Control and Cybernetics

vol. **25** (1996) No. 2

Sequential fuzzy system identification

by

Andreas Bastian

Laboratory for International Fuzzy Engineering Research (LIFE),
Siber Hegner Bldg. 4 Fl.,
89-1 Yamashita-cho, Naka-ku, Yokohama 231,
Japan

Abstract: In this work a novel fuzzy system identification approach which identifies the input variables, the rule structure and the involved membership functions is introduced. In the main, the input variables are identified using neural networks and fuzzy logic, while the rule structure and the membership functions of the system are identified using a fuzzy-logic based sequential approach.

1. Introduction

The identification of linear models is by now a mature field and there exist a number of well-defined and very powerful identification algorithms. Yet, linearity is a mathematical idealization which in some cases might be valid only as an approximation to the real system. Therefore, various types of nonlinear models have been studied in the past, e.g. bilinear models, threshold autoregressive models and exponential autoregressive models. Since basically those nonlinear models are extensions of linear models, they are related to one major problem: the model structure identification. While for a given input-output data set it is quite easy to detect the optimal structure of a linear model, due to the large amount of possible structures, the structure identification of nonlinear models is very difficult if no basic knowledge about the system is provided.

Recently, new nonlinear modeling approaches based on neural networks, Narendra, Parthasarathy (1990), and/or fuzzy logic, Takagi, Sugeno (1985), have been proposed. Those approaches are not application dependent and allow an easy introduction of even high nonlinearities to the model. It should be noted, that while neural networks have better learning and representation capabilities, fuzzy models allow user knowledge based adjustments and model validation. An important work in this field is the fuzzy model structure identification method proposed by Sugeno and Kang (1988). In their approach, input

variables are partitioned and added one by one to the model until certain criteria can not be improved any more. Although very practical, this approach has two main limitations. The first one is that related variables can not always be separated. Take for instance the input-output relation $Y = \cos(X_1)(1 - \cos(X_1))$. The available data set is (X_1, X_2, Y), where the inputs X_1 and X_2 are related as follows: $X_2 = \cos(X_1)$. It is obvious that only X_1 or X_2 are needed to model the system. Yet, this fact can not always be identified by the Sugeno-Kang approach. This is partially also due to the modeling approach itself, which is the second limitation mentioned above. Since the representation capability of a fuzzy model depends strongly on the amount of rules and the partitioning of the input variables, the importance of an input variable can actually only be evaluated by comparing models with and without that input, if it can be guaranteed that the optimal fuzzy model for the present input variables can be constructed. Unfortunately, such a guarantee can not be given.

For the last reason, for the input variable identification, some authors used neural networks as the modeling tool instead, e.g. Takagi, Hayashi (1991), Horikawa, Furuhashi, Uchikawa, Tagawa (1991). Yet, the problem of identifying related input variables still existed. Therefore, Bastian and Gasós (1994A;B) proposed REIGN, an input identification approach which separates related variables by using a combined Bottom-Up and Top-Down approach based on the generalization ability of neural networks. The decision whether a variable should be used as an input or not is made based on a fixed threshold value. Since recent applications showed that this threshold depends on the data quality, in this work REIGN is extended by a fuzzy logic decision maker which adjusts the threshold.

In the second part of this paper, a novel approach named sequential fuzzy system identification which identifies the fuzzy rules and the involved membership functions is introduced.

This paper is organized as follows: in Section 2 the modified REIGN algorithm is briefly described and the fuzzy logic decision maker is introduced. In Section 3 the sequential identification approach is introduced, followed by two experimental results. Finally outlook and conclusions are provided in Section 4.

2. Input variable identification

2.1. Theoretical background of REIGN

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a vector of n input variables with $\mathbf{x} = (x_1, \ldots, x_n)$ being an instance of this vector, and let Y be the output variable with y being an instance of it. Let us assume that, given a bounded subset Y in the one dimensional Euclidean space and a bounded subset $\mathbf{X}$ in the n-dimensional Euclidean space, there exists a function f: $\mathbf{R}^n \to \mathbf{R}$ with $Y = f(\mathbf{X})$. The goal is to obtain the bounded subset $\mathbf{Z} \subseteq \mathbf{X}$ in the m-dimensional Euclidean space so that there exists a function g: $\mathbf{R}^m \to \mathbf{R}$ with $Y = g(\mathbf{Z})$ where $\mathbf{Z}$ has

minimum cardinality. In other words, $\mathbf{Z}$ is the smallest set of variables that contains enough information to represent the output Y.

For this purpose, the influence of removing input variables from $\mathbf{X}$ is observed. Since for a functional relation $Y = f(\mathbf{X})$ all instances of $\mathbf{X}$ with a constant value are bound to have the same output value Y, the removal of a variable X_p of $\mathbf{X}$ implies that all outputs $Y = f(\mathbf{X})$, where all instances of $\mathbf{X}$, except those of X_p, have a constant value, have to be represented by a single value $y'_p = h(x_1, \ldots, x_{p+1}, \ldots, x_n)$. The index p at y'_p denotes that X_p has been removed from $\mathbf{X}$.

A measure for the loss of function accuracy due to the removal of the variable X_p is the so-called *loss of information*:

$$LI_p = \int |Y - Y'_p| d\mathbf{X} \text{ with } Y'_p = h(X_1, \ldots, X_{p-1}, X_{p+1}, \ldots, X_n) \tag{1}$$

Having removed the input variable X_p from the vector of input variables, there exists an almost infinite number of possible functions h: $\mathbf{R}^{n-1} \rightarrow \mathbf{R}$ of the remaining input variables. Yet, given a criterion to measure the error, there exists a function h_1: $\mathbf{R}^{n-1} \rightarrow \mathbf{R}$, with $Y'_p = h_1(X_1, \ldots, X_{p-1}, X_{p+1}, \ldots, X_n)$, which has a minimum loss of information.

Since the size of the data vector is finite, the problem is to identify the smallest set of input variables needed to approximate a bounded function f: $A \subset \mathbf{R}^n \rightarrow \mathbf{R}$ from a bounded subset A of a n-dimensional Euclidean space to a bounded subset $f(A)$ in an one dimensional Euclidean space by means of examples $(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^k, y^k), y^k = f(\mathbf{x}^k)$. We can define the minimum loss of information for the discrete case:

$$MLI_p = \sum_{i=0}^{k} |y^i - y'^i_p| \tag{2}$$

where $y'_p = h_1(x_1, \ldots, x_{p-1}, x_{p+1}, \ldots, x_n)$ has the smallest representation error due to the removal of the variable X_p from the vector of input variables $\mathbf{X}$.

Since due to noise and outliers in the data the theoretical ideal set of inputs can not always be identified, we concentrate on finding the set of input variables with minimum cardinality that contains enough information to model the input-output relation with a required precision ϵ instead.

