Improvements and Critique on Sugeno's and Yasukawa's Qualitative Modeling

Domonkos Tikk, György Biró, Tamás D. Gedeon, László T. Kóczy, *Member, IEEE*, and Jae Dong Yang

***Abstract*—This paper investigates Sugeno's and Yasukawa's qualitative fuzzy modeling approach. We propose some easily implementable solutions for the unclear details of the original paper, such as trapezoid approximation of membership functions, rule creation from sample data points, and selection of important variables. We further suggest an improved parameter identification algorithm to be applied instead of the original one. These details are crucial concerning the method's performance as it is shown in a comparative analysis and helps to improve the accuracy of the built-up model. Finally, we propose a possible further rule base reduction which can be applied successfully in certain cases. This improvement reduces the time requirement of the method by up to 16% in our experiments.**

***Index Terms*—Complexity reduction, feature selection, fuzzy modeling, membership function approximation, parameter identification, structure identification, Sugeno–Yasukawa (SY) method.**

I. Introduction

THIS PAPER deals with Sugeno's and Yasukawa's (SY) qualitative fuzzy modeling [1]. This method creates a fuzzy rule base (a set of fuzzy IF–THEN rules) from sample input–output data, and assigns meaningful linguistic labels to the fuzzy sets in the rule base. This assignment is very important in fuzzy systems because it makes the behavior of the system, which is modeled by the rule base, easily interpretable and transparent. In the original paper, there are some details which are not quite clear, thus, require clarification or leave room for improvement. These problems concern the determination of trapezoid membership function, the rule projection from sample data, the selection of important variables and parameter identification.

One of the advantageous properties of the SY method is that it produces only the necessary rules (more details in Section II), which is usually not a full (i.e., dense) rule base. Due to this property an apparent extension of the SY method could be its combination with inference techniques for sparse rule bases, such as fuzzy rule interpolation (see, e.g., [2]–[5]) or compatibility modification inference [6] and other approximate reasoning techniques [7], [8]. Although, it is out of the scope of this paper to investigate this rather straightforward extension of the SY method, we would remark a joint point of the referred papers and SY method, namely, the time complexity issue. The previously listed techniques were proposed partly to reduce the original exponential time complexity [9] of classical fuzzy controllers [10], e.g., by omitting redundant or replaceable fuzzy sets. The SY method executes this selection originally. To enhance this property, we also introduce an improvement for further rule base reduction.

The remaining of this paper is organized as follows. Section II describes the original SY method. Section III presents our solutions to the unclear details of the original method, namely, trapezoid approximation (Section III-A), an algorithm to determine the number of rules (Section III-B), and an improved parameter identification procedure (Section III-D). Some critical remarks on the regularity criterion (RC) method used for the selection of effective input variables in [1] can be found in Section III-C. Section IV contains the proposed rule base reduction technique. Finally, Section V concludes the paper.

II. The SY Method

The goal of the SY fuzzy modeling method is to create a transparent, i.e., linguistically interpretable fuzzy rule based model from input–output sample data. The construction of the rule base is performed in two main steps: *identification* and the build-up of the *qualitative model*. The former can be further divided into two tasks: structure identification and parameter identification. Having an identified model at hand, linguistic labels can be assigned to the finalized fuzzy sets in the rules in the qualitative modeling phase. In this paper, we focus solely on the identification step.

In [1], the authors classified the structure identification task into two types. The type I structure identification consists of finding the input candidates of the system and determining its actual variables which affect the output. In general, the selection of the input candidates is not a systematic process, i.e., one has to take a heuristic method based on experience and/or common sense knowledge for this purpose. The type II structure identification covers the determination of the number of rules and the partition of the (usually) multidimensional input space. The identification task is summarized in Table I. In this study, we discuss the latter three structure identification methods (type Ib, type IIa,b) and the parameter identification step.

