurons for NN1 and 17 neurons for NN2. This number of neurons was determined after several experiments involving the adjustment of the number of hidden neurons from one to 50 neurons, in order to assure meaningful learning while keeping the time costs to a minimum. The output layers have 29 neurons for NN1 and 23 neurons for NN2 where NN1 is the network for load curtailed and NN is the network for incentive paid. During the learning phase, the learning coefficient, and the momentum rate were adjusted during various experiments; achieving an error value of 0.009 for NN1 and 0.028 for NN2 which was considered as sufficient. The initial weights of the neural network were randomly generated with values between -0.35 and +0.35.

Table 5.9: Output Binary Coding For Load Curtailed

	Load Curtailed (kW)	Output Coding
1	965 - 975	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	976 - 985	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	986 - 995	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4	996 - 1005	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	1006 - 1015	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6	1016 - 1025	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7	1026 - 1035	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8	1036 - 1045	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9	1046 - 1055	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10	1056 - 1065	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11	1066 - 1075	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12	1076 - 1085	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13	1086 - 1095	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14	1096 - 1105	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15	1106 - 1115	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16	1116 - 1125	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
17	1126 - 1135	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
18	1136 - 1145	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
19	1146 - 1155	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
20	1156 - 1165	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
21	1166 - 1175	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
22	1176 - 1185	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
23	1186 - 1195	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
24	1196 - 1205	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
25	1206 - 1215	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
26	1216 - 1225	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
27	1226 - 1235	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
28	1236 - 1245	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
29	1246 - 1255	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

5.4.3 Experimental Results and Implementation

The results of implementing the neural network model were obtained using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and MATLAB v 7.9.0 (R2009b). NN1 learnt and converged after 13000 iterations and within 308.91 seconds while NN2 learnt and converged after 14000 iterations and within 299.6 seconds. Table 5.11 lists the final parameters of the successfully trained neural network, and the accuracy rates. Figure 5.6 shows the error graph for NN1 while Figure 5.7 shows the error graph for NN2.

The implementation results of the trained system were as follows: using the training attributes yielded an accuracy of 95.9% and 94.6% for NN1 and NN2 respectively. The testing of the trained neural model using the testing attributes yielded an accuracy of 84.01% and

