Neuro-Fuzzy Paradigms for Intelligent Energy Management

Ajith Abraham and Muhammad Riaz Khan*

Department of Computer Science, Oklahoma State University, 700 N Greenwood Avenue, Tulsa, OK 74106-0700, USA, Email: ajith.abraham@ieee.org

*AMEC Technologies, TTI, 400-111 Dunsmuir Street, Vancouver, BC V6B 5W3, Canada, E-mail: riaz.khan@amec.com

Abstract: Intelligent energy management has become one of the major research fields in electrical engineering. It constitutes an important tool for efficient planning and operation of power systems and its significance has been intensifying particularly, because of the recent movement towards open energy markets and the need to assure high standards on reliability. Hybrid neuro-fuzzy paradigms have recently gained a lot of interest in research and application. In this chapter, we discuss two neuro-fuzzy paradigms for intelligent energy management. In the first approach, a neural network learning algorithm is used to fine tune the parameters of a Mamdani and Takagi Sugeno Fuzzy Inference System (FIS). Mamdani FIS is used to predict the energy demand and the Takagi-Sugeno FIS is used to predict the reactive power flow. In the second approach, fuzzy *if-then* rules were embedded into an Artificial Neural Network (ANN) learning algorithm (fuzzy-neural network) to achieve improved performance for short-term load forecast. The performance of the different neuro-fuzzy paradigms were tested using real world data and compared with a direct neural network and FIS approach. The different performance results obtained clearly demonstrates the importance of the proposed techniques for intelligent energy management.

Keywords: neuro-fuzzy, computational intelligence, hybrid systems, neural network, fuzzy system

1. Introduction

Accurate load forecasting is of great importance for power system operation. It is the basis of economic dispatch, hydrothermal coordination, unit commitment, and system security analysis among other functions [23]. Short-term load forecasts have become increasingly important since the rise of the competitive energy markets [24][25][27][30]. Many countries have recently privatized and

deregulated their power systems, and electricity has been turned into a commodity to be sold and bought at market prices. Since the load forecasts play a crucial role in the composition of these prices, they have become vital for the supply industry.

Load forecasting is however a difficult task. First, because the load series is complex and exhibits several levels of seasonality: the load at a given hour is dependent not only on the load at the previous hour, but also on the load at the same hour on the previous day, and on the load at the same hour on the day with the same denomination in the previous week. Secondly, because there are many important exogenous variables that must be considered, specially weather-related variables.

We consider two different ways of integrating neuro-fuzzy paradigms [4]. The first approach is to apply a learning algorithm to a FIS [9], which is represented in a special ANN like architecture [26]. However the conventional ANN learning algorithms (gradient descent) cannot be applied directly to such a system as the functions used in the inference process are usually non differentiable. This problem can be tackled by using differentiable functions in the inference system or by not using the standard neural learning algorithm. The performance of the algorithms are validated by practical energy data [8][3][6].

In the second approach, the input parameters consisting of load patterns and weather parameters are fuzzified and used to train a neural network. We applied the backpropagation algorithm to train neural network to find a preliminary forecast load. In addition, the rule base of the fuzzy inference machine contains linguistic importance attached to them in terms of membership functions with knowledge in the form of fuzzy "*if-then*" rules. It makes the load correction inference from historical information and past forecast load errors to calculate the forecast load error. Adding the current forecast load error to the preliminary forecast load, we obtained the final forecast load. The effectiveness of the proposed approach to the short-term load-forecasting problem is demonstrated by the practical data collected from the Czech Electric Power Company (CEZ), Czech Republic [19][20][22][21].

This paper is organized as follows. In sections 2 and 3, we present the different neuro-fuzzy paradigms followed by the different experimentation results in Section 4. Some conclusions are also provided towards the end.

2. Integrating Neural Networks and Fuzzy Inference System

A conventional fuzzy controller makes use of a model of the expert who is in a position to specify the most important properties of the process. Expert knowledge is often the main source to design the fuzzy inference systems. Figure 1 shows the architecture of the fuzzy inference system controlling a process. According to the

performance measure of the problem environment, the MFs, rule bases and the inference mechanism are to be adapted [7].

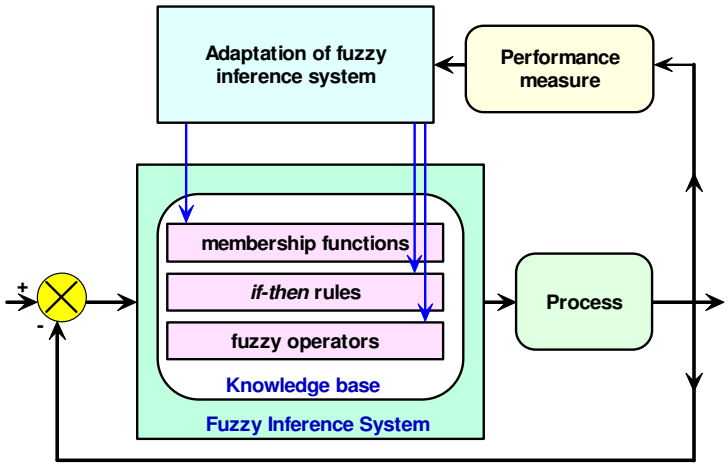

Figure 1. Architecture of adaptive fuzzy inference systems

Several research works are going on exploring the adaptation of fuzzy inference systems [13][11][15][36][29][2][1][26]. These include the adaptation of membership functions, rule bases, aggregation operator's etc. These techniques include but are not limited to:

- Self-organizing process controller by Procyk et al [34], which considered the issue of rule generation and adaptation.
- Evolutionary algorithms to optimize the fuzzy parameters, rule base etc.[5][10][31].
- Gadient descent and its variants have been applied to fine-tune the parameters of the input and output membership functions [38].
- Pruning the quantity and adapting the shape of input/output membership functions [39].
- Fuzzy discretization and clustering techniques [40]

In most cases the inference of the fuzzy rules is done using the '*min*' and '*max*' operators for fuzzy intersection and union. If the T-norm and T-conorm operators are parameterized then gradient descent technique could be used in a supervised learning environment to fine-tune the fuzzy operators.

In an integrated model, neural network learning algorithms are used to determine the parameters of fuzzy inference systems. Integrated neuro-fuzzy systems share data structures and knowledge representations. A fuzzy inference system can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a fuzzy inference system. There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate. On the other hand, neural network learning mechanism does not rely on human expertise. Due to the homogenous structure of neural network, it is hard to extract structured knowledge from either the weights or the configuration of the network. The weights of the neural network represent the coefficients of the hyper-plane that partition the input space into two regions with different output values. If we can visualize this hyper-plane structure from the training data then the subsequent learning procedures in a neural network can be reduced. However, in reality, the a priori knowledge is usually obtained from human experts and it is most appropriate to express the knowledge as a set of fuzzy if-then rules and it is very difficult to encode into an neural network. Modeling integrated neuro-fuzzy systems implementing Mamdani and Takagi Sugeno FIS is presented in Sections 2.1.1 and 2.1.2.

2.1 Adaptive Network Based Fuzzy Inference System (ANFIS)

ANFIS [15] is perhaps the first integrated hybrid neuro-fuzzy model. ANFIS structure as shown in Figure 2 is capable of implementing the Takagi and Sugeno FIS [14]. The detailed functioning of each layer is as follows:

Layer-1 (*fuzzification layer*): Every node in this layer has a node function

$$O_i^1 = \mu_{A_i}(x), \text{ for i =1, 2} \qquad (1)$$

O_i^1 is the membership grade of a fuzzy set A (= A_1, A_2, or B_1 or B_2) and it specifies the degree to which the given input x (or y) satisfies the quantifier A. Usually the node function can be any parameterized function.. A gaussian membership function is specified by two parameters c (membership function center) and σ (membership function width) .

$$\text{guassian } (x, c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \qquad (2)$$

Parameters in this layer are referred to premise parameters.

Layer-2 (*rule firing strength layer*): Every node in this layer multiplies the incoming signals and sends the product out. Each node output represents the firing strength of a rule.

$$O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1,2..... \quad (3)$$

In general any T-norm operators that perform fuzzy AND can be used as the node function in this layer.

Layer-3 Every *i*-th node in this layer calculates the ratio of the *i*-th rule's firing strength to the sum of all rules firing strength.

$$O_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2.... \quad (4)$$

Layer 4 *(rule strength normalization):* Every node in this layer calculates the ratio of the *i*-th rule's firing strength to the sum of all rules firing strength

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2.... \quad (5)$$

Layer-5 *(rule consequent layer):* Every node *i* in this layer is with a node function

$$\overline{w_i} f_i = \overline{w_i} (p_i x_1 + q_i x_2 + r_i) \quad (6)$$

where $\overline{w_i}$ is the output of layer 4, and $\{p_i, q_i, r_i\}$is the parameter set. A well-established way is to determine the consequent parameters using the least means squares algorithm.