THEOREM 2.1 *Given an input-output data set:* $(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^k, y^k)$ *and given a modeling method that approximates the input-output relation with a precision* ϵ*, it is possible to determine the set of input variables with minimum cardinality that contains enough information to approximate the output* y *with a precision* ϵ*.*

Proof: Given a modeling method that approximates the input-output relation with a precision ϵ, the importance of input variables can be investigated

by removing them one at a time while watching the corresponding representation errors. The effect of removing a set of input variables $\mathbf{P}$ can be shown by computing the representation error of the model

$$RE_p = \sum_{i=0}^{k} |y^i - \tilde{y}_P^i| \leq \epsilon \tag{3}$$

where $\tilde{y}_{\mathbf{P}}^i$ is the output of the model with the set of input variables $\{\mathbf{X} - \mathbf{P}\}$.

Take, for instance, the influence of removing the variable X_p from the data set

- if $RE_p > \epsilon$ then the removed variable X_p is important and has to be included in the final set.
- if $RE_p \leq \epsilon$ then the remaining variables contain enough information to model the system without the removed variable X_p. However, such a variable can not be discarded right away since related variables will be found unimportant when considered independently. Instead, such a variable has to be classified as dubious. Consequently, the set of input variables $\mathbf{Z}$ with minimum cardinality which causes $RE_{\{\mathbf{X}-\mathbf{Z}\}} \leq \epsilon$ can be found by removing all possible combinations of dubious variables while watching the RE-values. ■

2.2. The basic idea of REIGN

In order to identify the input variables based on a given input-output data set, the importance of an input variable may be judged by watching the representation error due to its absence, as shown in the proof of Theorem 2.1. Unfortunately, removing input variables and their combinations to find the smallest set of input variables can be very time consuming. Recall that for $\mathbf{n}$ input variables the amount of possible combinations is:

$$\text{possible combinations} = 2^n - 1, \tag{4}$$

assuming that there exists always at least one important input variable.

Therefore, a combined Top-Down and Bottom-Up approach is used by REIGN to find the smallest set of input variables. The basic idea of REIGN is as follows: first the system is modeled using all available input variables. The representation error may be viewed as an estimation of the smallest error to which the system can be modeled. This initial representation error is henceforth called the *reference error*. If this reference error is larger than a pre-defined threshold ϵ, it means that the quality of the input-output data is too low or the modeling method cannot approximate the functional relation. In such cases, a new set of data, a different modeling method or different model parameters have to be employed. If the reference error is smaller than ϵ, the importance of a set of

inputs can be investigated by removing combinations of them as indicated in the proof of Theorem 2.1.

The selecting logic employed to avoid the necessity of removing all possible combinations of dubious variables consists of two basic parts:

- First the influence of each input variable is observed independently by replacing it with a random noise signal. If the model error increases sharply, then this particular input variable is important. If the error is small, then this input variable is classified as dubious.
- In the second step all dubious input variables are investigated by removing combinations of them. This step is removing all nonlinear related variables. In order to save computing time, a recursive algorithm is used (see Appendix A). Roughly described, this algorithm works as follows: the first dubious variable is declared as the starting variable. Consequently, other dubious variables are replaced additionally by noise signals until the representation error exceeds a pre-defined threshold. In this case, the last replaced dubious variable, D_i, is returned to the input set (this algorithm step is henceforth named *stepping back*), and the next dubious variable, D_{i+1}, is additionally replaced. The algorithm also steps back if no new possible combination can be replaced. If the algorithm steps back until the starting variable, it is excluded from the set of dubious variables, and the next dubious variable is employed as the new starting variable. Since the goal of the algorithm is to remove as many variables as possible, solutions with a smaller number of variables than the present best set are never considered.

Notice that input variables are replaced by random signals, and not simply deleted. This measure ensures that the modeling/learning task is kept at a constant level. The following example highlights the way REIGN functions.

EXAMPLE 2.1 *Consider a set of five variables* $(X_1, \ldots, X_5)$. *Let us assume that after removing input variables independently (first step of the algorithm) four dubious inputs, denoted* $D = (D_1, D_2, D_3, D_4)$, *are found. Consequently, those variables have to be investigated.*

Assumed that the replacement of the combination $D_1 - D_2$ *yields a small representation error, in the following step the combination* $D_1 - D_2 - D_3$ *is replaced. Let's say that in this case the error is big. Thus, the algorithm goes one step back and replaces* $D_1 - D_2 - D_4$. *As mentioned, solutions with a smaller number of variables than the present best set are never considered. So if the algorithm has found a good solution by removing 3 variables, combinations of 2 variables, e.g.* $D_1 - D_3$, *are no longer considered.*

2.3. Applying REIGN using feedforward neural networks

Although this algorithm can be employed with any nonlinear modeling method, feedforward backpropagation neural networks Rumelhart, Hinton and Williams

(1986) are employed for the following reasons:

1. *they are universal approximates.*
   It was proved by Hornik, Stinchcomb, White (1989) that any continuous mapping over a compact domain can be accurately approximated by a three-layered feedforward neural network. Thus for a given error $\epsilon > 0$ a function $f(\mathbf{X})$ defined on $[0,1]^n$ for all $\mathbf{x} \in [0,1]^n$ can be approximated so that
   $$\left| f^k(x) - \sum_{i=1}^{m} v_1 \sigma \left(\sum_{j=1}^{n} w_{ij} x_j^k - \Theta_i \right) \right| < \epsilon, \tag{5}$$
   where $\sigma()$ is a bounded sigmoidal function, and v_i, w_{ij} and Θ_i are real numbers.
2. *no knowledge about the model structure is needed.*

In this study, fixed parameters of the backpropagation algorithm are throughout used in order to enable a fair comparison of the models. The learning rate is set to 0.7, while the momentum factor is set to 0.3.

It is well known that the mapping capability of a network depends on the number of layers and hidden units. An oversized network will loose its generalization ability and memorizes the training data instead, while a small network with few hidden layers and processing elements might not be adequate for a complex problem. Thus, the size of the network and its learning capability should be kept close to the complexity of the problem to be solved. Since there is no formal way for computing the network structure as a function of the complexity of the problem, the network structure is often selected by trial and error or by using algorithms that dynamically modify the structure of the network based on some penalty functions, e.g. estimating the sensitivity of the error function to the removal of an element, or rewarding the net for choosing efficient solutions.