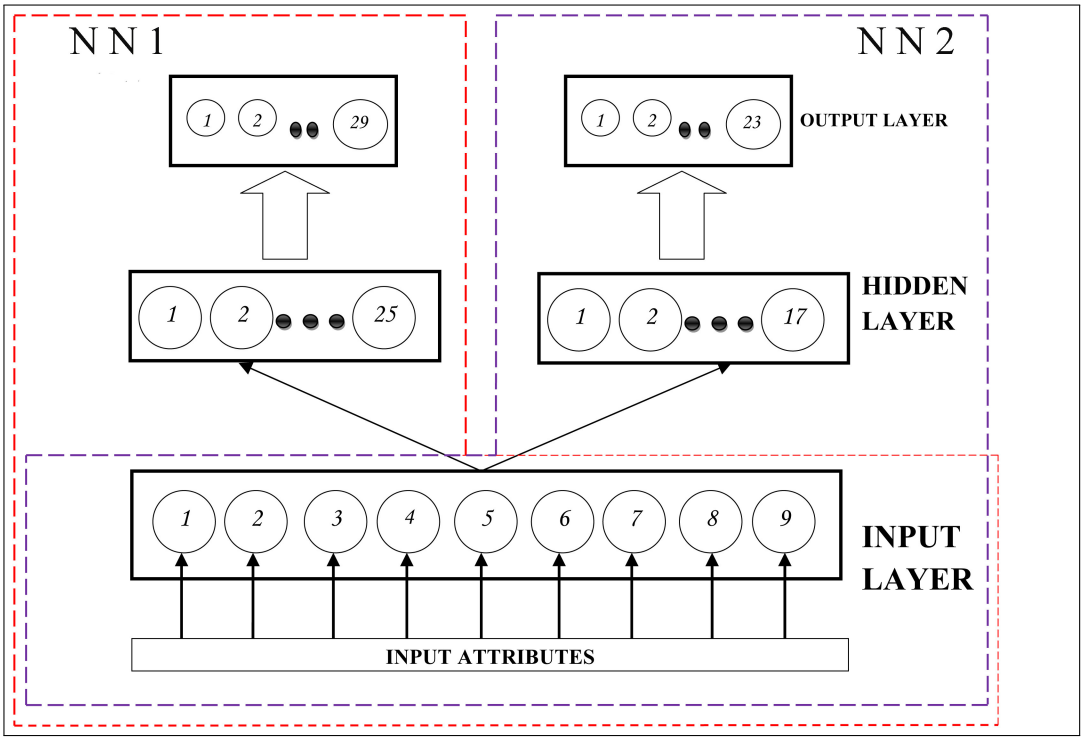

Figure 5.5: The Double Demand Management Contract Neural Network Architecture

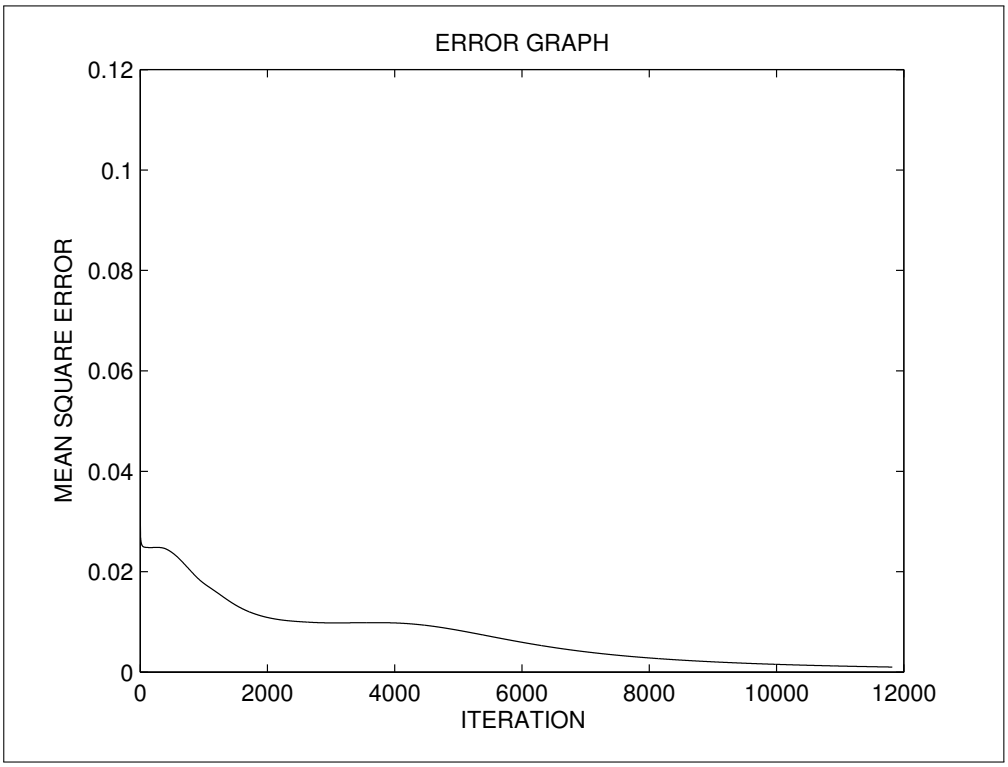

Figure 5.6: Mean Square Error vs Iteration Graph for NN1

Table 5.10: Output Binary Coding For Incentive Paid

	Incentive Paid ($)	Output Coding
1	18600 - 19100	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	19100 - 19600	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	19600 - 20100	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4	20100 - 20600	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	20600 - 21100	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6	21100 - 21600	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7	21600 - 22100	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8	22100 - 22600	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9	22600 - 23100	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10	23100 - 23600	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
11	23600 - 24100	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
12	24100 - 24600	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
13	24600 - 25100	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
14	25100 - 25600	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
15	25600 - 26100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
16	26100 - 26600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
17	26600 - 27100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
18	27100 - 27600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
19	27600 - 28100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
20	28100 - 28600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
21	28600 - 29100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
22	29100 - 29600	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
23	29600 - 30100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

82.58% for NN1 and NN2 respectively. Combining the training and testing accuracy results yields an overall accuracy rate of 89.955% and 88.59% for NN1 and NN2 respectively.

5.5 Contract Formulations With SVM

The SVM model is quite similar to the developed ANN model. However unlike the ANN model that used output binary coding the SVM model defines classes for the output. The data pre processing steps are essentially the same so they wont be repeated here. However the contract interval and corresponding output classes are given in Table 5.12 for load curtailed and Table 5.13 for incentive paid.

The results of implementing the SVM model were obtained using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and LIBSVM 2.9.1 The kernel used in this work is the RBF kernel. As stated before the reason for the RBF kernel is because it has fewer numerical difficulties, possesses less hyper-parameters than other kernels and its ability to handle cases when the