Layer-6 *(rule inference layer):* The single node in this layer computes the overall output as the summation of all incoming signals:

$$Overall\ output = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (7)$$

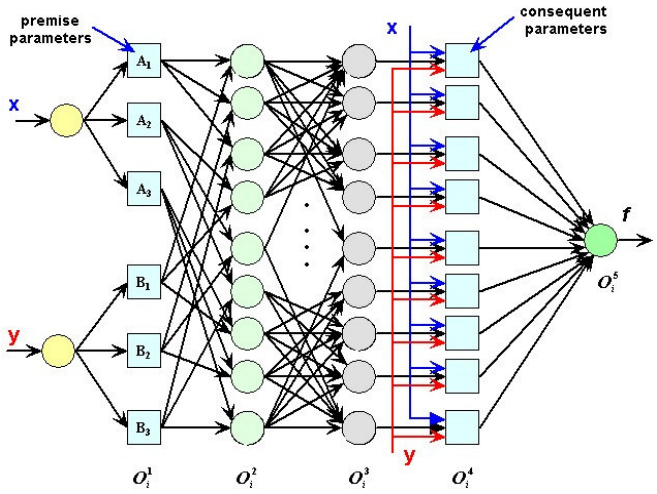

Figure 2. Architecture of the ANFIS

Takagi Sugeno neuro-fuzzy systems make use of a mixture of back propagation to learn the membership functions and least mean square estimation to determine the coefficients of the linear combinations in the rule's conclusions [14]. A step in the learning procedure got two parts: In the first part the input patterns are propagated, and the optimal conclusion parameters are estimated by an iterative least mean square procedure, while the antecedent parameters (membership functions) are assumed to be fixed for the current cycle through the training set. In the second part the patterns are propagated again, and in this epoch, back propagation is used to modify the antecedent parameters, while the conclusion parameters remain fixed. This procedure is then iterated. Assuming a single output ANFIS represented by

$$output = F(\vec{I}, S) \qquad (8)$$

where I is the set of input variables and S is the set of parameters, if there exist a function H such that the composite function $H \circ F$ is linear in some of the elements of S, then these elements can be identified by the least squares method. More formally the parameter set S can be decomposed into two sets:

$$S = S_1 \oplus S_2 \text{ (where } \oplus \text{ represents direct sum)}, \qquad (9)$$

such that $H \circ F$ is linear in the elements of S_2. Then upon applying H to (8), we have:

$$H(output) = H \circ F(\vec{I}, S) \qquad (10)$$

which is linear in the elements of S_2. Now the given values of elements of S_1, we can plug *P* training data into (10), and obtain a matrix equation:

$AX = B$ (*X* = unknown vector whose elements are parameters in S_2)*(11)*

If $|S_2|$=*M*, (*M*= number of linear parameters) then the dimensions of *A, X* and *B* are $P \times M$, $M \times 1$ and $P \times 1$ respectively. Since *P* is always greater than *M*, there is no exact solution to *(11)*. Instead a Least Square Estimate (LSE) of *X,* X^*, is sought to minimize the squared error $\|AX - B\|^2$. X^* is computed using the pseudo-inverse of *X*:

$$X^* = (A^T A)^{-1} A^T B \qquad (12)$$

where A^T is the transpose of A and $(A^T A)^{-1} A^T$ is the pseudo-inverse of *A* where $A^T A$ is non-singular. Due to computational complexity, in ANFIS a sequential method is deployed as follows:

Let the *i-th* row vector of matrix A defined in (11) be a_i^T and *i-th* element of matrix B defined be b_i^T, then X can be calculated iteratively using the following sequential formulae:

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \quad i = 0,1,\ldots\ldots,P-1 \end{aligned} \qquad (13)$$

where S_i is often called the covariance matrix and the least squares estimate X^* is equal to X_P. The initial condition to bootstrap (13) are X_O=*0* and S_O=γ *I,* where γ is a positive large number and *I* is the identity matrix of dimension *M x M.* For a multi-output ANFIS, (13) is still applicable except the $output = F(\vec{I}, S)$ will become a column vector. Each epoch of this hybrid learning procedure is composed of a forward pass and a backward pass. In the forward pass, we have to supply the input data and functional signals go forward to calculate each node output until the matrices *A* and *B* in (11) are obtained, and the parameters in S_2 are identified by the sequential least squares formulae given in (13). After identifying parameters in S_2, the functional signals keep going forward till the error measure is calculated. In the backward pass, the error rates propagate from the output layer to the input layers, and the parameters in S_1 are updated by the gradient method given by

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha} \qquad (14)$$

where α is the generic parameter, η is a learning rate and E the error measure. For given fixed values of parameters in S_1, the parameters in S_2 thus found are guaranteed to be the global optimum point in the S_2 parameter space due to the choice of the squared error measure[14].

The procedure mentioned above is mainly for offline learning version. However the procedure can be modified for an online version by formulating the squared error measure as a weighted version that gives higher weighting factors to more recent data pairs. This amounts to the addition of a forgetting factor λ to (13).

$$X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^T - a_{i+1}^T X_i)$$
$$S_{i+1} = \frac{1}{\lambda}\left[S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}}\right] \quad i = 0,1,\ldots\ldots,P-1 \qquad (15)$$

The value of λ is between 0 and 1. The smaller the λ is, faster the effects of old data decay. But a smaller λ sometimes causes numerical instability and should be avoided.

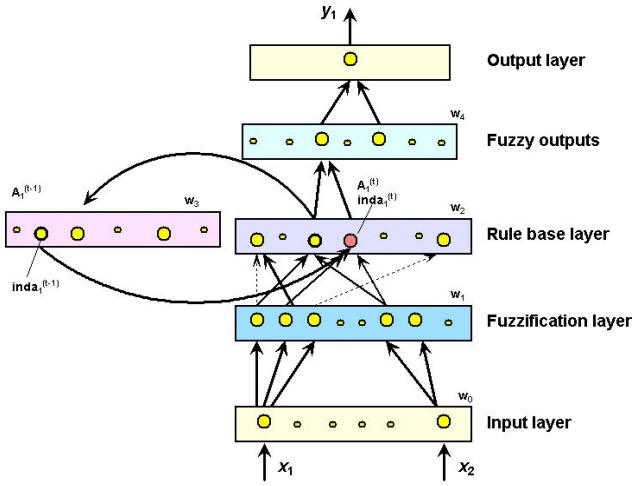

Figure 3. Architecture of EFuNN

2.2 Evolving Fuzzy Neural Networks

Evolving Fuzzy Neural Network (EFuNN) (Figure 3) implements a Mamdani type FIS and all nodes are created during learning [16]. The nodes representing

membership functions (MFs) can be modified during learning. Each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantization of this variable. For example, three neurons can be used to represent "small", "medium" and "large" fuzzy values of the variable. Different membership functions can be attached to these neurons (triangular, Gaussian, etc.). New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold.

The third layer contains rule nodes that evolve through hybrid supervised/unsupervised learning. The rule nodes represent prototypes of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node, e.g. r_j, represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space; $W_1(r_j)$ connection weights representing the co-ordinates of the center of the sphere in the fuzzy input space, and $W_2(r_j)$ – the co-ordinates in the fuzzy output space. The radius of an input hyper-sphere of a rule node is defined as (1- *Sthr*), where *Sthr* is the sensitivity threshold parameter defining the minimum activation of a rule node (e.g., r_1, previously evolved to represent a data point (X_{d1},Y_{d1})) to an input vector (e.g., (X_{d2},Y_{d2})) in order for the new input vector to be associated with this rule node. Two pairs of fuzzy input-output data vectors $d_1=(X_{d1},Y_{d1})$ and $d_2=(X_{d2},Y_{d2})$ will be allocated to the first rule node r_1 if they fall into the r_1 input sphere and in the r_1 output sphere, i.e. the local normalised fuzzy difference between X_{d1} and X_{d2} is smaller than the radius r and the local normalised fuzzy difference between Y_{d1} and Y_{d2} is smaller than an error threshold *Errthr*. The local normalised fuzzy difference between two fuzzy membership vectors d_{1f} and d_{2f} that represent the membership degrees to which two real values d_1 and d_2 data belong to the pre-defined MF, are calculated as $D(d_{1f},d_{2f}) = sum(abs(d_{1f} - d_{2f}))/sum(d_{1f} + d_{2f})$.

If data example $d_1 = (X_{d1},Y_{d1})$, where X_{d1} and X_{d2} are correspondingly the input and the output fuzzy membership degree vectors, and the data example is associated with a rule node r_1 with a center r_1^1, then a new data point $d_2=(X_{d2},Y_{d2})$, will also be associated with this rule node through the process of associating (learning) new data points to a rule node. The centers of this node hyper-spheres adjust in the fuzzy input space depending on a learning rate lr_1, and in the fuzzy output space depending on a learning rate lr_2, on the two data point's d_1 and d_2. The adjustment of the center r_1^1 to its new position r_1^2 can be represented mathematically by the change in the connection weights of the rule node r_1 from $W_1(r_1^1)$ and $W_2(r_1^1)$ to $W_1(r_1^2)$ and $W_2(r_1^2)$ according to the following vector operations:

$$W_2(r_1^2) = W_2(r_1^1) + lr_2 \,.\, Err(Y_{d1},Y_{d2}) \,.\, A_1(r_1^1) \qquad (16)$$

$$W_1(r_1^2) = W1(r_1^1) + lr_1 . Ds(X_{d1}, X_{d2}) \quad (17)$$

where $Err(Y_{d1},Y_{d2}) = Ds(Y_{d1},Y_{d2}) = Y_{d1}-Y_{d2}$ is the signed value rather than the absolute value of the fuzzy difference vector; $A_1(r_1^1)$ is the activation of the rule node r_1^1 for the input vector X_{d2}.