Manuscript received March 11, 2001; revised January 16, 2002 and March 4, 2002. This work was supported by the Hungarian Scientific Research Fund (OTKA) under Grants D034614, T34212, and T34233, by the Chonbuk National University, and by the Australian Research Council. This work was completed in part while D. Tikk was visiting the School of Information Technology, Murdoch University, Murdoch 6150 W.A., Australia, and the Department of Computer Science, Chonbuk National University, Chonju 561-756, Korea.

D. Tikk and L. T. Kóczy are with the Department of Telecommunications and Telematics, Budapest University of Technology and Economics, H-1117 Budapest, Hungary (e-mail: tikk@ttt.bme.hu; koczy@ttt.bme.hu).

G. Biró is with the Department of Informatics, Eötvös Loránd Science University, H-1117 Budapest, Hungary, and also with the Department of Artificial Intelligence, Computer Automation Research Institute, Hungarian Academy of Sciences, H-1111 Budapest, Hungary (e-mail: george_biro@yahoo.com).

T. D. Gedeon is with the School of Information Technology, Murdoch University, Murdoch 6150 W.A., Australia (e-mail: tgedeon@murdoch.edu.au).

J. D. Yang is with the Department of Computer Science, Chonbuk National University, Chonju 561-756, Korea (e-mail: jdyang@cs.chonbuk.ac.kr).

Digital Object Identifier 10.1109/TFUZZ.2002.803494.

TABLE I
CLASSIFICATION OF THE IDENTIFICATION [1]

Structure identification I	a: input candidates b: input variables
Structure identification II	a: number of rules b: partition of the input space
Parameter identification	

The given data set is the following; $x_1,\ldots,x_n$ are the input variables and y is the output variable. N sample data are given in the form of $(x_1^k,x_2^k,\ldots,x_n^k) \rightarrow y^k$, or briefly $\mathbf{x}^k \rightarrow y^k$, $k=1,\ldots,N$.

A. Identification of Input Variables

The structure identification of type Ib concerns the selection of input variables that influence truly the output. This means that one has to choose a set of effective variables among a finite set of original variables. For this purpose, one needs a criterion function to evaluate the various candidate sets of variables. This function assigns a value to a given set of variables and its task is to minimize or maximize it. In [1], they used the RC [11], which was performed between steps identification of type II and parameter identification. The outcome of the RC method depends on the identification of type II (see also Fig. 1).

Let X be the set of all possible input candidates $x_1,\ldots,x_n$ and the total number of different input candidate sets is one less than the cardinality of the power set of X, that is 2^n-1. The RC is a heuristic method which selects a set of inputs among the possible candidates.

In the first step, the sample data set is divided into two groups, A and B. The criterion function is

$$\mathrm{RC}=\frac{\left[\sum_{i=1}^{|A|}\frac{\left(y_i^A-y_i^{AB}\right)^2}{|A|}+\sum_{i=1}^{|B|}\frac{\left(y_i^B-y_i^{BA}\right)^2}{|B|}\right]}{2} \tag{1}$$

where $|A|$ and $|B|$ denote the number of data in groups A and B, respectively, y_i^A and y_i^B are the outputs of groups A and B, respectively, and, finally, $y_i^{AB}(y_i^{BA})$ is the model output for the group $A(B)$ input estimated by the identified model using group $B(A)$ data.

For evaluating (1), two models should be built from the data groups A and B at each evaluation stage. According to [1], both structure identification of type II and parameter identification should be done before calculating RC. However, it seems to be more reasonable to omit parameter identification at this stage because it is quite time consuming.

Theoretically, all the possible 2^n-1 different input candidate sets should be evaluated, which is very costly in terms of calculation time, if n is not very small (>6). (Note that, theoretically, at each stage two complete fuzzy models have to be built for determining the value RC. However, this can be avoided with astute implementation. See Section III-C for details). Although this procedure is done offline, to reduce the necessary time a heuristic algorithm is used for determining the order of evaluation and the optimal set of variables.