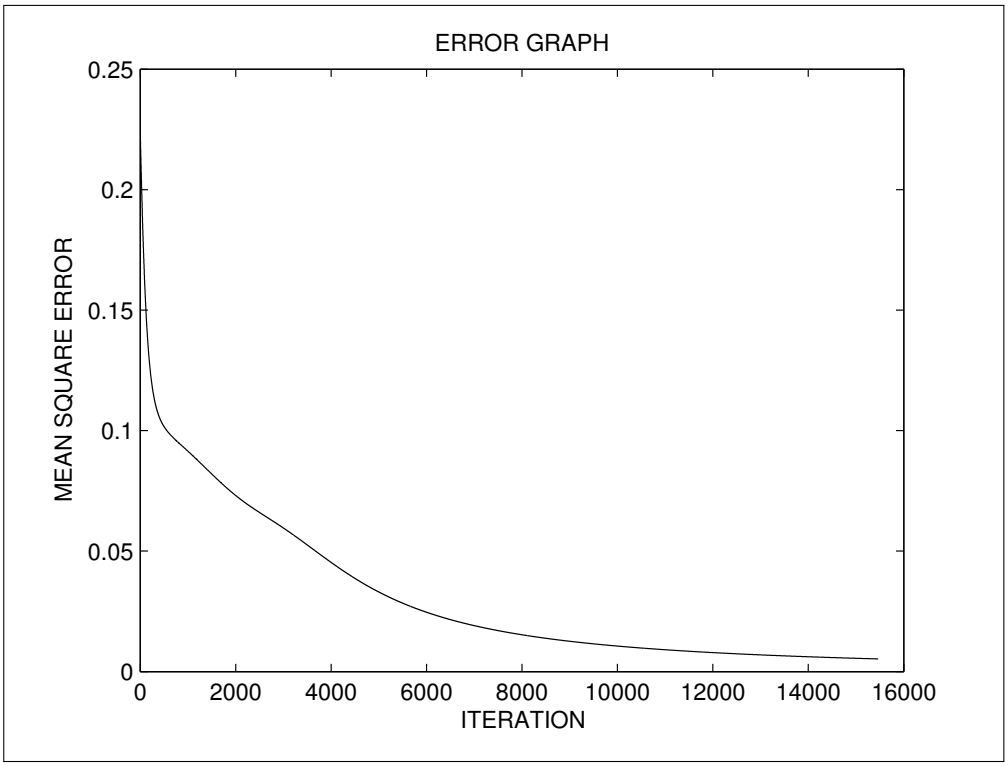

Figure 5.7: Mean Square Error vs Iteration Graph for NN2

relationship between class labels and attributes is highly non-linear. The RBF kernel has two parameters: Gamma (γ) and C (cost parameter of the error term). In order to search for suitable parameters (C and γ) for our RBF kernel we perform a parameter search using cross validation specifically the v-fold cross validation method where in this work v=5. In v-fold cross-validation, we first divide the training set into v subsets all with equal size. Sequentially one subset is tested using the SVM classifier trained on the remaining (v 1) subsets. Cross-validation accuracy is the percentage of data which are correctly classified. The parameters which produce the best cross validation accuracy are saved and then used to train the SVM learner. The saved model is then used on the out of sample data (testing set). The implementation results of the trained system were as follows: using the training attributes yielded an accuracy of 91.7% and 88.32% for SVM Model 1 and SVM Model 2 respectively. The testing of the trained neural model using the testing attributes yielded an accuracy of 86.68% and 82.619% for SVM Model 1 and SVM Model 2 respectively. Combining the training and testing accuracy results yields an overall accuracy rate of 89.19% and 85.4695% for SVM Model 1 and SVM Model 2 respectively. It is obvious that the SVM model gives a different result from the ANN models and this is probably due to the structural risk minimization

Table 5.11: Double Neural Network Final Parameters

Parameters	Neural Network 1	Neural Network 2
Number of Input Neurons	9	9
Number of Hidden Neurons	25	17
Number of Output Neurons	29	23
Learning Co-efficient	0.00312	0.00458
Momentum Rate	0.6	0.8
Minimum Required Error	0.001	0.001
Obtained Error	0.009	0.028
Maximum Allowed Iterations	25000	25000
Number of Iterations Performed	13000	14000
Training Time(seconds)[1]	308.91	299.6
Testing Time(seconds)[1]	0.00007	0.000052
Training Dataset Accuracy Rate	95.9%	94.6%
Testing Dataset Accuracy Rate	84.01%	82.58%
Overall Dataset Accuracy Rate	89.955%	88.59%

[1] Using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and MATLAB v $7.9.0(R2009b)$.

principle of SVM as opposed to ANN's error risk minimization approach. Table 5.14 gives the final parameters of the trained double SVM model.

5.6 Contract Formulations With Traditional Regression Approach

In this section, a regression approach is applied to the demand management contract formulation. The aim is to compare obtained experimental results and furthermore to determine out of the 8 input attributes (attribute 9 is dropped) the attribute with the greatest effect on demand management contract values. A simple least square regression approach is used and implemented in the E-Views regression software environment. The input values are essentially the same with the double ANN model (experiment 2) and the double SVM model. Also regression analysis is performed for both load curtailed(kW) and incentive paid ($). Table 5.15 - Table 5.18 detail the results obtained from the regression analysis. It can be noted from the analysis that the regression model gives acceptable error metrics for the load curtailed case and extremely high error metrics for the incentive paid category. This is an area where soft computing models triumph over regression analysis. Another critical observation is in the t- statistic for each input variable in Table 5.15 and Table 5.17. The t- statistic enables us to determine the input attribute with the greatest effect on demand management contract values. It is obvious from both tables that value of power interruptibility (A2), denoted by (λ)