In this study, the structure of the neural networks are dynamically configured using the algorithm proposed by the author, Bastian (1994), where feature extraction of the data by employing the Fuzzy C-Means (FCM) clustering algorithm, Bezdek (1981), was proposed to determine the initial number of hidden units. Having determined the initial network size, a hidden neuron is added. If the modeling performance increases, then neurons are recursively added until the representation capability of the network does not increase significantly. If the representation capability of the net does not increase after adding the first hidden neuron, then neurons are going to be deleted recursively from the hidden layer until a large decrease of the representation error occurs. The main advantage of this pruning algorithm is the small computing time. Since this algorithm was already published and explained in Bastian (1994), the interested reader is referred to that publication. The algorithm itself is given in Appendix B.

E^A/E^B	Big	Medium	Small	Very Small
Big	very small	very small	small	medium
Medium	very small	medium	quite big	quite big
Small	small	quite big	big	big
Very Small	medium	quite big	big	very big

Table 1. Rule base

2.4. The fuzzy logic decision maker

In the REIGN-algorithm the decision whether a variable should be classified as important or not depends on a fixed threshold value T which is obtained by comparing the actual RC-value, which is a measure of the model accuracy, with the RC-value of the initial model, the so-called reference value RC_{ref}. Yet, a fixed threshold T is connected with some problems. Consider the following example: the first obtained reference value RC_{ref} is 0.00001, which is actually a quite good representation. The removal of one input causes a RC-value of 0.00002, which is also very low. However the ratio RC/RC_{ref} is 2.0 which is quite high and would indicate that this input is decisive if the threshold T is set to values below 2.0.

Further experiments showed that if the REIGN algorithm was applied to data with varying noise, in order to identify the inputs correctly, the threshold T should vary between 1.2 and 5.4 depending on the noise. It also turned out that the individual representation error of a network is also decisive. Thus, taking the average of the network errors to judge the model performance is not adequate in cases where the difference between the network errors is big.

Therefore fuzzy logic rules are employed to determine the threshold T. The inputs of the rules are the representation errors of the networks M^A and M^B:

$$E^A = \sqrt{\sum_{i=1}^{k_A} (y_i^A - y_i^{AB})^2 / k_A} \tag{6}$$

and

$$E^B = \sqrt{\sum_{i=1}^{k_B} (y_i^B - y_i^{BA})^2 / k_B} \tag{7}$$

where k_A and k_B are, respectively, the numbers of data in the training sets A and B of the networks, y^A and y^B are the desired outputs, y^{AB} is the output obtained for the data set A using the trained network M^B, and y^{BA} is the output of the network M^A using the data set B.

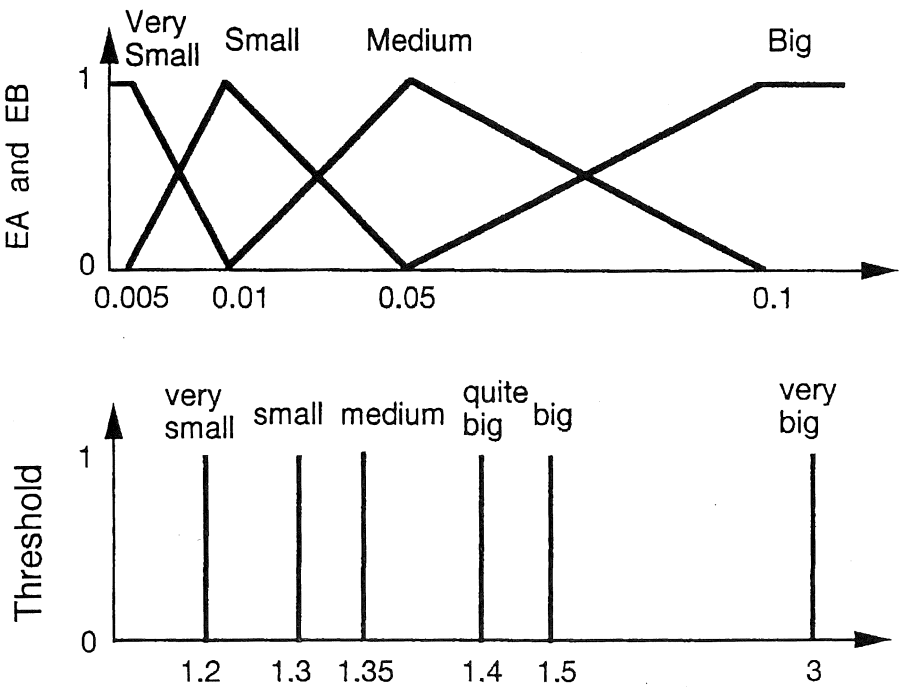

Figure 1. Membership functions

The consequent of the rules is the threshold T. Table 1 shows the rule base. One rule is for example:

$$\text{IF } (E^A \text{ is small}) \text{ AND } (E^B \text{ is small}) \text{ THEN } (T \text{ is big}) \tag{8}$$

Figure 1 shows the membership functions of the linguistic values used in the rules.

In this study the max-prod inference method is used. The degree of match in the premises derived for the i.th rule of the type

$$\text{IF (x is A) AND } (y \text{ is } B) \text{ THEN } (z \text{ is } C), \tag{9}$$

with $1 \leq i \leq m$, is given by

$$\mu_{C_i^*}(z) = \mu_{A_i}(x_0)\mu_{B_i}(y_0)\mu_{C_i}(z) \tag{10}$$

where x, y and z are linguistic variables representing the inputs and output of the model, x_0 and y_0 are the crisp inputs, and A, B and C are the terms of the

linguistic variables x, y, and z in the universes of discourse X, Y and Z. The final consequence from all rules is obtained by taking the union:

$$\mu_{C^*}(z) = \mu_{C_1^*}(z) \vee \mu_{C_2^*}(z) \vee \mu_{C_3^*}(z) \vee \ldots \vee \mu_{C_m^*}(z) \tag{11}$$

where $\vee$ stands for the maximum. The representative crisp value $\underline{z}$ for the resulting fuzzy set C is obtained by using the center of gravity defuzzification strategy:

$$\underline{z} = \frac{\sum_j z_j \mu_{C^*}(z_j)}{\sum_j \mu_{C^*}(z_j)} \tag{12}$$

To adjust the threshold T to the initial reference value, the inferred output T is multiplied with the function $t(RC_{ref})$:

$$t(RC_{ref}) = 1.0 + \frac{0.00001}{RC_{ref}}. \tag{13}$$

EXAMPLE 2.2 *The new REIGN is applied to identify the inputs of the nonlinear function:*

$$z = 0.5x_1^3 + 1.2x_2^2 + x_3 \tag{14}$$

300 data sets with the following five inputs were presented to the algorithm:

$$0 \leq x_1 \leq 5;\ 0 \leq x_2, x_3 x_4 \leq 2;\ x_5 = x_2^2 \tag{15}$$

Notice the dummy inputs x_4 and x_5, with x_5 being related to x_2. Table 2 shows the result after the first steps (RC_{ref}-value of 0.00014). Only the removal of x_1 and x_3 causes a big representation error. Thus, those variables are classified as important, while the dubious variables x_2, x_4 and x_5 have to be observed in the next steps.