The evaluation procedure is organized with respect to the cardinality of the candidate sets. First, the one-element candidate sets are evaluated with the corresponding RC value. Then those two-element candidate sets are evaluated which contains the best candidate of the previous level and so on. The candidate sets are structured in a tree, whose root (level 0) is the empty set and level ℓ contains nodes with candidate sets of cardinality ℓ. Two nodes are connected if the lower level one is obtained from the upper level one by adding one new variable to the candidate set. Those branches of the tree can be trimmed which have worse RC value than the best of the previous level, moreover, in this case the "bad" variable is removed permanently from the possible candidates. By means of this heuristic algorithm at last $n(n+1)/2$ nodes are evaluated.

We found that the efficiency and reliability of the RC method is low. The results are shown in Section III-C.

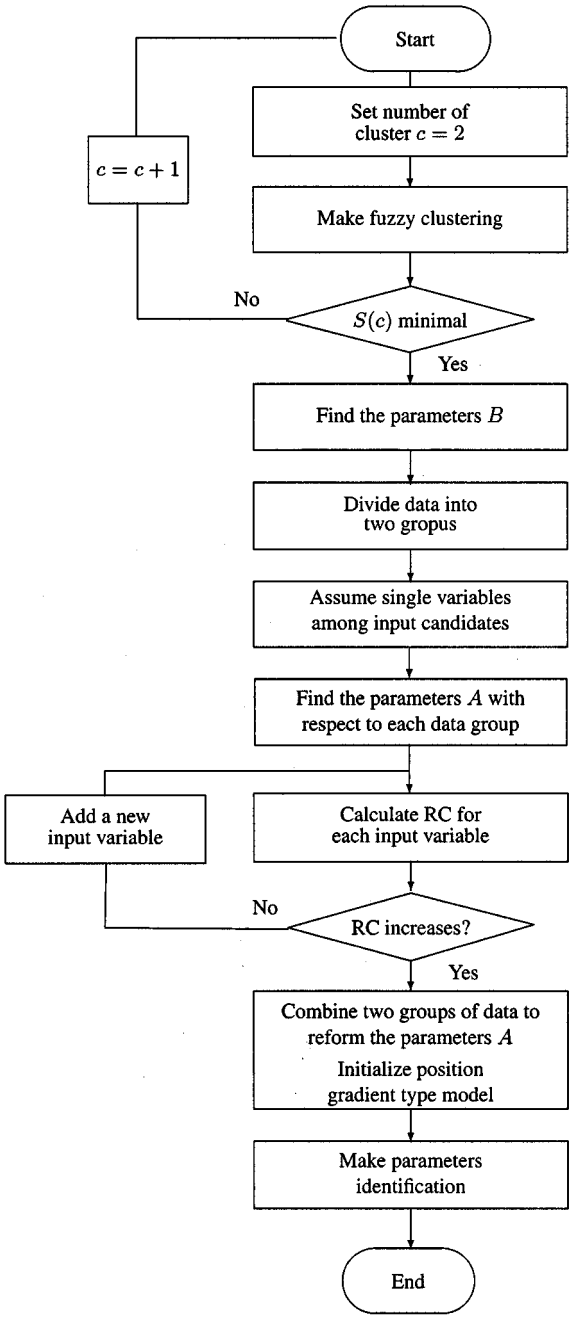

Fig. 1. Flowchart of the identification steps (redrawn from [1]).

B. Determination of the Number of Rules and the Input Partition

Usually in the design of a fuzzy system rule antecedents and the partition of the input domain are determined first. This (dense) rule base design methodology results in exponentiality in terms of the number of rules. To avoid this significant drawback, the SY method proceeds oppositely. First, the partition of the output space is determined, which is done by clustering the whole output data set by the fuzzy c-means (FCM) clustering [12]. Note that this means the clustering of the one-dimensional output values y^k, $k = 1, \ldots, N$. The optimal number of cluster are determined by means of the following criterion [13]:

$$S(C) = \sum_{k=1}^{N} \sum_{i=1}^{C} (\mu_{ik})^m \left(\left\| y^k - v_i \right\|^2 - \| v_i - \overline{y} \|^2 \right) \qquad (2)$$

where N is the number of data to be clustered, C is the number of clusters, $C \geq 2$, y^k is the output of the kth datum, $\overline{y}$ is the average of data y^k, v_i is the the centre of the ith cluster (here: scalar), μ_{ik} is the membership degree of the kth datum with respect to the ith cluster, and m is the fuzzy exponent $m > 1$. Note that membership grade μ_{ik} belongs to the whole kth datum, i.e., not only to the output y^k, but also to the input $\mathbf{x}^k$.