While the connection weights from W_1 and W_2 capture spatial characteristics of the learned data (centers of hyper-spheres), the temporal layer of connection weights W_3 captures temporal dependencies between consecutive data examples. If the winning rule node at the moment $(t-1)$ (to which the input data vector at the moment $(t-1)$ was associated) was $r_1 = inda_1(t-1)$, and the winning node at the moment t is $r_2 = inda_1(t)$, then a link between the two nodes is established as follows:

$$W_3(r_1,r_2)^{(t)} = W_3(r_1,r_2)^{(t-1)} + lr_3 . A_1(r_1)^{(t-1)} A_1(r_2))^{(t)}, \quad (18)$$

where: $A_1(r)^{(t)}$ denotes the activation of a rule node r at a time moment (t); lr_3 defines the degree to which the EFuNN associates links between rules (clusters, prototypes) that include consecutive data examples (if $lr_3=0$, no temporal associations are learned in an EFuNN structure).

The learned temporal associations can be used to support the activation of rule nodes based on temporal, pattern similarity. Here, temporal dependencies are learned through establishing structural links. The ratio spatial-similarity/temporal-correlation can be balanced for different applications through two parameters S_s and T_c such that the activation of a rule node r for a new data example d_{new} is defined as the following vector operations:

$$A_1(r) = f(S_s . D(r, d_{new}) + T_c . W_3(r^{(t-1)}, r)) \quad (19)$$

where f is the activation function of the rule node r, $D(r, d_{new})$ is the normalised fuzzy distance value and $r^{(t-1)}$ is the winning neuron at the previous time moment. The fourth layer of neurons represents fuzzy quantification for the output variables. The fifth layer represents the real values for the output variables.

EFuNN evolving algorithm is given as a procedure of consecutive steps [16].

1. *Initialize an EFuNN structure with a maximum number of neurons and zero value connections. If initially there are no rule nodes connected to the fuzzy input and fuzzy output neurons, then create the first node* $r_j=1$ *to represent the first data example* $EX=(X_{d1}, Y_{d1})$ *and set its input* $W_1(r_j)$ *and output* $W_2(r_j)$ *connection weights as follows:*

   <Create a new rule node r_j> *to represent a data sample* EX: $W_1(r_j)$=EX: $W_2(r_j)$= TE, *where* TE *is the fuzzy output vector for the (fuzzy) example* EX.

2. *While <there are data examples> Do*

Enter the current, example (X_{di}, Y_{di}), EX *being the fuzzy input vector (the vector of the degrees to which the input values belong to the input membership functions). If there are new variables that appear in this example and have not been used in previous examples, create new input and/or output nodes with their corresponding membership functions.*

3. *Find the normalized fuzzy similarity between the new example* EX *(fuzzy input vector) and the already stored patterns in the case nodes* $r_j = r_1, r_2, \ldots, r_n$

   $D(EX, r_j) = \text{sum}(\text{abs}(EX - W_1(r_j))) / \text{sum}(W_1(r_j) + EX)$

4. *Find the activation* $A_1(r_j)$ *of the rule nodes* $r_j = r_1, r_2, \ldots, r_n$. *Here radial basis activation (*radbas*) function, or a saturated linear (*satlin*) one, can be used, i.e.*

   $A_1(r_j) = \text{radbas}(S_s D(EX, r_j - T_c W_3)$, *or* $A_1(r_j) = \text{satlin}(1 - S_s D(EX, r_j + T_c W_3))$.

5. *Update the pruning parameter values for the rule nodes, e.g. age, average activation as pre-defined.*

6. *Find* m *case nodes* r_j *with an activation value* $A_1(r_j)$ *above a predefined sensitivity threshold* Sthr.

7. *From the* m *case nodes, find one rule node* inda_1 *that has the maximum activation value* maxa_1.

8. *If* $\text{maxa}_1 <$ Sthr, *then,* <create a new rule node> *using the procedure from step 1.*

   Else

9. *Propagate the activation of the chosen set of* m *rule nodes* ($r_{j1}, \ldots, r_{jm}$) *to the fuzzy output neurons:* $A_2 = \text{satlin}(A_1(r_{j1}, \ldots, r_{jm}) \cdot W_2)$

10. *Calculate the fuzzy output error vector*

    $Err = A_2 - TE$

11. *If* $(D(A_2, TE) >$ Errthr) <create a new rule node> *using the procedure from step 1.*

12. *Update (a) the input, and (b) the output of the* m-1 *rule nodes* $k = 2 : j_m$ *in case of a new node was created, or* m *rule nodes* $k = j_1 : j_m$, *in case of no new rule was created:*

    $Ds(EX - W_1(r_k)) = EX - W_1(r_k)$; $W_1(r_k) = W_1(r_k) + lr_1 . Ds(EX - W_1(r_k))$, *where* lr_1 *is the learning rate for the first layer;*

    $A_2(r_k) = \text{satlin}(W_2(r_k).A_1(r_k))$; $Err(rk) = TE - A_2(r_k)$;

    $W_2(r_k) = W_2(r_k) + lr_2 . Err(r_k) . A_1(r_k)$, *where* lr_2 *is the learning rate for the second layer.*

13. *Prune rule nodes* r_j *and their connections that satisfy the following fuzzy pruning rule to a pre-defined level representing the current need of pruning:*

    IF (a rule node r_j *is OLD) and (average activation* $A_1av(r_j)$ *is LOW) and (the density of the neighboring area of neurons is HIGH or MODERATE) (i.e. there are other prototypical nodes that overlap with j in the input-output space; this condition apply only for some strategies of inserting rule nodes as explained below) THEN the probability of pruning node (*r_j*) is HIGH. The above pruning rule is fuzzy and it requires that the fuzzy concepts as OLD, HIGH, etc. are predefined.*

14. *Aggregate rule nodes, if necessary, into a smaller number of nodes. A C-means clustering algorithm can be used for this purpose.*

15. *End of the* while *loop and the algorithm*

The rules that represent the rule nodes need to be aggregated in clusters of rules. The degree of aggregation can vary depending on the level of granularity needed. At any time (phase) of the evolving (learning) process, fuzzy, or exact rules can be inserted and extracted [17]. Insertion of fuzzy rules is achieved through setting a new rule node for each new rule, such as the connection weights W_1 and W_2 of the rule node represent the fuzzy or the exact rule. The process of rule extraction can be performed as aggregation of several rule nodes into larger hyper-spheres. For the aggregation of two-rule nodes r_1 and r_2, the following aggregation rule is used

$$If\ (D(W_1(r_1),W_1(r_2)) <= Thr_1)\ and\ (D(W_2(r_1),W_2(r_2)) <= Thr_2) \qquad (20)$$

then aggregate r_1 and r_2 into r_{agg} and calculate the centers of the new rule node as

$$W_1(r_{agg}) = average\ (W_1(r_1),W_1(r_2)),\ W_2(r_{agg}) = average\ (W_2(r_1),W_2(r_2)) \quad (21)$$

Here the geometrical center between two points in a fuzzy problem space is calculated with the use of an average vector operation over the two fuzzy vectors. This is based on a presumed piece-wise linear function between two points from the defined through the parameters *Sthr* and *Errthr* input and output fuzzy hyper-spheres.

2.3 Hybrid Fuzzy Neural Network (FNN)

In a hybrid model, the neural network is used to learn and classify the patterns, automatically creating the fuzzy rules and the fuzzy logic is used to infer the defuzzified output. The structure of the fuzzy-neural network used is shown in Figure 4 [21].

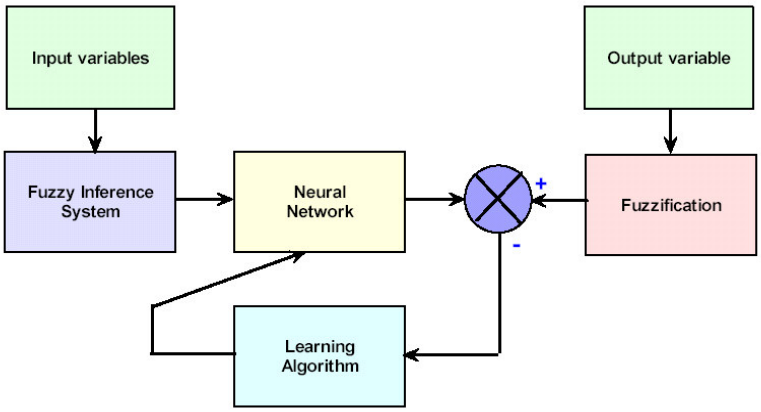

Figure 4. Structure of the fuzzy neural network

The main features of FNN are:

- It provides a general method for combining available numerical information and human linguistic information in a common framework.
- It requires much less construction time than a comparable neural network.
- Significant accuracy is achieved in predicting chaotic time-series models.
- The mean absolute error is reduced in case of FNN as compared to ANN.
- The advantage of the FNN is its efficient adaptive tracking capability that results in the development of a robust and accurate forecasting technique, which gives an accurate forecast even under diverse weather conditions.

This hybrid approach utilizes the inherent properties of artificial neural networks, such as generalization and adaptability, self-organization, ability to solve nonlinear problems, retrieval from partial information, learning from well-defined patterns; and properties of fuzzy logic, such as abstract reasoning and human-like responses in cases involving uncertainty and contradictory data.