Table 3 shows that a removal of x_2 and x_4 causes a small error. Removing additionally x_5 causes an error bigger than the threshold $T = 1.501$, which means that all three variables can not be removed. Thus, in the next step the combination $x_2 - x_5$ is removed which yields a big error. Finally the inputs $x_4 - x_5$ are removed. Since the removal of the combination $x_2 - x_4$ causes a smaller error than the removal of $x_4 - x_5$, the identified input set is x_1, x_3 and x_5.

Notice that the algorithm identified x_5 instead of x_2. Yet, this result is also correct. Important is, that the algorithm succeeded in separating the inputs x_2 and x_5.

Replaced variable	RC/RC_{ref}	Threshold
x_1	325.8	1.212
x_2	0.7151	1.598
x_3	3.4094	1.605
x_4	0.4092	2.177
x_5	1.1219	1.585

Table 2. Result after the first steps

Replaced variables	RC/RC_{ref}	Threshold
$x_2 - x_4$	0.5031	1.951
$x_2 - x_4 - x_5$	1.6031	1.561
$x_2 - x_5$	2.0422	1.560
$x_4 - x_5$	0.9462	1.586

Table 3. Final result

EXAMPLE 2.3 *In this example we identify the inputs of the function:*

$$z = 0.5x_1^3 + x_4^2 + x_5 \tag{16}$$

with the inputs:

$$0 \leq x_1 \leq 5;\ 0 \leq x_2 = x_1^2;\ , x_3 = x_1^3;\ 0 \leq x_4, x_5 \leq 2 \tag{17}$$

Notice that all variables are related to the output. Thus, no "real" dummy inputs are given. The results after the first steps are displayed in Table 4 (RC_{ref}-value = 0.000009).

According to this result the inputs x_2 and x_3 are selected to be observed in the next step, while the other inputs are considered important. The removal of both x_2 and x_3 causes an error RC/RC_{ref} of 1.10612 which implies low importance of those inputs. Thus, they are removed from the input set.

Comparing the threshold values in the Examples 2.2 and 2.3 it again becomes obvious that only one fixed threshold value is not sufficient for identifying input variables at cases where all data are closely related to the output.

3. Sequential system identification

3.1. Basic Idea

Having identified the inputs, the next step towards the identification of the fuzzy model is the rule and parameter identification. For this purpose, a novel

Replaced variable	RC/RC_{ref}	Threshold
x_1	5.42121	4.794
x_2	1.01272	6.282
x_3	3.98273	4.790
x_4	120.021	2.533
x_5	80.0202	3.007

Table 4. Result after the first steps

approach called sequential system identification is introduced in this section. The basic algorithm is as follows:

1. Identify the inputs of the model using the modified REIGN algorithm.
2. Cluster the most important input, that is, the input variable whose removal causes the biggest representation error, in dependency of the output variable.
3. Construct triangular membership functions of the most important input based on the resulting clusters. The core of each membership function is the cluster center.
4. Project each membership function to the next input variable and construct the corresponding membership functions. Note that a projection can yield several membership functions. The construction of the membership functions is done by utilizing the downhill simplex method proposed by Nelder and Mead (1994), an optimization method which requires only function evaluation. The error to be minimized is:
$$Error = \sum(\mu - \mu_{opt}) + M\beta \tag{18}$$
where the first term describes the representation error of the designed membership functions, while the second term punishes the extensive construction of membership functions. M is the amount of the constructed membership functions, and β is a sensitivity factor. Note that performing an optimization at this early stage helps to find the optimal rule structure. If due to the existing data no meaningful regression can be performed, this input variable (only for the present rule) is not employed.
5. Project the resulting membership functions to the next input variable and construct the corresponding membership functions using the procedure described in Step 4.
6. Repeat step 5 until the membership functions of the last input variable are projected to the output variable.
7. Calculate the mean squared error (MSE) of this initial model and the average error of each rule which is the MSE divided by the number of rules.
8. Perform a fine tuning of the model using the downhill simplex method.

9. Validate each rule by calculating the error caused by it. If the error of a rule is bigger than the double amount of the average error, split the involved membership function of the most import input variable and repeat the algorithm from step 4. Else write down the rules and quit.

One interesting feature of this identification approach is the fact that different rule structures with different inputs are identified.

EXAMPLE 3.1 *Given two identified inputs, with Input 1 being the most important input. The clustering of Input 1 in dependency to the output yields 2 clusters. Thus, Input 1 is partitioned into two membership functions (***Big** *and* **Small**, *respectively).*

The projection of the membership function **Small** *of Input 1 results in one membership function, while the projection of the membership function* **Big** *yields two membership functions. In the next step, those three membership functions of the Input 2 are projected to the output space, resulting in three output membership functions. This sequential rule construction is shown in Figure 2.*

The identified rules are:

$$\begin{aligned} &\textit{IF (Input 1 is } \textbf{Small}\textit{) AND (Input 2 is } \textbf{Small}\textit{) THEN (Output is } \textbf{Small}\textit{)} \\ &\textit{IF (Input 1 is } \textbf{Big}\textit{) AND (Input 2 is } \textbf{Medium}\textit{) THEN (Output is } \textbf{Medium}\textit{)} \\ &\textit{IF (Input 1 is } \textbf{Big}\textit{) AND (Input 2 is } \textbf{Big}\textit{) THEN (Output is } \textbf{Big}\textit{)} \end{aligned} \tag{19}$$

In the following step, each the error caused by each rule is calculated. Let's assume that the error of the first rule exceeds a pre-determined threshold. Therefore, the membership function **Small** *of Input 1 is divided into two (***Small** *and* **Medium***), and each new membership function is projected to Input 2. Again, depending on the data, new rules are created. A possible outcome is shown in Figure 3.*

3.2. Application examples

3.2.1. Human operation of a chemical plant

The proposed system identification approach was tested using data of a human operator's control of a chemical plant producing a polymer by the polymerization of some monomers Sugeno, Yasukawa (1993). The operator determines the set point for the monomer flow rate and passes this information to a PID controller, which calculates the actual monomer flow rate input for the plant. There are 70 data sets, each set consisting of five inputs, $\mathbf{X}$ =(monomer concentration, change of monomer concentration, monomer flow rate, temperature 1 and temperature 2 inside the plant), and one single output, namely the set point for the monomer flow rate.