As a result of the clustering, every output datum is associated with a membership degree in all the clusters B_i, $i = 1, \ldots, C$. From an output fuzzy clusters B_i, we can induce a fuzzy cluster A_i in the multidimensional input space. This cluster can be projected onto the axes of the variables, hence, defining the antecedent fuzzy sets in each input dimension. Starting from a cluster B_i and assuming that we have two input variables x_1 and x_2, we usually obtain a rule like

$$\text{If } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ then } y \text{ is } B_i.$$

We remark that, although, this notation implies that the number of rules is identical with the number of output clusters, it can happen that this is not the case.

In the original paper, two procedures are not specified clearly. However, we found them crucial with respect to the performance of the model. On the one hand, in [1] the authors stated that, despite the input cluster B was convex, the corresponding input set A might not also be convex. Hence, it needed to be approximated. For simplicity, they proposed to approximate the (nonconvex) input clusters with trapezoidal membership functions. On the other hand, they remarked that more than one input cluster could belong to an output cluster. As a solution, they suggested to "form carefully two convex fuzzy clusters" in the input space.

Although, these details seem to be not very important from the methodology aspect of the SY fuzzy modeling, they affect, especially when using the RC method for input identification, significantly the performance of the built model. Our solutions are presented in Sections III-A and III-B.

C. Parameter Identification

The parameter identification step can be accomplished in two stages in the fuzzy model design. The authors in [1] proposed to repeat it in every input candidate evaluation step, but this is mostly superfluous and very time consuming. Performing it may be enough after the important input variables have been identified.

At this stage we have to measure the performance of the rough fuzzy model. For this purpose, the following performance index (PI) is used:

$$PI = \sum_{k=1}^{N} \frac{\left(y^k - \hat{y}^k\right)^2}{N} \qquad (3)$$

where $\hat{y}^k$ is the model output for the kth sample datum.

In the case of fuzzy model, the parameters are those of the membership functions. Having trapezoidal membership functions that means four parameters for each antecedent $p_1 \leq p_2 \leq p_3 \leq p_4$. In the parameter identification step, they adjusted these four values in an iterative algorithm. The algorithm works as follows.

1) Set an adjusting value f.
2) Pick the parameter to be adjusted, in general, the ℓth parameter of a trapezoid membership function in an arbitrary rule p_ℓ ($\ell = 1, 2, 3, 4$).
3) Calculate $p_\ell + f$ and $p_\ell - f$. For $\ell = 2, 3, 4$: if $p_\ell - f$ is smaller than $p_{\ell-1}$, then $\hat{p}_\ell = p_{\ell-1}$, otherwise $\hat{p}_\ell = p_\ell - f$. Analogously, for $\ell = 1, 2, 3$: if $p_\ell + f$ is bigger than $p_{\ell+1}$, then $\tilde{p}_\ell = p_{\ell+1}$, otherwise $\tilde{p}_\ell = p_\ell + f$.
4) Choose the parameter among $\{p_\ell, \hat{p}_\ell, \tilde{p}_\ell\}$ which produces the best performance according to (3) and replace p_ℓ with it.
5) Go to step 2) while unadjusted parameters exist.
6) Repeat the iteration until we are satisfied with the result.

In [1], 5% of the width of the actual input space is used as the adjusting value.

The flowchart on Fig. 1 shows the overall design of the identification steps of the SY modeling.

III. Implementation Details of the SY Method

A. Trapezoid Approximation

The trapezoid approximation of the clustered raw data is done in two steps. First, the convex hull of the original data set is determined, then the convex hull is approximated by a trapezoidal membership function. Fig. 2 depicts the idea of the construction of trapezoidal membership function.