Table 5.12: Output Class Interval For Load Curtailed

	Load Curtailed (kW)	Output Class Interval
1	965 - 975	1
2	976 - 985	-1
3	986 - 995	2
4	996 - 1005	-2
5	1006 - 1015	3
6	1016 - 1025	-3
7	1026 - 1035	4
8	1036 - 1045	-4
9	1046 - 1055	5
10	1056 - 1065	-5
11	1066 - 1075	6
12	1076 - 1085	-6
13	1086 - 1095	7
14	1096 - 1105	-7
15	1106 - 1115	8
16	1116 - 1125	-8
17	1126 - 1135	9
18	1136 - 1145	-9
19	1146 - 1155	10
20	1156 - 1165	-10
21	1166 - 1175	11
22	1176 - 1185	-11
23	1186 - 1195	12
24	1196 - 1205	-12
25	1206 - 1215	13
26	1216 - 1225	-13
27	1226 - 1235	14
28	1236 - 1245	-14
29	1246 - 1255	15

has the greatest effect on both load curtailed and incentive paid as it has the highest t-statistic of about 25. The other input attributes that have the greatest effect on contract values are Reactive Power (A7), Amount of power curtailed (A1), Probability Function (A8) and Bus voltage phase angle (A6)

5.7 Summary

In this chapter, the design of optimal demand management contracts using soft computing techniques was spelt out. The proposed models were implemented and tested on the IEEE 9 bus test power system and the IEEE 14 bus test power system and the obtained results presented. Further more, an attempt is made to compute the same contract values with a least

Table 5.13: Output Class Interval For Incentive Paid

	Incentive Paid ($)	Output Class Interval
1	18600 - 19100	1
2	19100 - 19600	-1
3	19600 - 20100	2
4	20100 - 20600	-2
5	20600 - 21100	3
6	21100 - 21600	-3
7	21600 - 22100	4
8	22100 - 22600	-4
9	22600 - 23100	5
10	23100 - 23600	-5
11	23600 - 24100	6
12	24100 - 24600	-6
13	24600 - 25100	7
14	25100 - 25600	-7
15	25600 - 26100	8
16	26100 - 26600	-8
17	26600 - 27100	9
18	27100 - 27600	-9
19	27600 - 28100	10
20	28100 - 28600	-10
21	28600 - 29100	11
22	29100 - 29600	-11
23	29600 - 30100	12

square regression method and results obtained from the least square method indicate the input attributes with the greatest effect on contract values.

Table 5.14: Double SVM Model Final Parameters

Parameters	SVM Model 1	SVM Model 2
Number of Features	9	9
Number of Classes	29	23
C Search Range	2^{-100} to 2^{100}	2^{-100} to 2^{100}
γ Search Range	2^{-100} to 2^{100}	2^{-100} to 2^{100}
C	512	2048
γ	32	4
V	5	5
Kernel	RBF	RBF
Training Optimization Time	0.692 seconds	0.55 seconds
Training Dataset Accuracy Rate	91.70%	88.32%
Testing Dataset Accuracy Rate	86.68%	82.619%
Overall Dataset Accuracy Rate	89.19%	85.4695%

[1] Using a 2.2 GHz PC with 2 GB of RAM, Windows XP OS and LIBSVM v 2.9.1.

Table 5.15: Parameters of the Least Square Regression Model for Load Curtailed

Variable	Coefficient	Std.Error	t-Statistic	Prob.
C	-100.7104	104.0273	-0.968116	0.3339
A5	-36.14365	78.07535	-0.462933	0.6438
A6	0.048425	0.236409	0.204835	0.8379
A1	2.377338	0.877388	2.709562	0.0072
A2	30.64947	1.188373	25.79112	0.0000
A7	0.860281	0.174288	4.935985	0.0000
A3	-41.45827	90.04355	-0.460425	0.6456
A8	180.7992	92.61940	1.952066	0.0521
A4	-0.626952	0.100577	-6.233532	0.0000
R-squared	0.908303	Mean dependent var	1113.225	
Adjusted R-squared	0.905297	S.D. dependent var	40.69416	
S.E. of regression	12.52318	Akaike info criterion	7.927965	
Sum squared resid.	38266.55	Schwarz criterion	8.053659	
Log likelihood	-993.8876	F-statistic	302.1176	
Durbin-Watson stat	2.857954	Prob(F-statistic)	0.000000	

Table 5.16: Error Values and other parameters of the least square regression model for load curtailed (kW)

Parameters	Numerical Values
Root Mean Square Error	12.298429
Mean Absolute Error	7.604885
Mean Abs. Percent Error	0.690207
Theil Inequality Coefficient	0.005520
Bias Proportion	0.000000
Variance Proportion	0.024040
Covariance Proportion	0.975960

Table 5.17: Parameters of the Least Square Regression Model for Incentive Paid

Variable	Coefficient	Std.Error	t-Statistic	Prob.
C	-24453.93	4129.284	-5.922076	0.0000
A5	-1179.900	3099.142	-0.380718	0.7037
A6	7.226620	9.384090	0.770093	0.4420
A1	60.72499	34.82726	1.743605	0.0825
A2	1217.492	47.17156	25.80986	0.0000
A7	33.97664	6.918212	4.911189	0.0000
A3	1625.490	3574.211	0.454783	0.6497
A8	7090.022	3676.457	1.928493	0.0550
A4	-24.13231	3.992343	-6.044650	0.0000
R-squared	0.905301	Mean dependent var	24186.82	
Adjusted R-squared	0.902196	S.D. dependent var	1589.510	
S.E. of regression	497.0983	Akaike info criterion	15.29038	
Sum squared resid.	60294049	Schwarz criterion	15.41607	
Log likelihood	-1925.233	F-statistic	291.5715	
Durbin-Watson stat	2.851864	Prob(F-statistic)	0.000000	