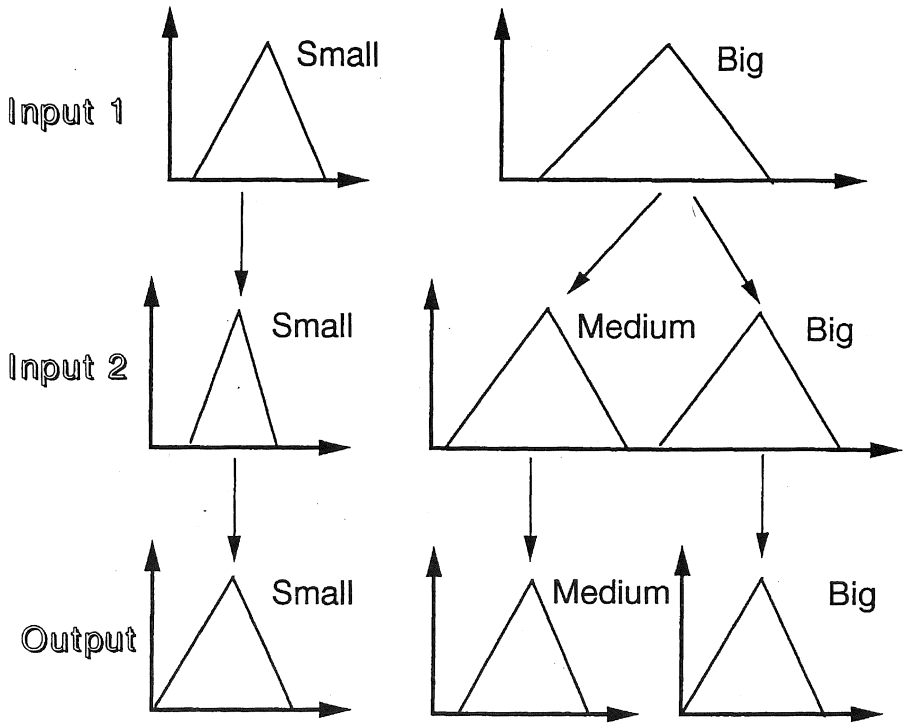

Figure 2. Constructing rules and membership functions

Neurons in hidden layer	4	3	2
Error	0.000119	0.000119	0.01013

Table 5. Result of the net identification algorithm

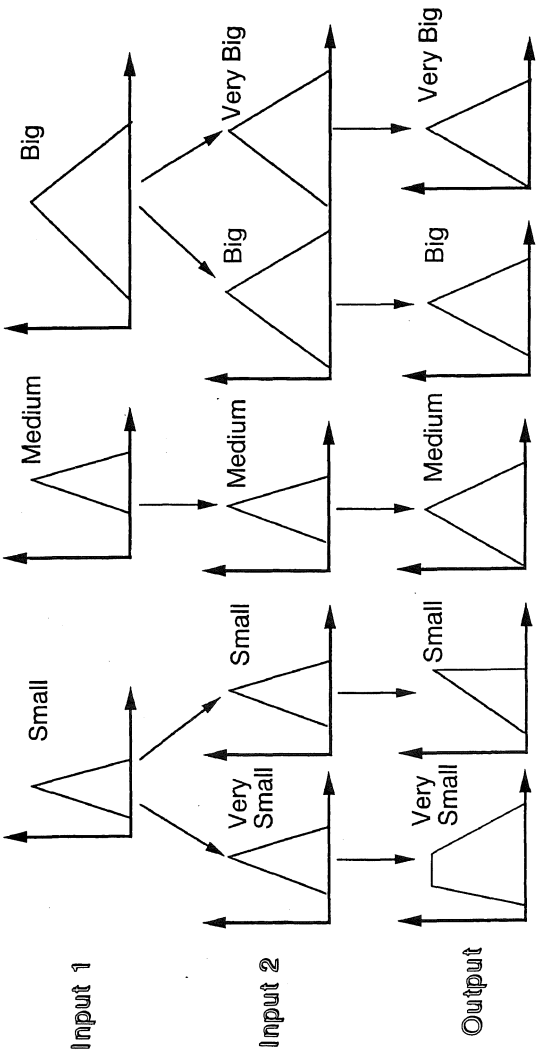

Figure 3. Final rules and their membership functions

Replaced variable	RC/RC_{ref}
X_1	2.11
X_2	1.02
X_3	45.03
X_4	0.99
X_5	0.88
$X_2 - X_4$	1.57
$X_2 - X_4 - X_5$	1.59

Table 6. Replaced variables and the corresponding ratios RC/RC_{ref}

The pruning algorithm yielded a network with three layers and three units in the hidden layer. The results are displayed in Table 5.

Using all available inputs a reference value $RC_{ref} = 0.000119$ was computed. The fuzzy logic decision maker issued threshold values around 1.6. Then, replacing each input variable with a random noise signal, the algorithm identified the monomer concentration and the monomer flow rate as the important input variables. Further study of the remaining (dubious) variables showed that no other input variable is necessary. The results obtained by the algorithm are shown in Table 6.

The FCM algorithm yielded 6 clusters. The sequential system identification resulted also in 6 rules, which means that in no case a projection of a membership function yielded 2 membership functions. In this example, the fuzzy model uses rules of the IF-THEN type as shown in (9), and employs the max-prod inference method with the center of gravity defuzzification. Figure 4 shows the resulting rules and their membership functions.

In Figure 5 the model performance is compared to the original data. As one can see, using only two input variables, a good fit to the model could be achieved.

3.2.2. Box-Jenkins gas furnace data

The work of Box and Jenkins (1970) is well known, and their data have become a kind of benchmark for identification techniques. For this reason, the ability of the proposed identification approach was further tested using those gas furnace data. The data set consists of 296 pairs of input-output observation where the input $u(t)$ is the gas flow rate into the furnace and the output $y(t)$ is the CO_2-concentration in the outlet. The sampling rate is nine seconds. Since the process is dynamical, ten candidates for the input variables were selected, namely $\{u(t-1), \ldots, u(t-6), y(t-1), \ldots, t(t-4)\}$.