We propose a simple and fast trapezoid approximation algorithm in three different versions. The three versions differ in the determined support length, or in other words, in the angle of the slopes of the trapezoid. However, in ordinary situations the three versions generate identical or almost identical results. The difference is significant when the distribution of the data with high membership grade is large near the minimum/maximum of the cluster. In such cases, the first version results in close-to-oblong shape trapezoids with steep slopes, the third in trapezoids with long and smooth slopes, while the second version generates an average solution of the former two. For brevity, these versions are termed steep-slope, smooth-slope, and average-slope.

The common base algorithm is the following.

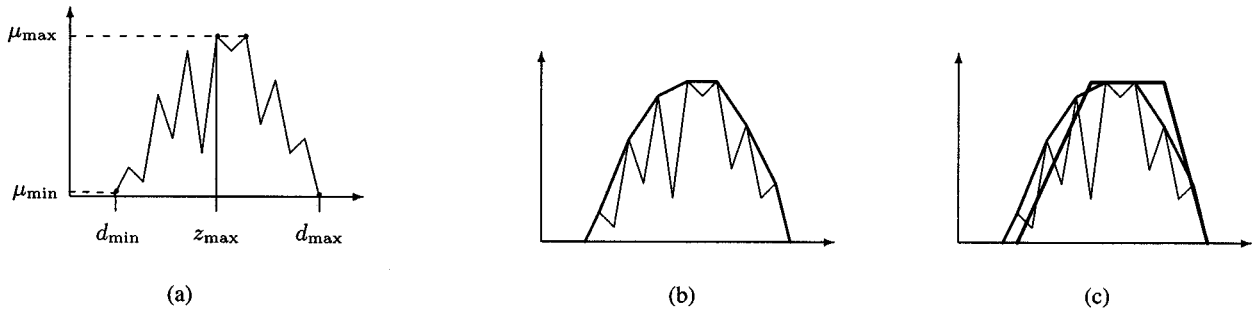

Fig. 2. The two steps of trapezoidal membership function construction. (a) C-means clustered raw data. (b) The convex hull of the input cluster. (c) Approximated trapezoidal fuzzy set (reproduced from [1]). We inserted the notation of our trapezoid approximation algorithm on (a).

Trapezoid Approximation

1. Determine μ_{min} and $\mu_{\max}$ the minimum/maximum of the membership degree of the data point in the given cluster, $z_{\max}$ the first point where the maximum $\mu_{\max}$ is attained, further $d_{\min}$ and $d_{\max}$ the minimum/maximum of all the data values in the given cluster domain (see also Fig. 2). Here z denotes the sample data points of the dimension in which the trapezoid approximation of membership function is performed; it can be either the output (y^k) or any of the input dimensions (x_j^k), $k = 1,\ldots,N$; $j = 1,\ldots,n$.
2. Set the boundaries of investigated interval to

$$m = \mu_{\min}\frac{q-1}{q} + \frac{\mu_{\max}}{q}$$
$$M = \mu_{\max}\frac{q-1}{q} + \frac{\mu_{\min}}{q} \qquad (4)$$

where $q > 1$ is a given parameter, usually between 2 and 4.
3. Determine the parameters of the left slope, p_1 and p_2, as:
(a) Let us initialize z^i as the last data point which has smaller membership degree than m, $i = \arg\max_{k\in[1,N]}\{z^k < m\}$ and z^j be the next point of the convex hull, i.e.,:

$$\mu(z^i) < m \quad \mu(z^j) > m, \qquad 1 \le i < j = i+1 \le N.$$

(If there is no such point z in the convex hull which satisfies $\mu(z) < m$ then z_i and z_j are the first two leftmost points of the convex hull). Further the parameters of the left slope $p_1 = d_{\min}$ and $p_2 = d_{\min}$, i.e., the starting membership function is a crisp interval.
(b)Let p_1^* and p_2^* be the location of the intersection made by the support and the core, respectively, with the line passing through the points $(z^i, \mu(z^i))$ and $(z^j, \mu(z^j))$ (see Fig. 3).
(c) If $p_1^* > p_1$ then $p_1 := p_1^*$.
(d)If $p_2^* < p_2$ or $j = i+1$ then $p_2 := p_2^*$.
(e)If $z^{j+1} \le z_{\max}$ then let $j := j+1$ and go to step (3b), otherwise continue.
4. Determine the parameters of the right slope, p_3 and p_4, analogously as in the previous step.
5. Order the parameters according to $p_1 \le p_2 \le p_3 \le p_4$

In step 2., the reason for narrowing the range is twofold. From below, it is important to exclude the data points with very low membership grades. Notice that the clustering algorithm assigns (almost) every datum a positive membership grade in each cluster, however, usually a datum has significant membership degree in at most two clusters. From above, it is also reasonable to disregard points with high membership grades, because, on one hand, they most probably belong to the core of the membership function and they do not play role in the determination of the slopes; on the other hand, the technique used for the determination of the slope is sensitive to high membership grades, as it is described in the next paragraph.

The difference between the three versions appears in step 3(c). Observe that in such a situation where the two leftmost points have high membership grades, then the left slope is very fuzzy. Therefore, in this case the minimum of the support is the default value: $d_{\min}$ (see Fig. 4). If this phenomenon is present at both ends of the support of a trapezoid, then the final membership function is positive on the whole domain. However, it is unlikely that the original cluster solely dominates the whole dimension.

To alleviate this drawback we propose two possible solutions: a steep-slope version and an average-slope version. The steep-slope version generates a short support by giving z^i as the left end of the support, while the average-slope version determines the arithmetic mean of $d_{\min}$ and z^i as the leftmost point (see Fig. 4).

In order to provide a simple schema, we did not include exception checking and handling in the aforementioned algorithm. Nevertheless, these are important parts of the implementation process.

We remark that the SY method uses parameter identification for fine tuning of the rules, when parameters of the membership

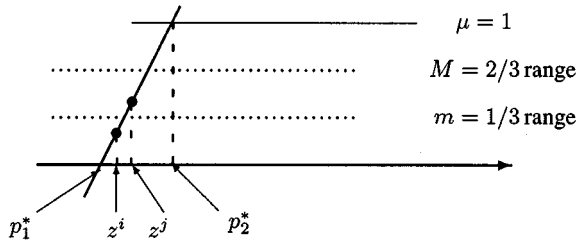

Fig. 3. Determination of the location p_1^* and p_2^*. Here $q = 3$.

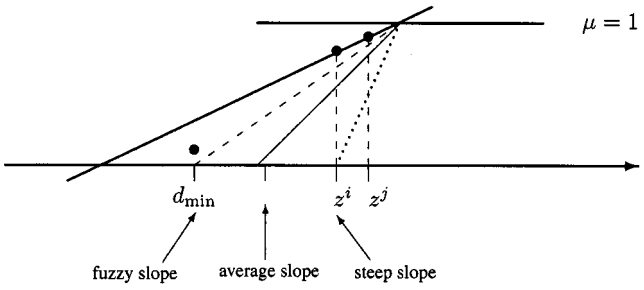

Fig. 4. Too smooth left slope (thick line). The default smooth-slope version then takes $d_{\min}$ as p_1 (dash line). The steep-slope version takes z^i as p_1 (dotted line), while the average-slope version takes $(d_{\min} + z^i)/2$ (thin line).

functions are shifted. Hence, starting from a wide membership function with smooth slopes one can end up with a much narrow one with steep slope, if the testing data set sustains this modification. From this point of view it is better to start with a wide membership function, which may make more flexible the parameter identification procedure. We shall return to this issue in Section III-D.