Table 5.18: Error Values and other parameters of the least square regression model for incentive paid ($)

Parameters	Numerical Values
Root Mean Square Error	488.1766
Mean Absolute Error	301.7012
Mean Abs. Percent Error	1.272915
Theil Inequality Coefficient	0.010071
Bias Proportion	0.000000
Variance Proportion	0.024867
Covariance Proportion	0.975133

CHAPTER 6

CONCLUSIONS

In this thesis soft computing schemes have been successfully applied to determining optimal demand management contract values. The two soft computing schemes used in this work are Artificial Neural Networks (ANN) and Support Vector Machines (SVM). The variants of the models used are supervised learners meaning that there is the need for a teacher or a means to bench mark the results obtained and Game theory's mechanism design serves as the target for the two proposed schemes. The developed system overcomes two major disadvantages inherent in present demand management contract schemes, namely computational complexity and real time adaptability. The ANN and SVM models each comprise of two phases; The data pre processing phase and the learning phase In the first phase, we apply input attribute normalization (ANN) or scaling (SVM) which basically means that all the input attributes are normalized to an interval of 0 and 1. Care has to be taken when doing this and in this thesis a novel normalization pattern is used. Another important data pre processing measure is efficient and accurate output representation. In this thesis for each soft computing model we selected a reasonable interval range and created a binarised coding system corresponding to the intervals (ANN) and output class representation also corresponding to the interval(SVM). and output binary coding. In the second phase a 3-layer supervised learner based on the back propagation learning algorithm was used for the ANN model while the SVM model utilized a C-SVM with RBF kernel. Different experimental configurations consisting of single and double models for the ANN and SVM schemes were tried and all of the trained soft computing models gave highly encouraging results with very fast processing times. The average of the accuracy of all the models was above 85% and it obtained these results in very fast times. We do not posit that ANN and SVM models be used solely for contract design but experimental results suggests that they will be a massive aid in demand management programs and power system operations when used in conjunction with other models. The major advantage of soft computing models as designed in this thesis is in its real time applicability and due to the fact that they involve less computational procedures as

compared to other methods like system dynamics and game theory. A trained soft computing model that has efficiently generalized a particular power system would be able to give spontaneous optimal demand management contract values as system parameters change. This would ensure that contracts are beneficial to both utility and the customers at all times.

CHAPTER 7

FUTURE WORK

7.1 Overview

In this chapter the probable direction of future research on this topic is presented. Both long term and short term research aims are discussed and preliminary research findings are presented with encountered or expected research difficulties highlighted.

7.2 Probable Future Research

At the time of preparing this manuscript, four refereed publications (3 conferences and 1 journal) have already been published. Four more publications are in the works and are at their completion stages. However there is still a lot of room for more research. As the thesis is multi disciplinary (involves power systems engineering, soft computing and economics) there is still need for in-depth multidisciplinary research. Future research can focus on how to incorporate a penalty factor in the contract formulations, so that customers who fail to cut down load consumption as at when due (a specified time duration) are penalized. Also a hybrid soft computing model that combines Genetic Algorithms, Artificial Neural Networks and Fuzzy Logic is being investigated as this will significantly improve training times and furthermore lead to a reduction in the output intervals to more precise values. Another pending soft computing investigation is how the training to testing ratio can be reduced as this will make for a more robust system.

At the moment, the major research going on is how to obtain some of the present demand management input attributes through soft computing means and emphasis is placed on determining the value of power interruptibility otherwise known as (λ) which is traditionally obtained from existing power flow routines. (λ) represents the cost of not delivering power to a particular location and being able to determine this via soft computing schemes would ultimately lead to an increase in the demand management contract input attributes which will also lead to better learning for the developed soft computing models. In the next section

preliminary research findings on determining (λ) using a single soft computing model (SVM) is presented.

7.3 Determining λ Using Support Vector Machines

Electricity nodal prices are also known as Locational Marginal Price (LMP) or Locational Based Marginal Price (LBMP). They are simply defined in (Alvarado, 2001) as the cheapest way one can deliver one MW of electricity to an electric power system node from the available system generators while respecting all the system limits and constraints in effect. Thus nodal prices are vital information for electricity market participants to build bidding strategies. Furthermore its also a useful tool for the system security coordinator or Independent System Operator (ISO) to perform market re-dispatch in the event congestion occurs in the system. Various formulations and algorithms have been designed to calculate electricity nodal price. These methods can be simply grouped as either conventional optimization methods or methods using soft computing principles. The conventional method of calculating electricity nodal price is calculating nodal price as a by-product of the Optimal Power Flow (OPF). OPF is simply an optimization problem where the goal is to optimize a pre-specified objective function through controlling various power system variables (generator real power, transformer tap settings, generator bus voltage, etc) while simultaneously satisfying various power system constraints (system operating limits and power flow equations).The OPF method is usually a single slack power flow formulation. Other factors affecting nodal price are transmission network configuration, available dispatch units, economic dispatch, transmission constraints, energy demand, etc (Ott, 2001). In this thesis, LMP or LBMP is also termed the value of power interruptibility or simply (λ) and it represents the cost of not delivering power to a particular location. In the presented manuscript LMP's are determined from existing power flow routines and it is the aim that they are determined via a soft computing scheme and we would have a complete soft computing based model. To this end we attempt this on the IEEE 57 test power system shown in Figure 7.1 as we steadily increase the complexity of the power system studied [1]

.