The pruning algorithm yielded a network with three layers and eight units in

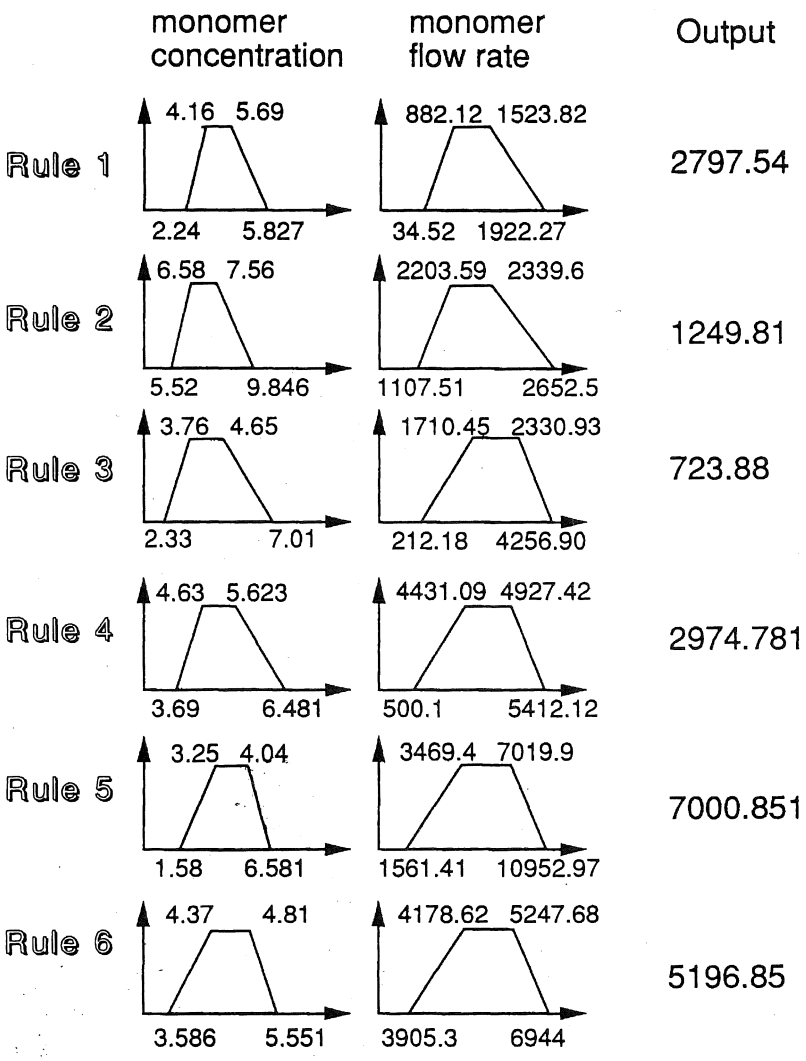

Figure 4. Rules and membership functions of the human operator

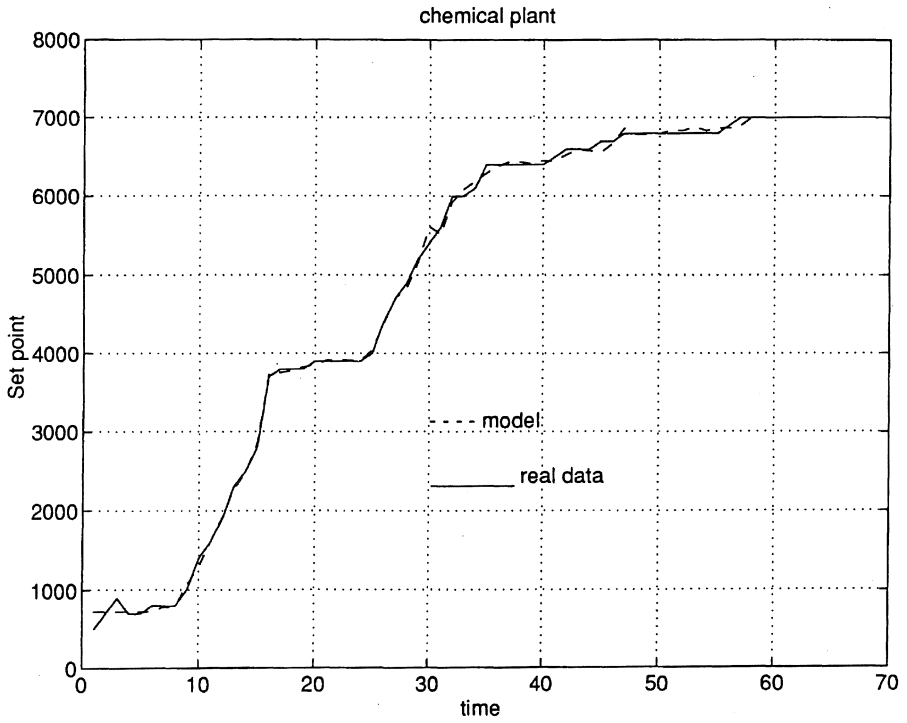

Figure 5. System identification result

Replaced variable	RC/RC_{ref}
$u(t-1)$	0.9678
$u(t-2)$	1.0813
$u(t-3)$	0.9981
$u(t-4)$	1.2060
$u(t-5)$	1.1724
$u(t-6)$	1.1094
$y(t-1)$	3.4872
$y(t-1)$	1.3698
$y(t-1)$	1.0471
$y(t-1)$	1.0939

Table 7. Replaced variables and the corresponding ratios RC/RC_{ref}

the hidden layer. Using all available inputs a reference value $RC_{ref} = 0.000219$ was computed. The fuzzy logic decision maker issued threshold values around 2.35.

Then, replacing each input variable by a random signal, in the first stage the algorithm only identified $y(t-1)$ as the important input variables, thus leaving nine dubious variables to be investigated in the following stage. Table 7 shows the results after the first stage.

In the following stage, the combinations of dubious variables were removed. After 21 trials the algorithm already found a minimal input set consisting of only three input variables. Thus, only combinations of at least seven variables were consequently investigated in 15 trials. The algorithm then terminated with the following identified inputs: $y(t-1)$, $y(t-2)$, and $u(t-3)$.

The FCM algorithm yielded 6 clusters. The sequential system identification resulted in 8 rules. Those rules and the membership functions are shown in Figure 6.

In Figure 7 the model performance is compared to the original data. The mean squared error (MSE) of the model is 0.063495, which is even smaller than the one achieved using the Takagi-Sugeno model as shown in Table 8.

4. Outlook and conclusions

A sequential system identification approach was introduced in this paper. In order to perform an input identification, the REIGN algorithm was extended by a fuzzy logic decision maker which provides a flexible decision threshold. Using, a sequential approach, rules and also membership functions could be identified. The fine tuning was done by employing the downhill simplex method. Using this approach, the model of a human operator of a chemical plant, and also the