A fuzzy processor has been utilized for preprocessing the inputs with the application of fuzzy rules. The fuzzy processor effectively handles the numeric data and produces a fuzzy output vector, which is then fed to 3-layered feedforward neural network trained using backpropagation algorithm. The inputs to the neural network are processed by the ANN, which generates response at its output layer. This response is compared with the desired output and their difference is propagated backwards through the network connections to reduce this error. The learning data is presented repeatedly until the errors are reduced to an acceptable level. Once trained, the outputs of the neural network, interpreted as

fuzzy membership functions of the desired output, are defuzzified to get the required output. In defuzzification, all significant fuzzy outputs are combined into a specific, comprehensive result for that output variable. In this process, all the fuzzy output values effectively modify their respective output membership functions. The ANN is allowed to train until it maps the input-output relationship with the desired accuracy [19][22][20].

3. Modern Energy Management

Energy generation, distribution and management is changing dramatically. In a few years, the concepts we have all grown up will might be gone forever. The traditional gas and electric companies as we know them today will be in a very different business, competing on the open market and possibly combining services. Deregulated electricity supply markets can mean substantial electricity cost savings for consumers willing to take the time to study the opportunities. In a deregulated electricity supply market, there are theoretically numerous merchant power companies with generation assets standing by to provide consumers with the power they need to operate the homes and businesses at a fair price. Consumers are theoretically able to negotiate with the merchant power companies to agree to a price for power, which will be added to the charges imposed by the incumbent transmission and distribution grid owners to deliver power to the consumer. However, not all customers will be able to get the same good deal as every other customer, economies of scale aside.

In a perfect world, every customer would use a constant amount of power at all times of the day and every day of the year. This would make it easy for the companies saddled with the responsibility of maintaining and operating the electricity generation and distribution systems to keep everything running smoothly, and at an economical price. Unfortunately for everybody, the world doesn not work that way. People use more power at peak hours during the day when they are operating power-hungry machines under bright fluorescent lights in air-conditioned offices, than they use at night when they at home in bed. This means that utility companies must make allowances for mid-day peaks in power consumption as they provide for the generation, transmission and distribution of power to customers in the city.

The prediction of electricity demand has been of much interest to the electricity supply industry for some years, both to aid long term planning strategies, involving the forecasting of seasonal peak demands, and for use in the short term (up to 24 hours) operation of generating plant. The nature of electricity market is changing very rapidly with a widespread international movement towards competitiveness. Traditionally, the energy sector, and particularly the electricity sector, has been dominated by monopoly or near monopoly enterprises, typically

either owned or regulated by government. The recent privatization of the electricity supply industry has brought a renewed interest in this subject.

Some countries, such as Norway, Chile, Japan, UK and the United States have commonly been supplied electricity by a large number of different regional Generators and have developed a variety of mechanisms to allow some form of trade between them. In 1994 Victoria started the process of privatization and restructuring electricity industry to generate competition. The objective was to promote a more flexible, cost-effective and efficient electricity industry with the aim of delivering cheaper electricity to business and the general community. Following success of this operation, Australia started the process of implementing a unified National Electricity Market in December 1998 [8].

3.1 Modeling Electricity Demand Prediction in Victoria (Australia)

To meet the electricity market demands a highly reliable supply and delivery system is required. Additionally, in order to gain a competitive advantage in this market through the competitive spot-market pricing an accurate forecast of electricity demand at regular time intervals is essential. Until 1996, Victorian Power Exchange (VPX) the body responsible for the secure operations of the power system, generated electricity demand forecasts based on weather forecasts and historical demand patterns [28]. Our research is focused on developing more accurate and reliable forecasting models that improves current forecasting methods. Our approach is to develop reliable and accurate prediction models predicting 96 half-hourly (two days ahead) demands for electricity, and compares their performance with forecasts used by VPX. We considered an integrated neuro-fuzzy system and a feedforward artificial neural network trained using the scaled gradient conjugate algorithm and backpropagation algorithm. For developing the forecasting models we used the energy demand data for ten months period from 27th January to 30th November 1995 in the State of Victoria. We also made use of the associated data stating the minimum and maximum temperature of the day, time of day, season and the day of week. The forecasting models were trained using 3 randomly selected samples containing 20% of the data during the period 27th January 1995 to 28th November 1995. To ascertain the forecasting accuracy the developed models were tested to predict the demand for the period (29th –30th) November 1995 [8].

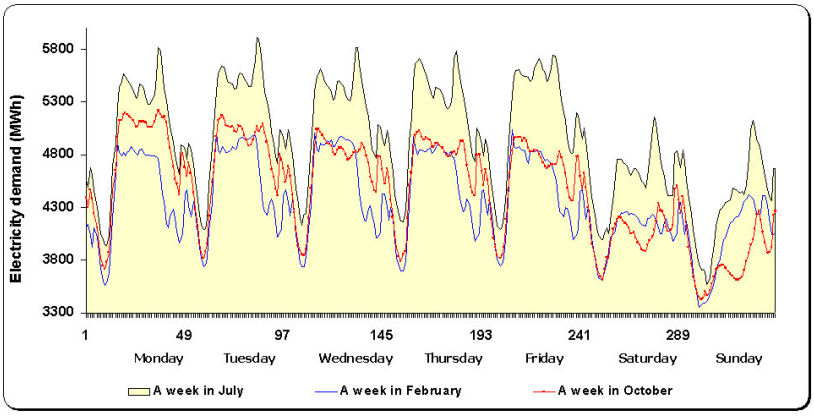

Figure 5. Typical weekly demand variations

The data for our study were the recorded half-hourly actual electricity demand for the ten months period from January to November 1995 in the State of Victoria. Figure 5 shows a typical weekly cycle of electricity demand during three different months of the year. Fluctuations in daily demand are prevalent with peaks occurring around midday. Extreme weather conditions in winter and summer months accentuate peaks in electricity demand due to the widespread use of electricity for heating and cooling. Other times, electricity demand is dominated primarily by ambient temperature, time of day, working or non-working day and the day of the week.

The experimental system consists of two stages: modeling the prediction systems (training in the case of soft computing models) and performance evaluation. For network training, the six selected input descriptor variables were: the *minimum* and *maximum recorded temperatures*, *previous day's demand*, a value expressing the *half-hour period of the day*, *season*, and the *day of the week*. To evaluate the learning capability of the soft computing models, the network was trained only on 20% of the randomly selected data. We created 3 different samples of training data to study the effect of random sampling and periodicity. Each training sample consisted of 2937 data sets representing 20% random data. Our objective is to develop an efficient forecasting model capable of producing a short-term forecast of demand for electricity. The required time-resolution of the forecast is half-hourly, and the required time-span of the forecast is 2 days. This means that the system should be able to produce a forecast of electricity demand for the next 96 time periods. The training was replicated three times using three different samples of training data and different combinations of network parameters.

- **Neuro-Fuzzy Training**

We used 4 Gaussian membership functions for each input variable and the following evolving parameters: sensitivity threshold *Sthr* = 0.99, error threshold

Errthr=0.001 and learning rates for first and second layer = 0.05. EFuNN uses a one pass training approach. The network parameters were determined using a trial and error approach. The training was repeated three times after reinitializing the network and the worst errors were reported. Online learning in EFuNN resulted in creating 2122 rule nodes. Training results and test results are summarized in Table 1.

- **Neural Network training**

Our preliminary experiments helped us to formulate a feedforward neural network with 1 input layer, 2 hidden layers and an output layer [6-40-40-1]. Input layer consists of 6 neurons corresponding to the input variables. The first and second hidden layers consist of 40 neurons respectively using tanh-sigmoidal activation functions. To illustrate the convergence feature of Scaled Conjugate Gradient Algorithm (SCGA), we also trained a neural network (with same architecture) using backpropagation (BP) algorithm. To evaluate the neural network performance, training was terminated after 2500 epochs. Training and testing errors are summarized in Table 1. Figure 6 shows the convergence of SCGA with respect to BP algorithm. Figure 7 depicts the test results for prediction models considered. To have a performance evaluation the actual energy demand and the forecasts used by VHP and Box - Jenkins ARIMA model are also plotted in Figure 7. Compared to neural networks, an important advantage of neuro-fuzzy systems is its reasoning ability (*if-then* rules) of any particular state. A fully trained EFuNN could be replaced by a set of *if-then* rules [17]. A simple example of a learned EFuNN learned rule is illustrated below.

" If the *maximum temperature* of the day is HIGH and *minimum temperature* of the day is LOW and *previous days demand* is MEDIUM and it is *summer* (HIGH) and *9.00 AM* (HIGH) and a *Monday* (HIGH) then the *electricity demand* is MEDIUM."