[1]This work was published in the 6th International Symposium on Electrical and Electronics Engineering and Computer Systems (EEECS), November 25-26, 2010. Lefke, North Cyprus. as "Determining Electricity Nodal Prices Using Support Vector Machines".

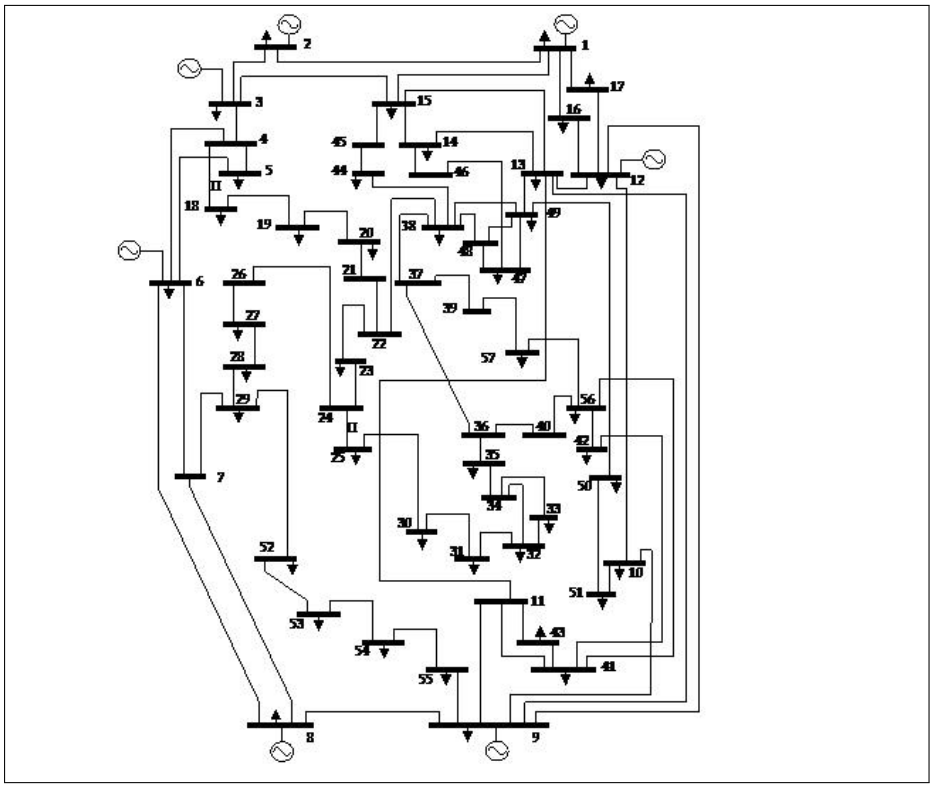

Figure 7.1: The IEEE 57 Bus Test System

The following are our tentative SVM input attributes:

- Bus Number: Since there are 57 buses in this work, the values range from 1 to 57.
- Bus voltage magnitude: This is the voltage magnitude in per unit at each load node. In this work, values range from 0.951 to 1.06 per unit.
- Bus voltage phase angle: This is the phase angle at each load node. In this work, values range from -12.158 to 4.723 degrees.
- Active Power Generated: The power generated at each node. There are 7 generators in this system. In this work, values range from 0 to 459.83 MW.
- Reactive Power Generated: The reactive power generated at each node. In this work, values range from 0 to 87.19 MVAr.
- Load Active Power: The real power at each load node. There are 42 loads in this system. In this work, values range from 0 to 377 MW.
- Load Reactive Power: The reactive power at each load node. In this work values range from 0 to 88 MVAr

7.4 Experimental Analysis and Obtained Results

In this study the parameter search range for C and γ was conducted from 2^{-100} to 2^{100} The best C obtained was 4 while was 8. These values of C and γ were then used for training the SVM learner

The output of the trained SVM model provides the electricity nodal price output with an interval of $ 0.5/MVA-hr which gives us 17 intervals or classes. The trained SVM model consists of two phases: Firstly, the data processing phase and the SVM learning phase. In the first phase, we obtain input features from the test power system. These features undergo scaling prior to using them as inputs to the SVM system. We also designed a novel method of representing the output by creating suitable intervals of $ 0.5/MVA-hr. The second phase is training the SVM classifier. We used a C-SVM model with an RBF kernel. The optimal values of parameters C and were searched for. Training this model successfully required approximately 0.38 seconds.