B. Determining the Number of the Rules

In this section, we present an algorithm which determines the number of rules belonging to one output cluster by defining the input clusters in the multidimensional input space. This is an important detail of the original algorithm in our view, which was not given the required attention in [1]. This question has significant effect on the total number of rules and, thus, on the computational complexity of the final model. The method plays an important role, especially when the modeled function is not strictly monotone, unlike the examples of the original paper (see, e.g., Fig. 5), but the same output is assigned to several regions of the input space.

Algorithm for Determining the Antecedents

Inputs: training data set, membership degree vector of the fuzzy c-means clustered training data set,
output: the actual rules.

1. Select a cluster i, where $i = 1,\ldots,C$ and divide the training data set into two groups: group A consists of data with membership degree not smaller than 0.5 in cluster i, while group B includes the remaining data set:

$$A = \{\mathbf{x}^k \,|\, \mu_{ik} \geq 0.5, \quad k = 1,\ldots,N\}$$
$$B = \{\mathbf{x}^k \,|\, \mu_{ik} < 0.5, \quad k = 1,\ldots,N\}$$

2. Create a rule base R of $2\cdot|A|$ different rules, by forming separate rules from each of the group A data and the first $|A|$ group B data. If $|A| > |B|$ then we form N rules. At this point the rules are crisp in the sense that the antecedents as well as the consequent are crisp values.
3. Evaluate the performance of this rule base by checking this model on the group B data set. We determine the rule base performance by counting the data with incorrectly high membership grades as

$$p_R = \frac{|B'|}{|B|}$$

where

$$B' = \{\mathbf{x}^k \in B \,|\, W_R(\mathbf{x}^k) \geq t_1, k = 1,\ldots,N\}.$$

Here function W ($W : \mathbb{R}^n \to [0,1]$) determines the firing weight of the datum $\mathbf{x}^k$ by the rule base R and the t_1 is a given threshold, usually 0.5.
4. Temporarily merge two consecutive rules by using the "and" operation for the corresponding antecedents and the consequents. Denote the obtained rule base by R'. Calculate $p_{R'}$. If

$$p_{R'} > p_R + t_2$$

then the performance of the new rule base is significantly worse, therefore undo the fusion of the two rules; otherwise merge them permanently: $R := R'$. Here t_2 is another threshold parameter, usually ≤ 0.5.
5. If all the rule pairs are checked then stop, otherwise check the next pair and go to step 4.

This algorithm proceeds for all the output clusters consecutively. The group A data set serves as the basis of the rule forming procedure. The group B data set serves as the local training data set and it also balances the impact of group A data in rule base formation. It is advantageous to use B as the training set, because its cardinality is in general less than $N/2$, if the number of clusters exceeds two.

When we merge two rules by the "and" operation, this includes the creation of trapezoidal membership functions on the basis of the added data.