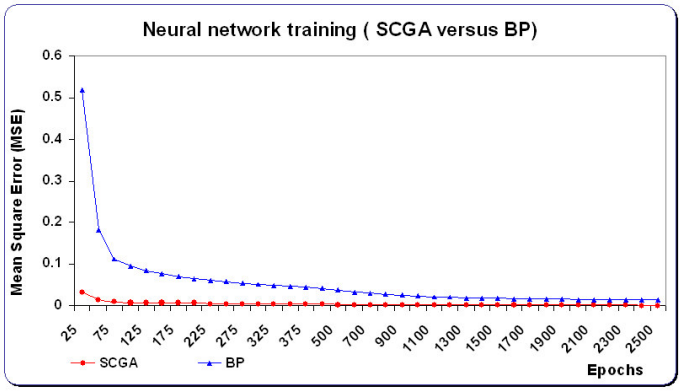

Figure 6. Convergence of neural network training

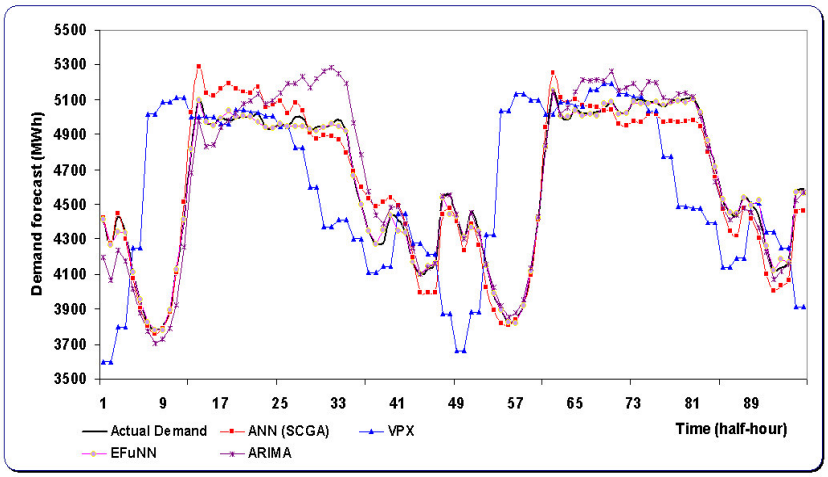

Figure 7. Test results and performance comparison of demand forecasts (2 days)

EFuNN uses a hybrid learning technique (a mixture of unsupervised and supervised learning) to fine-tune the parameters of the fuzzy inference system. As EFuNN adopts a single pass training (1 epoch) it is more adaptable and easy for further online training which might be highly useful for online forecasting and bidding. Another important feature of EFuNN is that the user has the flexibility to construct the network (by selecting the parameters). Hence for applications where speed is more important than the accuracy a faster network can be selected. However an important disadvantage of EFuNN is the determination of the network parameters like number and type of membership functions for each input variable, sensitivity threshold, error threshold and the learning rates. Even though a trial and error approach is practical, when the problem becomes complicated (large number of input variables) determining the optimal parameters will be a tedious task.

Table 1. Test results and performance comparison of demand forecasting [8]

	EFuNN	**ANN (BP)**	**ANN (SCGA)**	**ARIMA**
Learning epochs	1	2500	2500	-
Training error (RMSE)	0.0013	0.116	0.0304	-
Testing error (RMSE)	0.0092	0.118	0.0323	0.0423
Computational load (in billion flops)	0.536	87.2	175.0	-

Our experiments on three separate data samples reveal that the results are not dependent on the data sample. We used only 20% of the total data to evaluate the learning capability of the soft computing models. Network performance could have been further improved by providing more training data. Another interesting fact about the considered soft computing models are their robustness and capability to handle noisy and approximate data that are typical in power systems, and therefore should be more reliable in worst situations.

3.2 Automation of Reactive Power Control

The ratio of active power (P) measured in watts to the apparent power (S) in volt-amperes is termed the power factor:

$$\text{Power factor} = \cos(\varphi) = \frac{P}{S} = \frac{\text{resistance } R}{\text{impedance } Z} \qquad (22)$$

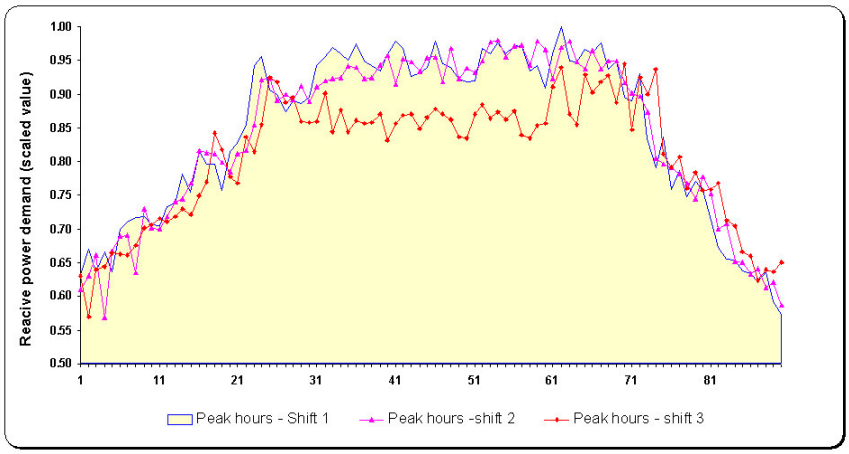

Figure 8. Reactive power demand variations during peak hours

It has become a normal practice to say that the power factor is lagging when the current lags the supply voltage and leading when the current leads the supply voltage. This means that the supply voltage is regarded as the reference quantity. A majority of loads served by a power utility draw current at a lagging power factor. When the power factor of the load is unity, active power equals apparent power ($P = S$). But, when the power factor of the load is less than unity, say 0.6, the power utilized is only 60%. This means that 40% of the apparent power is being utilized to supply the reactive power, VAR, demand of the system. It is therefore clear that the higher the power factor of the load, the greater the utilization of the apparent power. For the generating and transmission stations, lower the power factor the larger must be the size of the source to generate that power, and greater must be the cross-sectional area of the conductor to transmit it.

In other words, the greater is the cost of generation and transmission of the power. Moreover, lower power factor will also increase the I^2R (I denotes current) losses in lines/equipment as well as result in poor voltage regulation.

Most of the utility companies use a complex set of formulas, rewards/penalties etc. to receive an adequate return for their considerable investment in the larger capacity generators, transformers, cables and switchgear required to provide necessary KVA service to their customers. These formulas are generally referred to as power factor adjustments or KVAR reactive demand charges. In recent years, increased attention has been given to plant automation to reduce operational costs. Many manufacturing industries use human operators or timer controlled switching relays to turn on the power capacitors to compensate the reactive power requirement. Operational costs could be reduced and utilization efficiency improved if the power capacitor switching on/off process is automated using some intelligent techniques. We proposed a neuro-fuzzy approach to predict the reactive power trend (at time $t+1$) just by knowing the load current (at time t). Efficient usage of the VA loading will not only improve the overall grid condition but also reduce the consumer's industrial tariffs. Depending on the predicted reactive power demand, power factor corrective measures could be turned on or off to control the VA inflow into the plant. The developed model could be extremely useful for automated control of power inflow, especially in the countries where there are limitations on the usage of consumers' peak VA maximum demand [6].

We considered a heavy automobile manufacturing industry that works on 3 shifts of 8 hours duration for studying the load demand patterns. Observed data for a 24 hour period shows that the maximum and minimum VAR requirements are 2.96 MVAR and 0.014 MVAR, respectively [3][6]. The variation of the reactive power flow during the peak hours of the three 8-hour shifts is depicted in Figure 8. If suitable power factor compensation was made when the reactive power demand was increasing, the plant might not have drawn much apparent power from the grid. The task is to predict the upward and downward trend of the reactive power demand and provide required reactive power compensation. Load flow analysis of the captioned plant reveals that the demand patterns are very similar every day (as long as the production of automobiles remain fixed).

The proposed neuro-fuzzy models and neural network were trained on the data taken at every minute for a 24-hour period to predict the reactive power demand, and tested to evaluate the prediction accuracy. To evaluate the efficiency of the prediction models, three different training and testing data sets were extracted and the experiments were performed three times.

- **Experimentation Setup and Test Results**

24-hour load flow patterns were used to train the neuro-fuzzy models and neural network. The training data comprises of 1440 data sets representing the 24-hour period. The input parameters considered are the phase voltage (V) and current (I). The normal value of input parameter voltage (V) was fluctuated with +/- 2.5% of the normal value. All the data sets were scaled from 0 to 1. The input voltage was

fluctuated to test the learning capability and robustness of the considered connectionist models. Training and testing data sets were extracted randomly from the complete dataset. 60% of data was used for training and remaining 40% for testing. To ensure that the data sample does not have any bias, we created 3 sets of data for training and testing (random extraction). Experiments with all 3 data sets were repeated 3 times for all the connectionist models.

- **Neural network training**

We used a feedforward neural network with 2 hidden layers and trained using the backpropagation algorithm. The 2 input neurons correspond to the input variables and 1 output neuron for predicting reactive power. Initial weights, learning rate and momentum used were 0.3, 0.1 and 0.1, respectively. The training was terminated after 700 epochs.

- **ANFIS training**

In the ANFIS network, we used 3 Gaussian membership functions for each input parameter variable for predicting the reactive power demand. Nine rules were learned based on the training data. The training was terminated after 50 epochs.

- **EFuNN training**

We used 3 Gaussian membership functions and the following evolving parameters: sensitivity threshold *Sthr*=0.95, error threshold *Errthr*=0.05 and 544 rule nodes were created during training.

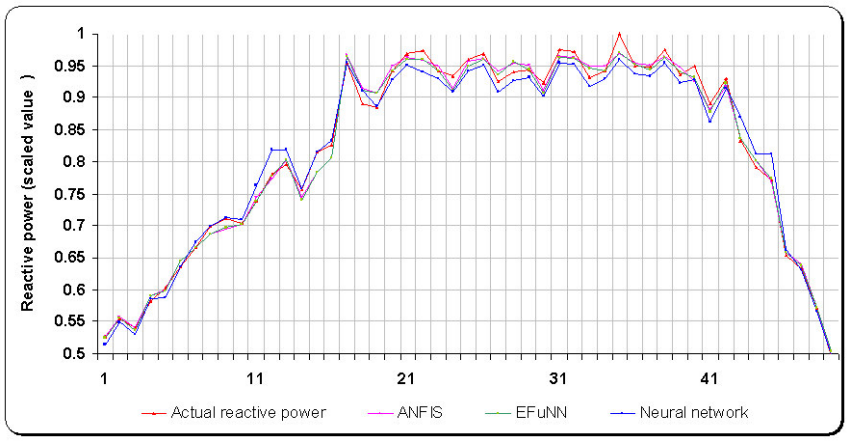

Figure 9. Test results showing the predicted reactive power using different models during the peak hours of shift 1.