The implementation results of the trained system were as follows: using the training dataset yielded 89.6591% accuracy. The testing of the trained SVM model using the testing dataset yielded a correct accuracy rate of 67.8947%. Combining the training and testing results yields an overall correct accuracy rate of 78.7769%. It is imperative to mention that the obtained results are average as high accuracy is needed and this is due to the sparse nature of the training dataset. There are 57 nodes in the test system and since we always strive to maintain an even training to testing ratio, 28 nodes are used for training while 29 nodes are used for testing. Obviously 27 nodes will not deliver robust learning results and there is need to either try the algorithm on a much larger power system or simulate more scenarios for the present power system for enhanced learning.

7.5 Summary

In this chapter a brief presentation was provided about the future research spawning out of this thesis. Preliminary results are presented and the difficulties encountered are briefly described.

REFERENCES

Aalami, H. A., Moghaddam, M. P., and Yousefi, G. R. Demand response modelling considering interruptible / curtailable loads and capacity market programs. *Applied Energy*, 87(1):243–250, January 2010. ◇ pp: 4

Alvarado, F. L. Obtaining efficient prices: Nodal pricing and ptdfs. EEI Transmission Pricing School, July 2001. ◇ pp: 54

Atwa, Y. M., El-Saadany, E. F., and Salama, M. M. Dsm approach for water heater utilizing elman neural network. In *Proceedings of the IEEE Electrical Power Conference*, Canada, October 2007. ◇ pp: 6

Bhattacharya, A. and Chakraborty, C. A shunt active power filter with enhanced performance using ann based predictive and adaptive controllers. *IEEE Transactions on Industrial Electronics*, 58(2):421–428, 2011. ◇ pp: 14

Canizares, C. A. and Alvarado, F. L. Point of collapse and continuation methods for large ac/dc systems. *IEEE Transactions on Power Systems*, 8(1):1–8, February 1993. ◇ pp: 28

Chang, C. C. and Lin, C. J. Libsvm: a library for support vector machines. Software available at: /http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001. ◇ pp: 19

Chen, J. C., Hwang, J. C., Pan, J. S., and Huang, Y. C. Pso algorithm applications in optimal demand decision. In *Proceedings of the 6th International Power Electronics and Motion Control Conference*, Wuhan - China, May 2009. ◇ pp: 6, 7

Chogumaira, E. N., Hiyama, T., and Elbaset, A. A. Short-term load forecasting using dynamic neural networks. In *Proceedings of the Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, Chengdu - China, March 2010. ◇ pp: 14

Chu, W. C., Chen, Y. P., Xu, Z. W., and Lee, W. J. Multiregion short term load forecasting in consideration of hi and load/weather diversity. *IEEE Transactions on Industry Applications*, 47(1):232–237, 2011. ◇ pp: 14

Dommel, H. W. and Tinney, W. F. Optimal power flow solutions. *IEEE Transactions on Power Apparatus and Systems*, PAS-87(10):1866–1876, October 1968. ◇ pp: 25

Fahrioglu, M. and Alvarado, F. L. Designing incentive compatible contracts for effective demand management. *IEEE Transactions on Power Systems*, 15(4):1255–1260, November 2000. ◇ pp: 6, 21

Fahrioglu, M., Lasseter, R. H., Alvarado, F. L., and Yong, T. Integrating distributed generation technology into demand management schemes. In *Proceedings of the IEEE PES/IAS Conference on Sustainable Alternative Energy (SAE)*, Valencia - Spain, September 2009. ◇ pp: 6

Forsyth, D. A. and Ponce, J. *Computer Vision: A Modern Approach*. Prentice Hall, 2003. ISBN 978-0-130851987. ◇ pp: 15, 16

Fudenberg, D. and Tirole, J. *Game Theory*. The MIT Press, 1991. ◇ pp: 15, 26

Gellings, C. W. The concept of demand-side management for electric utilities. *Proceedings of the IEEE*, 73(10):1468–1470, 1985. ◇ pp: 1, 4

Greene, S., Dobson, I., and Alvarado, F. L. Sensitivity of the loading margin to voltage collapse with respect to arbitrary parameters. *IEEE Transactions on Power Systems*, 12(1):262–272, February 1997. ◇ pp: 28

Huneault, M. and Galiana, F. D. A survey of the optimal power flow literature. *IEEE Transactions on Power Systems*, 6(2):762–770, May 1991. ◇ pp: 25

Imbert, P., Kariniotakis, G., Blanc, P., and Nierac, F. Resolution enhancement of input parameters in a demand side management model. In *Proceedings of the 11 International Conference on Probabilistic Methods Applied to Power Systems*, Singapore, June 2010. ◇ pp: 5, 7

Khashman, A. A neural network model for credit risk evaluation. *International Journal of Neural Systems*, 19(4):285–294, August 2009. ◇ pp: 13, 20

Khashman, A. Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, 37(9):6233–6239, September 2010. ◇ pp: 13, 20

Kinhal, V., Agarwal, P., and Gupta, H. O. Performance investigation of neural network based unified power quality conditioner. *IEEE Transactions on Power Delivery*, 26(1):431–437, 2011. ◇ pp: 14