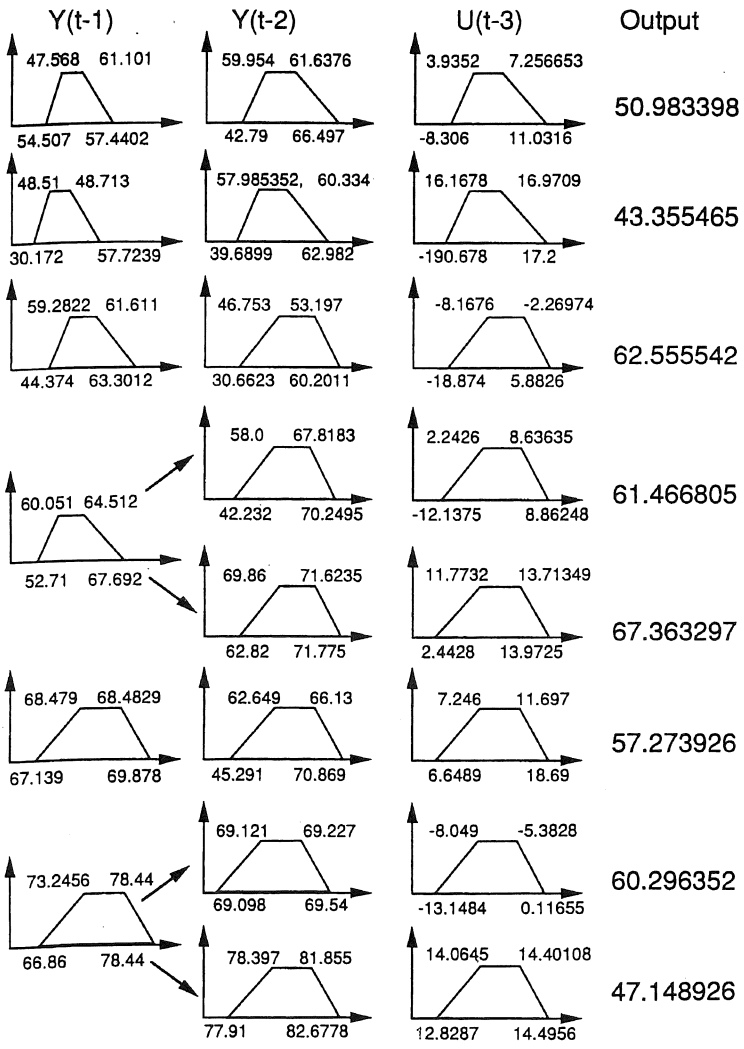

Figure 6. Rules and membership functions of the gas data model

Approach by	Inputs	Number of rules	MSE
Tong (1980)	$y(t-1), u(t-4)$	19	0.469
Sugeno-Yasukawa (1993)	$y(t-1), u(t-4), u(t-3)$	6	0.190
Pedrycz (1984)	$y(t-1), u(t-4)$	81	0.320
Xu (1988)	$y(t-1), u(t-4)$	25	0.328
Takagi-Sugeno (1993)	$y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3)$	2	0.068
Bastian	$y(t-1), y(t-2), u(t-3)$	8	0.063

Table 8. Comparison of various model results

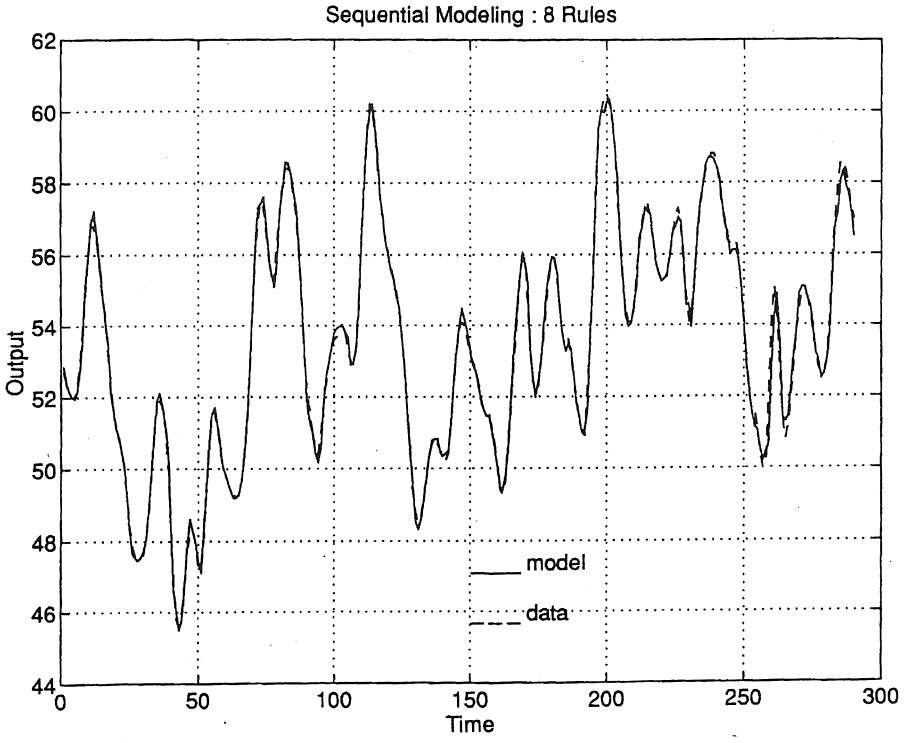

Figure 7. System identification result

model of the Box-Jenkins gas furnace data were identified successfully.

Further studies are going to be conducted to embed the REIGN algorithm directly into the fuzzy model in order to save computing time. Here, the main problem faced is the limited representation capability of the fuzzy model when not enough rules are provided.

Acknowledgment

The author wishes to sincerely thank Prof. T. Terano, the director of LIFE, for his kind guidance and everlasting support during the conduct of this study. Also the fruitful discussions with Prof. M. Mukaidono, Meiji University, are highly appreciated.

References

BASTIAN, A. (1994) An Effective Way to Generate Neural Network Structures for Function Approximation. *Mathware*, **1**, 1, 139-161, December 1994.

BASTIAN, A., GASÓS, J. (1994A) Modeling Using Regularity Criterion Based Constructed Neural Networks. *Proc. 16th Int. Conf. on Comp. & Indust. Eng.*, 237-240, Ashikaga, 1994.

BASTIAN, A., GASÓS, J. (1994B) REIGN: Recursive Identification by Generalizing Neural Networks. *LIFE Technical Report 1994.*

BEZDEK, J.C. (1981) *Pattern Recognition with Fuzzy Objective Function Algorithm.* New York, Plenum Press.

BOX, G.E.P., JENKINS, G.M. (1970) *Time Series Analysis, Forecast and Control.* Holden Day, San Francisco.

HORIKAWA, S., FURUHASHI, T., UCHIKAWA, Y., TAGAWA, T. (1991) A study on fuzzy modeling using fuzzy neural networks. *Proc. IFES'91*, 562-572.

HORNIK, K., STINCHCOMB, M., WHITE, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359-366.

IHARA, J. (1980) Group method of data handling towards a modeling of complex systems-IV. *System and control* (in Japanese), **24**, 158-168.

NARENDRA, K.S., PARTHASARATHY, K. (1990) Identification and control of dynamical systems using neural networks. *IEEE Trans. on NN*, **1**, 1, 237-240.