Table 2. Reactive power prediction –comparative performance

	ANFIS	EFuNN	ANN
Learning epochs	50	1	700
Training time (seconds)	36	25	188
Training error (RMSE)	0.0103	0.0116	0.0142
Testing error (RMSE)	0.0102	0.0120	0.0130

- **Performance and Results Achieved**

Test data (input parameters) is passed through the trained connectionist models and the predicted output value is compared with the observed reactive power value to calculate the RMSE. Figures 9 illustrates the test results for predicted outputs using ANFIS, EFuNN and ANN. Table 2 shows an empirical comparative performance of the different connectionist models for the reactive power prediction problem. The empirical values shown in Table 2 are the worst values of the three trials with the three data sets for each model.

Among all the connectionist models neuro-fuzzy systems performed better than artificial neural network in terms of performance error achieved and training time. ANFIS performed marginally better than EFuNN in terms of low error. However ANFIS took more training time than EFuNN. Hence there is a compromise between performance error and training time. An important advantage of the EFuNN network is its online learning capability. Hence future training would be much easier. The predicted RMSE values are within acceptable rates and hence the developed models are reliable. The prediction accuracy could have been improved if we had not used the noisy input parameter (voltage) or if the actual voltage values were used.

3.3 Load Forecasting in Czech Republic

The hourly load in the Czech Republic for the year 2000 is shown in Figure 10. The humidity is sorted into seven categories and labeled as extremely low (ExL), very low (VL), low (L), medium (M), high (H), very high (VH) and extremely high (ExH). The wind speed is labeled as zero (Z), positive very small (PVS), positive small (PS), medium (M), positive medium (PM), big (B) and positive big (PB). Similarly, wind chill is labeled as zero (Z), very very low (VVL), very low (VL), low (L), high (H), very high (VH) and extremely high (ExH) as shown in Figure 11. Inputs to the FNN are previous load and weather parameters i.e.

temperature, humidity, wind-speed, and wind-chill and the output of the FNN is the load forecast for a given day [21][22].

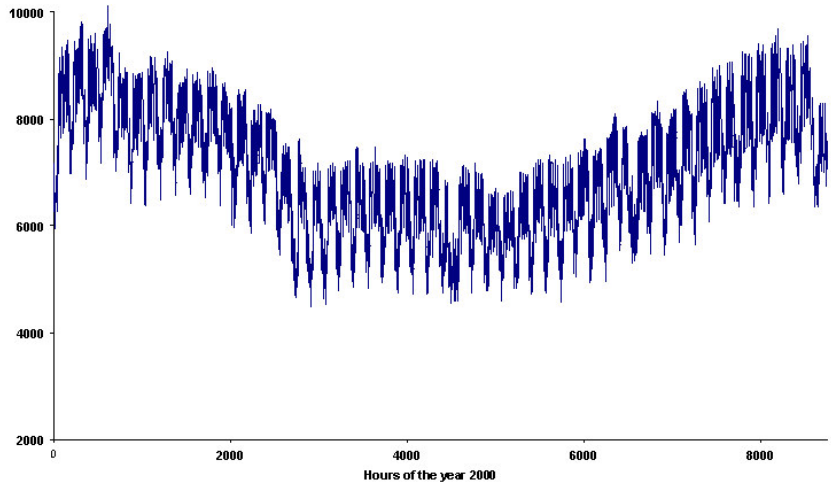

Figure 10. Hourly load of the year 2000

The proposed fuzzy processor develops a linguistic model that describes dynamic behavior of the system by using a nonlinear mapping between input and output. Application of fuzzy inference rules allows amalgamation of different kinds of knowledge, reduces the dimensionality of the input data and effectively deals with nonprecise (fuzzy) information. The potentials of this technique come from its powerful hybrid methodology and extraordinary approximation power. Furthermore, the ability to incorporate human knowledge makes it a more beneficial candidate over others. The results obtained outperform the connectionist approaches.

The fuzzy processor increases the robustness of the forecast model in tackling uncertainties and ambiguity in the inputs. It also avoids the use of large chunk of historical data, and frequent retraining in response to changing input conditions. The fuzzy-neural network learns the training set near perfect and shows precise prediction. Further improvements to the current implementation could be be made using more complete historical load/meteorological data, improving the rules and refining the training strategies to incorporate incremental learning.

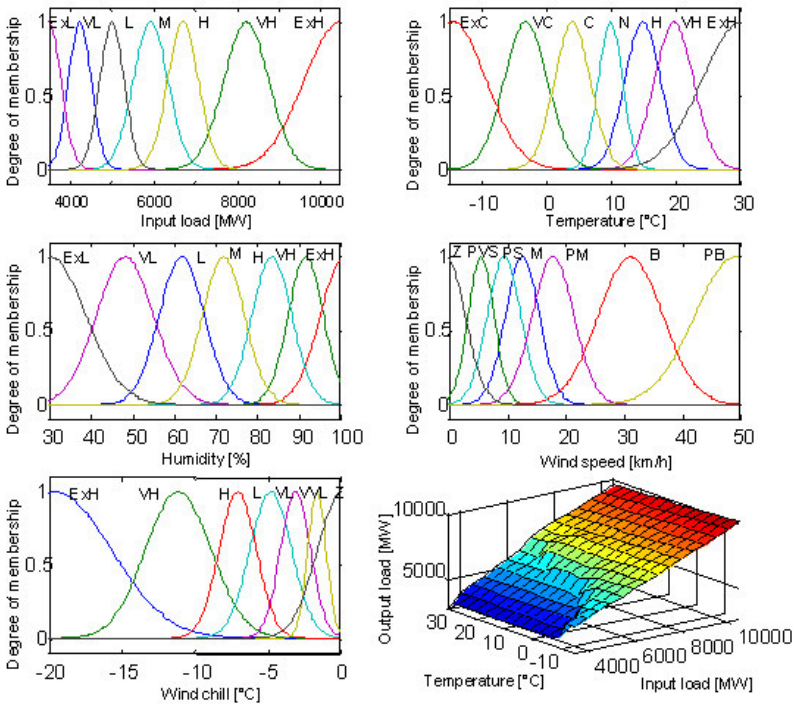

Figure 11. Input parameters using Gaussian-curve membership function.

In our experiments, two years of historical load and weather data were used, one year (1999) for designing the fuzzy rule base design and the following year (2000) for testing the model performance. We used a Mamdani fuzzy inference system for predicting the 24-hour ahead (weekdays and weekends) load demand. To ensure prediction accuracy, the number of fuzzy membership functions and shape of the fuzzy membership functions were changed and new fuzzy rule base was obtained. The iterative process of designing the rule base, choosing a defuzzification algorithm, and testing the system performance was repeated several times with a different shapes number of fuzzy membership functions. The fuzzy rule base that provided the minimum error measure for the test set was selected for real-time forecast. We used various Membership Functions (MF) such as triangular, trapezoidal, Gaussian-curve and bell-shaped. The performance of triangular and Gaussian-curve membership functions were better as compared to bell-shaped and trapezoidal. Using different MF, the mean absolute percentage error (MAPE) and maximum absolute percentage error (MAP) for working days of the week and weekend are depicted in Tables 3 and 4 respectively. Empirical values from Tables 3 and 4 illustrates that the selection of different MFs e.g.,

triangular, Gaussian, trapezoidal etc. significantly affect the prediction performance.

Table 3: MAPE and MAP for working days of a week using various MFs.

Working Days	Membership Functions					
	Triangular		Gaussian Curve		Trapezoidal	
	MAPE (%)	MAP (%)	MAPE (%)	MAP (%)	MAPE (%)	MAP (%)
Monday	2.58	6.46	2.84	7.79	2.73	5.84
Tuesday	1.67	4.89	1.18	4.97	2.59	6.82
Wednesday	0.99	3.23	1.87	4.64	1.91	4.74
Thursday	0.97	4.89	1.53	5.52	2.67	5.74
Friday	1.34	3.89	1.69	4.96	2.94	4.85

Table 4. MAPE and MAP for one weekend using various membership functions.

Weekend Days	Membership Functions					
	Triangular		Gaussian Curve		Trapezoidal	
	MAPE (%)	MAP (%)	MAPE (%)	MAP (%)	MAPE (%)	MAP (%)
Saturday	3.52	8.08	4.82	7.74	5.03	8.66
Sunday	3.76	7.47	4.68	8.71	5.06	9.84

- **Training and Test Data**

The basic training set using 24 values of load data and 6 weather parameters (i.e. minimum, maximum and average temperatures, humidity, wind speed and wind chill) and the day of the week was used for training and testing as shown in Figure 12. To ensure good training, the network was trained using a large data set using data containing values from 1995 to 1999. Besides, the utilization of a large amount of training patterns is an effective antidote to fight against over fitting .