Lee, W. L. and Yik, F. W. H. Framework for formulating a performance based incentive rebate scale for the demand side energy management scheme for commercial buildings in hong kong. *Applied Energy*, 73(2):139–166, October 2002. ◇ pp: 4

Li, X., Ruan, D., and van der Wal, A. J. Discussion on soft computing at flins'96. *International Journal of Intelligent Systems*, 13(2-3):287–300, 1998. ◇ pp: 8

Liao, W., Wang, H., and Han, P. Application of neural network combined with improved algorithm in distorted waveform analysis. In *Proceedings of the Chinese Control and Decision Conference (CDCC)*, Xuzhou - China, May 2010. ◇ pp: 14

Moura, P. S. and de Almeida, A. T. The role of demand side management in the grid integration of wind power. *Applied Energy*, 87(8):2581–2588, August 2010. ◇ pp: 5

Ott, A. L. Pjm locational marginal pricing. EEI Transmission Pricing School, July 2001. ◇ pp: 54

Paulus, M. and Borggrefe, F. The potential of demand side management in energy intensive industries for electricity markets in germany. *Applied Energy*, 88(2):432–441, February 2011. ◇ pp: 5

Perera, N. and Rajapakse, A. D. Recognition of fault transients using a probabilistic neural network classifier. *IEEE Transactions on Power Delivery*, 26(1):410–419, 2011. ◇ pp: 14

Ravi, P. B., Divya, V. P. S., Venkatesh, K., Kodad, S. F., and Sankar, B. V. R. Application of ann and dsm techniques for peak load management : A case study. In *Proceedings of the IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronic Engineering*, Novosibirsk - Russia, July 2008. ◇ pp: 6, 7

Saffre, F. and Gedge, R. Demand side management for the smart grid. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, Osaka - Japan, April 2010. ◇ pp: 5, 14

Vapnik, V. N. *The nature of statistical learning theory : Statistics for engineering and information science*. Springer 2nd Edition, New York, USA, 2000. ◇ pp: 15

Wang, R. Load curtailing strategies considering impacts of interruptible load on spot prices. In *Proceedings of the Asia Pacific Power and Energy Engineering Conference APPEEC*, Chengdu - China, March 2010. ◇ pp: 6, 7

Weili, H. and Wei, D. Wavelet neural network applied to power disturbance signal in distributed power system. In *Proceedings of the Chinese Control and Decision Conference (CDCC)*, Guilin - China, June 2009. ◇ pp: 14

Wikipedia. Support vector machine. Wikipedia, 2010. `http://en.wikipedia.org/wiki/Support_vector_machine`. Accessed on November 24, 2010. ◇ pp: ix, 17, 19

Yang, H., Zhang, Y., and Tong, X. System dynamics model for demand side management. In *Proceedings of the 3rd International Conference on Electrical and Electronic Engineering*, Veracruz - Mexico, September 2006. ◇ pp: 6, 7

Zahlay, F. D., Rao, K. S. R., and Ibrahim, T. B. A new intelligent autoreclosing scheme using artificial neural network and taguchi's methodology. *IEEE Transactions on Industry Applications*, 47(1):306–313, 2011. ◇ pp: 14

Zhang, X., Wang, X., and Wang, X. Exotic options bundled with interruptible electricity contracts. In *Proceedings of the 7th International Power Engineering Conference*, Singapore, November/December 2005. ◇ pp: 6, 7

Zimmerman, R. D., Murillo-Sanchez, C. E., and Thomas, R. J. Matpower's extensible optimal power flow architecture. In *IEEE Power and Energy Society General Meeting*, pages 1–7, July 2009. ◇ pp: 31

APPENDICES

APPENDIX A

Sample LIBSVM Code

This appendix lists a section of the LIBSVM code that is used for training the SVM learner. The LIBSVM package has to be installed in the MATLAB path for successful program execution. The parameters in the sample code are those used to determine demand management contract values (specifically load curtailed).

```
cont = csvread('contr.csv'); % read a csv file

labels = cont(:, 1); % labels from the 1st column

features = cont(:, 2:end);
features_sparse = sparse(features); % features must be in a sparse matrix

libsvmwrite('contr.txt', labels, features_sparse);
[cont_label, cont_inst] = libsvmread('contr.txt');

bestcv = 0;
for log2c = -100:100,
  for log2g = -100:100,
    cmd = ['-v 5 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
    cv = svmtrain(cont_label, cont_inst, cmd);
    if (cv >= bestcv),
      bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
    end
    fprintf('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
  end
end

train_data = cont_inst(1:253,:);
train_label = cont_label(1:253,:);
test_data = cont_inst(254:506,:);
test_label = cont_label(254:506,:);
model_rbf = svmtrain(train_label, train_data, '-c 512 -t 2 -g 32');

[predict_label_L, accuracy_L, dec_values_L] = svmpredict(train_label, train_data, model_rbf);

[predict_label_L, accuracy_L, dec_values_L] = svmpredict(test_label, test_data, model_rbf);
```

■