NELDER, J.A., MEAD, R. (1965) Downhill simplex method. *Computer Journal*, **7**, 308-313.

PEDRYCZ, W. (1984) An identification algorithm in fuzzy relational systems. *Fuzzy Sets and Systems*, **13**, 153-167.

RUMELHART, D.E., HINTON, G.E., WILLIAMS, R.J. (1986) Learning representation by back-propagating errors. *Nature*, **232**, 533-536.

SUGENO, M., KANG, G.T. (1988) Structure identification of fuzzy model. *Fuzzy Sets and Systems*, **28**, 15-33.

SUGENO, M., YASUKAWA, T. (1993) A fuzzy-logic-based approach to qualitative modeling. *IEEE Trans. on Fuzzy Systems*, **1**, 1, 7-31.

TAKAGI, H., HAYASHI, I. (1991) Artificial neural network driven fuzzy reasoning. *Int. J. of Approximate Reasoning*, **5**, 3, 191-212.

TAKAGI, T., SUGENO, M. (1985) Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man, Cybernetics*, **15**, 116-132.

TONG, R.M. (1980) The evaluation of fuzzy models derived from experimental data. *Fuzzy Sets and Systems*, **4**, 1-12.

XU, C.W. (1988) Fuzzy Systems Identification. *IEE Proc*, **136**, Pt D, 146-150.

Appendix

A. REIGN: Recursive Identification by Generalizing Neural Networks

The basic algorithm of REIGN is, see Bastian, Gasós (1994):

1. Identify the network size using the algorithm given in Appendix B.
2. Divide the input/output data into two groups: A and B.
3. Using all **n** available input variables, obtain a model M_A^n using data A and a model M_B^n using data B. The models are utilizing three-layered feedforward neural networks, employing back-propagation learning where the network learns to map a set of inputs to a set of outputs by adjusting the network weights w by:
$$\Delta w_{ji}(t+1) = \eta \delta_{xj} o_{xi} + \alpha \Delta w_{ji}(t) \tag{20}$$
where t, η, δ_{xj}, o_{xi}, α, and w_{ji} are respectively the presentation number, the learning rate, the error signal for unit j, the activation of unit i as a result of the input pattern x, the momentum factor and the weight from unit i to unit j.
4. After training, the reference network M_A^n is tested using data set B and reference network M_B^n is tested using data set A.
Compute the RC, Ihara (1980):
$$RC = \left[\sum_{i=1}^{k_A} (y_i^A - y_i^{AB})^2 / k_A + \sum_{i=1}^{k_B} (y_i^B - y_i^{BA})^2 / k_B \right] / 2 \tag{21}$$
where: k_A and k_B are respectively the number of data in sets A and B, y^A and y^B are the output data in sets A and B, y^{AB} is the output obtained for the data set A using the network M_B^n, and y^{BA} is the output obtained for data set B using the network M_A^n. This calculated RC will be used as a reference value.
If the reference value is bigger than a defined threshold ϵ then the algorithm is terminated since the data quality or the modeling method is not sufficient. Else continue with Step 5.
5. Repeat the Steps 3-4 for $i = 1 \ldots n$ replacing each input variable X_i in the test sets A and B with a random variable and compute the RC_i according

to (21).
Calculate the ratio:

$$RC_{ratio} = RC/RC_{ref} \tag{22}$$

6. Replaced input variables which cause a high RC_{ratio} are considered important variables, denoted $I_1, I_2, \ldots, I_p$, and should not be replaced by random variables anymore, while the replaced variables which cause a low RC_{ratio} are regarded as dubious variables, denoted $D_1, \ldots, D_q$.
If $(q = 0)$ all input variables are necessary and the algorithm is terminated.
Else save the best input selection in the 3-tuple:

$$\Re = \{var; conf; RC_{\min}\}, \tag{23}$$

where var is the number of input variables not replaced by random signals, $conf$ is the input variable configuration that generated the smallest RC, and $RC_{\min}$: is the smallest RC_{ratio}.
7. In this step all combinations of two dubious variables are generated and the recursive routine next_input_combination in Step 8 is called to check them.

```
for i = 1,...,q-1
  {
    replace input variable i with a random signal;
    for j = i+1,...,q
      {
        if (amount_of_variables ¿= conf:)
          next_input_combination (j); /* see Step 8 */
      }
    recover original values of variable i;
  }
```

8. The recursive routine next_input_combination receives an input configuration and calculates the RC_{ratio}. Is the RC_{ratio} small then it checks whether this input variable combination is the best up to this time, and continues replacing the next dubious input variable with a random signal. Otherwise the algorithm returns to the calling function.

```
next_input_combination (additional_input)
  {
    replace additional_input with a random signal;
    train the networks;
    calculate the RC_ratio for this configuration;
    if (RC_ratio is big)
      {
        recover original values of variable additional_input;
        return;
      }
    if (RC_ratio is small)
```

```
{
  /* N is the number of variables that keep their
  original values in this configuration */
  if (N < var OR (N = var AND RC_ratio < RC_min))
    save results in the tuple ℜ = {var; conf; RC_min};
  for k = additional_input + 1, ..., q
    {
      next_input_combination (k);
    }
  }
recover original values of variable additional_input;
return;
}
```

B. Algorithm to determine the optimal number of hidden units

Find the optimal network structure, Bastian (1994):

1. Determine the number of features $\mathbf{f}$ by clustering the output using the FCM-algorithm.
2. Divide the data into training and validation set.
3. Generate a three-layered network with $\mathbf{q} = \mathbf{f}$ neurons in its hidden layer.
4. Train the network using the training data. The training is terminated if the error on the validation set begins to rise or 50000 training cycles have been completed.
5. Calculate the mean square error $E(q)$ on the training set.

$$E(q) = \sum_{i=1}^{k} (y_i - t_i^{net})^2 / k, \tag{24}$$

where k is the number of data, y_i is the desired output, y_i^{net} is the output of the net, and q the number of neurons in the hidden layer as optimal solution.

6. Add one hidden unit, initialize the network and repeat the Steps 4 and 5. If

$$\frac{E(q) - E(q+1)}{E(q)} \geq 0.01 \tag{25}$$

then set $\mathbf{q} = \mathbf{q} + 1$, and repeat Step 6.
Else go to Step 7.

7. If $\mathbf{q} = \mathbf{f}$ go to Step 8. Else terminate the algorithm with q neurons in the hidden layer.
8. Remove one neuron from the hidden layer, initialize the network and repeat the Steps 4 and 5. If

$$\frac{E(q+1) - E(q)}{E(q)} \geq 0.01 \tag{26}$$

then set $\mathbf{q} = \mathbf{q} - 1$, and repeat Step 8.
Else terminate the algorithm with q neurons in the hidden layer as optimal solution.