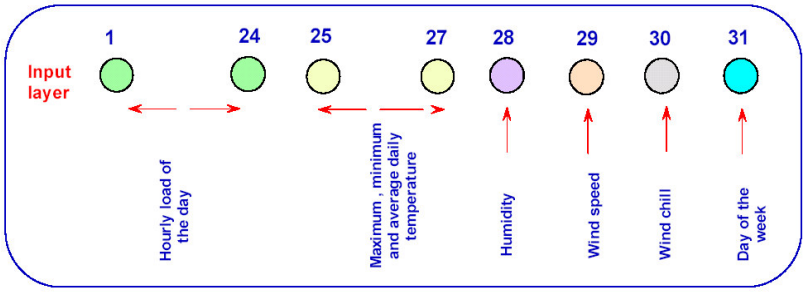

Figure 12. Input data scheme to load forecasting models.

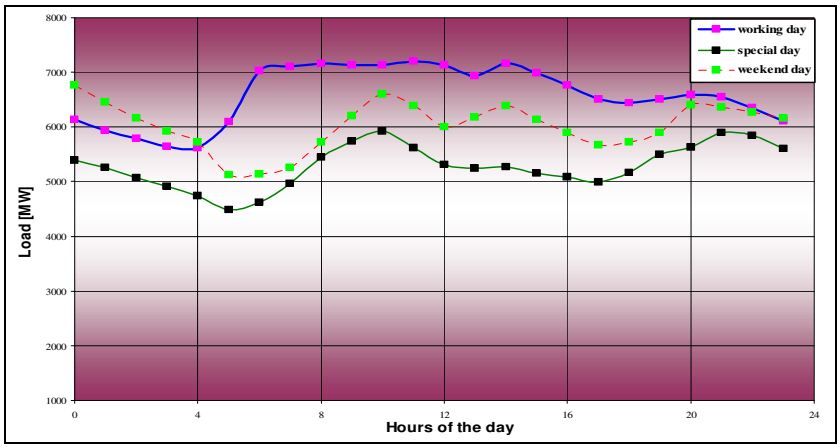

Figure 13. Load trend for a working day, weekend and holiday (special day) of a week in the year 2000.

On the other hand, the whole process is slower and morose, due to the computation time required. The load demand patters are different for working days, weekends and holidays as shown in Figure 13. The data for holidays were eliminated. Thus the training data was distributed into two sets for working days and weekends, respectively. The data for the year 2000 is used for testing and evaluation of generalization performance. For empirical comparison purposes, using the same training data, we also trained an artificial neural network using backpropagation algorithm and also developed a Takagi-Sugeno fuzzy inference system. Test data is passed through the trained networks and the performances are depicted in Tables 5 and 6. Figures 14 and 15 illustrates the forecast load versus the actual load along with the forecast error for one working day (Tuesday) and one weekend day (Sunday), respectively. Our training results also reveals that the FNN converged much faster when compared to a pure neural network approach.

Table 5. MAPE and MAP of 24-hour forecast during weekends.

	Saturday		Sunday	
Models	***MAPE (%)**	***MAP (%)**	**MAPE (%)**	**MAP (%)**
ANN	2.361	5.22	2.614	5.76
FIS	2.882	6.15	2.764	6.42
FNN	1.893	3.81	2.008	4.23

Table 6. MAPE and MAP of 24-hour forecast during weekdays (working days).

	Monday		Tuesday		Wednesday	
Models	**MAPE**	**MAP**	**MAPE**	**MAP**	**MAPE**	**MAP**
ANN	2.416	5.123	2.321	5.446	2.272	6.142
FIS	2.540	4.678	2.524	5.120	1.987	5.378
FNN	2.000	3.210	1.729	3.151	1.672	3.870

	Thursday		Friday	
Models	**MAPE**	**MAP**	**MAPE**	**MAP**
ANN	2.614	6.113	2.583	5.893
FIS	2.786	5.820	2.862	5.416
FNN	1.340	3.672	1.435	3.163

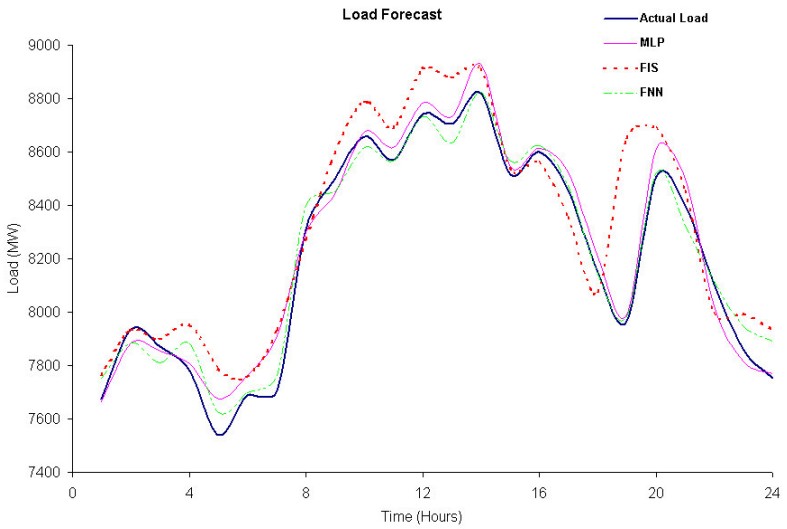

Figure 14. Comparison of 24 hours ahead load forecast for working day (Tuesday) using FNN, FIS and ANN

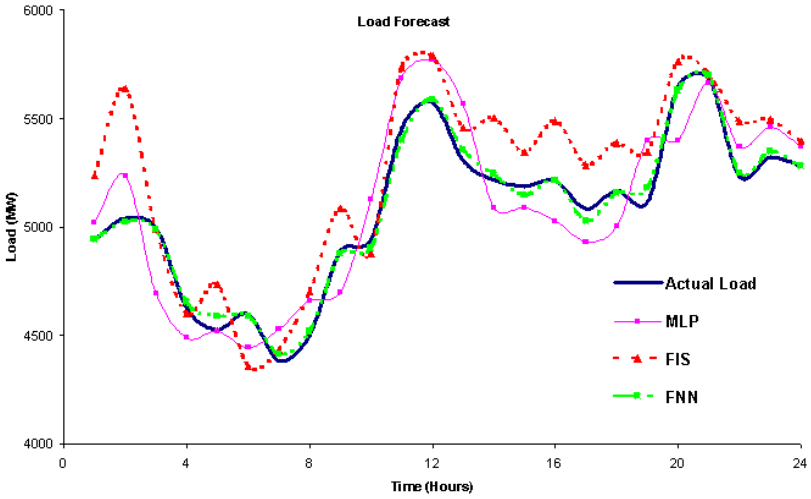

Figure 15. Comparison of 24 hours ahead load forecast for weekend day (Sunday) using FNN, FIS and ANN

4. Conclusions

In this chapter, we attempted to model the integration of neural networks and fuzzy inference systems for energy management problems. For the demand energy demand and reactive power forecast problem, the proposed neuro-fuzzy approach performed better than the neural network model in terms of low RMSE error and less computational load (less performance time). As depicted in Figure 6 our experimentation results reveal that ANN trained by SCGA converged much faster than BP algorithm. Alternatively, BP training needs more epochs (longer training time) to achieve better performance. The neuro-fuzzy models considered on the other hand are easy to implement and produces desirable mapping function by training on the given data set. All the neuro-fuzzy models require information only about the input variables for generating forecasts thereby reducing the tedious analysis and trail and error methods as in the case of ARIMA model.

EFuNN makes use of the linguistic knowledge of FIS and the learning capability of neural networks. Hence, the neuro-fuzzy system is able to precisely model the uncertainty and imprecision within the data as well as to incorporate the learning ability of neural networks. Even though the performance of neuro-fuzzy systems is dependent on the problem domain, very often the results are better while compared to pure neural network approach. Compared to neural networks, an important advantage of neuro-fuzzy systems is its reasoning ability (*if-then* rules) of any particular state. A fully trained EFuNN could be replaced by a set of *if-then* rules. A simple example of a learned EFuNN learned rule is illustrated below.

"If the *maximum temperature* of the day is HIGH and *minimum temperature* of the day is LOW and *previous days demand* is MEDIUM and it is *summer* (HIGH) and *9.00 a.m.* (HIGH) and a *Monday* (HIGH) then the *electricity demand* is MEDIUM."

As EFuNN adopts a single pass training (1 epoch) it is more adaptable and easy for further on-line training which might be highly useful for on-line forecasting and bidding. Another important feature of EFuNN is that the user has the flexibility to construct the network (by selecting the parameters). Hence, for applications where speed is more important than the accuracy a faster network can be selected. However, an important disadvantage of EFuNN is the determination of the network parameters like number and type of MF for each input variable, sensitivity threshold, error threshold and the learning rates. Even though a trial and error approach is practical, when the problem becomes complicated (large number of input variables) determining the optimal parameters will be a tedious task.

The hybrid fuzzy neural network combine the advantages of fuzzy systems, which deal with explicit knowledge, which can be explained and understood, and neural networks, which deal with implicit knowledge, that could be acquired by learning. Neural network learning provides a good way to adjust the expert's knowledge and

automatically generate additional fuzzy rules and membership functions, to meet certain specifications and reduce design time and costs. The proposed FNN approach is found to be very powerful and robust for short-term load predictions. The simulated results using the FNN network are quite significant compared with the simple ANN and FIS based techniques. By using this hybrid approach, the load forecasting errors are reduced considerably. The advantage of the FNN is its adaptive tracking capability that has resulted in the development of a robust and effective forecasting technique. On the other hand, fuzzy logic enhances the generalization capability of a neural network system by providing more reliable output when extrapolation is needed beyond the limits of the training data. Furthermore, modified and new rules can be extracted from a properly trained hybrid network model, to explain how the results are derived.

Another interesting fact about the considered soft computing models are their robustness and capability to handle noisy and approximate data that are typical in power systems, and therefore, should be more reliable in worst situations.

Acknowledgements

Authors are grateful to the anonymous reviewers for the valuable comments which improved the clarity of the paper.

References

[1] Abe A. and Lan M. S. (1995), A Method for Fuzzy Rules Extraction Directly from Numerical Data and its Application to Pattern Classification, IEEE Transactions on Fuzzy Systems, 3(1): pp. 18-28.

[2] Abe S. and Lan M. S. (1995), Fuzzy Rule Extraction Directly from Numerical Data for Function Approximation, IEEE Trans. Systems, Man & Cybernetics, 25: pp. 119-129.

[3] Abraham A. (2000), An Evolving Fuzzy Neural Network Model Based Reactive Power Control, In Proceedings of The Second International Conference on Computers in Industry, Bahrain, pp. 247-253.

[4] Abraham A. (2001), Neuro-Fuzzy Systems: State-of-the-Art Modeling Techniques, Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, Jose Mira and Alberto Prieto (Eds.), Lecture Notes in Computer Science 2084, Springer-Verlag Germany, pp. 269-276.

[5] Abraham A. (2002), EvoNF: A Framework for Optimization of Fuzzy Inference Systems Using Neural Network Learning and Evolutionary Computation, The 17th IEEE International Symposium on Intelligent Control, ISIC'02 Canada, IEEE Press, Canada.

[6] Abraham A. and Nath B. (1999), Artificial Neural Networks for Intelligent Real Time Power Quality Monitoring Systems, First International Power & Energy Conference, INT-PEC'99, CD ROM Proceeding, Isreb M. (Editor), ISBN 0732 620 945, Australia,

[7] Abraham A. and Nath B. (2000), Evolutionary Design of Fuzzy Control Systems - An Hybrid Approach, The Sixth International Conference on Control, Automation, Robotics and Vision, (ICARCV 2000), December 2000.

[8] Abraham A. and Nath B. (2001), A Neuro-Fuzzy Approach for Forecasting Electricity Demand in Victoria, Applied Soft Computing Journal, Elsevier Science, Volume 1/ 2, pp.127-138.

[9] Cherkassky V. (1998), Fuzzy Inference Systems: A Critical Review, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Kayak O, Zadeh LA et al (Eds.), Springer, pp.177-197.

[10] Cordón O., Herrera F., Hoffmann F. and Magdalena L. (2001), Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientific Publishing Company, Singapore.

[11] Furuhashi T. (1997), Development of If-Then Rules with the Use of DNA Coding, Fuzzy Evolutionary Computation, Pedrycz W (Ed.), Kluwer Academic Publishers, pp.107-125.

[12] Hippert, H. S. Pedreira, C. E. and Souza R. C. (2001), Neural networks for short-term load forecasting: A review and Evaluation, IEEE Transactions on Power Systems, Vol. 16, No. 1, pp. 44-55, February 2001.

[13] Jager R. (1995), Fuzzy Logic in Control, PhD Thesis, Technische Universiteit Delft, Netherlands.

[14] Jang J.S.R. (1992), Neuro-Fuzzy Modeling: Architectures, Analyses and Applications, PhD Thesis, University of California, Berkeley.

[15] Jang J.S.R., Sun C T and Mizutani E (1997), Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence, Prentice Hall Inc, USA.

[16] Kasabov N. (1998), Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation, In Yamakawa T and Matsumoto G (Eds), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, pp. 271-274.

[17] Kasabov, N. and Woodford B. (1999), Rule Insertion and Rule Extraction from Evolving Fuzzy Neural Networks: Algorithms and Applications for Building Adaptive, Intelligent Expert Systems, In Proceedings of the FUZZ-IEEE'99 International Conference on Fuzzy Systems, Seoul, Korea, pp. 1406-1411.

[18] Kaufmann A. (1975), Introduction to the Theory of Fuzzy Subsets, New York, Academic Press.

[19] Khan M. R., Zak L., and Ondrusek C. (2001), Forecasting Weekly Load Using a Hybrid Fuzzy-Neural Network Approach, International Journal of

Engineering Mechanics, pp. 327-336, No. 5, ISSN 1210-2717, Czech Republic.

[20] Khan M.R. (2001), Short-term Load Forecasting for Large Distribution Systems Using Artificial Neural Networks and Fuzzy Logic, Ph.D. Thesis, UVEE, FEI, VUT Brno, Czech Republic.

[21] Khan M.R. and Abraham A. (2003), Short Term Load Forecasting Models in Czech Republic Using Soft Computing Techniques, International Journal of Knowledge-Based Intelligent Engineering Systems, United Kingdom, (forth coming).

[22] Khan M.R., Abraham A. and Ondrusek C. (2002), Soft Computing Models for Short-Term Load Forecasting in Czech Republic, 1st International Workshop on Hybrid Intelligent Systems, Physica Verlag, Germany, pp. 207-222.

[23] Khotanzad, A. Afkhami-Rohani, R. and Maratukulam, D. (1998), ANNSTLF-Artificial Neural Network Short-term Load Forecaster-Generation Tree, IEEE Transactions on Power Systems, Vol. 13, No. 4, pp. 1413-1422.

[24] Khotanzad, A. Hwang R. C. Abaye, A. and Maratukulam, D. (1995), An Adaptive Modular Artificial Neural Network Hourly Load Forecaster and its Implementation at Electric Utilities, IEEE Transactions on Power Systems, Vol. 10, No. 3, pp. 1716-1722.

[25] Kiartzis, S.J. Zoumas, C.E. Theocharis, J.B. Bakirtzis, A.G. and Petridis V. (1997), Short-term Load Forecasting in an Autonomous Power System Using Artificial Neural Networks, IEEE Transactions on Power Systems, Vol. 12, No. 4, pp. 1591-1596.

[26] Lin C.T. and Lee C.S.G. (1996), Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice Hall Inc, USA.

[27] Mori H, Hidenori K. (1996), Optimal Fuzzy Inference for Short-term Load Forecasting, IEEE Transactions on Power Systems, Vol. 11, No. 1, pp. 390-396.

[28] Nath B. and Nath M. (2000), Using Neural Networks and Statistical Methods for forecasting Electricity Demand in Victoria, International Journal of Management and Systems (IJOMAS), Special issue on Mathematics for Industry, Volume 16, No. 1, pp. 105-112.

[29] Nauk D., Klawonn F. and Kruse R. (1997), Foundations of Neuro-Fuzzy Systems, John Wiley & Sons Ltd, United Kingdom.

[30] Papalexopoulos A. D. Hao, S. Peng, T. M. (1994), An Implementation of a Neural Network Based Load Forecasting Model for the EMS, IEEE Transactions on Power Systems, Vol. 9, No. 4, pp. 1956-1962.

[31] Pedrycz W (Editor) (1997), Fuzzy Evolutionary Computation, Kluwer Academic Publishers, USA.

[32] Peng, T.M. Hubele, N.F. and Karady, G.G. (1992), Advancement in the Application of Neural Networks for Short-term Load Forecasting, IEEE Transactions on Power Systems, Vol. 7, No. 1, pp. 250-257.

[33] Piras, A. Germond, A. and Buchenel, B. Imhof, K. and Jaccard, Y. (1996), Heterogeneous Artificial Neural Network for Short-term Electrical Load Forecasting, IEEE Transactions on Power Systems, Vol. 11, No. 1, pp. 397-402.

[34] Procyk T J and Mamdani E H (1979), A Linguistic Self Organizing Process Controller, Automatica, Volume 15, pp. 15-30.

[35] Ranaweera D.K., Hubele N.F., Karady G.G. (1996), Fuzzy Logic for Short-Term Load Forecasting, Electrical Power and Energy Systems, Vol. 18, No. 4, pp. 215-222.

[36] Takagi T. and Sugeno M. (1983), Derivation of Fuzzy Control Rules from Human Operators Control Actions, Proceedings of the IAFC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis, pp 55-60.

[37] Tsukamoto Y. (1979), An Approach to Fuzzy Reasoning Method, Gupta MM et al (Eds.), Advances in Fuzzy Set Theory and Applications, pp. 137-149.

[38] Wang L.X. and Mendel J.M. (1992), Backpropagation fuzzy system as Nonlinear Dynamic System Identifiers, In Proceedings of the First IEEE International conference on Fuzzy Systems, San Diego, USA, pp. 1409-1418.

[39] Wang L.X. and Mendel J.M. (1992), Generating Fuzzy Rules by Learning from Examples, IEEE Transactions on Systems, Man and Cybernetics, Volume 22, No 6., pp. 1414-1427.

[40] Yoshinari Y., Pedrycz W., Hirota K. (1993), Construction of Fuzzy Models Through Clustering Techniques, Fuzzy Sets and Systems, Volume 54, pp. 157-165.

[41] Zadeh L.A. (1965), Fuzzy Sets, Information and Control, Volume 8: pp. 338-353.

[42] Zadeh L.A. (1973), Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Transactions on Systems, Man and Cybernetics, 3(1): pp. 28-44.