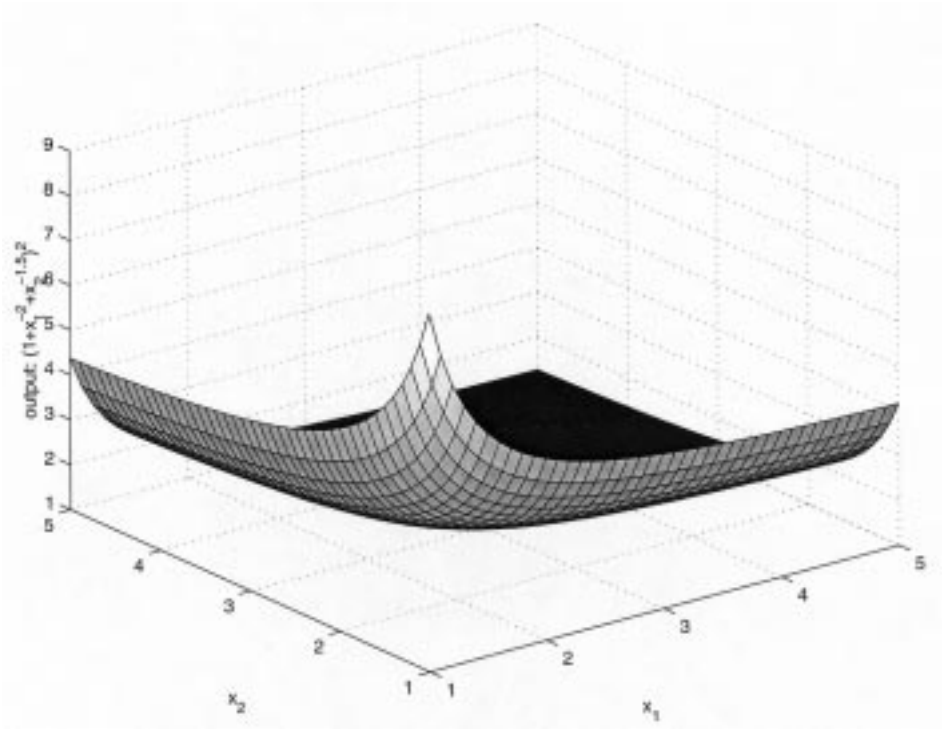

Fig. 5. Input–output relation of the function $y = (1 + x_1^{-2} + x_2^{-1.5})^2$ used in [1].

When we end up with only one rule we have to go through step 4, $2|A|-1$ times, which is proportional with $O(N)$ for each cluster. In general, step 4 is executed in worst case

$$2r|A| - \frac{r(r-1)}{2} < N^2 - \frac{N(N-1)}{2} < \frac{N^2}{2} = O\left(N^2\right) \quad (5)$$

where r is the number of rules generated from one cluster. For details, see the Appendix A. We would like to emphasize that this is not the total complexity of the algorithm but the number of iterations of step 4. The asymptotic number of operation in step 4 depends on the implementation of W_R. We should point out that the calculation of W_R involves the firing of the rule bases that necessitates normally exponential time for dense rule bases [9]. Due to the construction of the rule base, however, in this case the time is proportional with $n|R|$, i.e., the time is polynomial.

The role of the thresholds t_1 and t_2 is to control the number of rules. The lower the thresholds are the more rules are generated. From computational complexity reasons, it is not useful to choose a very low threshold, because it may increase the number of rules drastically. Beside this, by setting the thresholds t_1 and t_2 low, the obtained model relies very much on the training set, rules tend to be more crisp than fuzzy and the generalization capability of the model declines. The reasonable choice for t_1 and t_2 is $0.1 \sim 0.5$.

C. The Reliability of the RC Method

We investigated how the changes in the parameters of the SY method (such as trapezoid approximation version and parameters, division of data into two groups in the starting step of the RC method) affect the result of the input selection performance of RC. We found that the RC method is very sensitive to the above parameters, that is a slight change in those values may result in a different identified input variable set. Of course this issue is highly problem dependent. So, for more robust training data sets the RC shows more stable performance.

We illustrate the instability of the RC on data sets taken from [1]. The first set contains input–output data of a chemical plant. The system has five inputs which are the following: x_1—monomer concentration, x_2—change of monomer concentration, x_3—monomer flow rate, x_4, and x_5—local temperatures inside the plant. The output is the set point for monomer flow rate. There are 70 sample data provided. The number of output clusters is 6.

The original paper found the variables 3, 2, and 1 to be the true inputs. Here, the order of the variables refers to the ranking of importance. We have to admit that we could not produce this result with our implementation regardless of the configuration applied. This was a challenge for us and partly the reason for our further investigations in order to find the reason of this ill-success.

The second data set contains stock price data. The goal is the prediction of the stock price based on ten input variables and 100 samples. For more details, consult the paper of Sugeno and Yasukawa.

1) Sensitivity to Data Set Division: We investigated the result of the RC method with different ordering and different division of the input data set. We applied three different grouping and order techniques to the chemical plant data set.

1) We ordered the training data set according to the output value of each data set. We composed the two groups by putting the data items alternately into the two groups.
2) The ordering is the same, but one group was formed of the data points with low output values, while the other was the high ones.

3) We ordered the training data set according to the Euclidean norm of data calculated as

$$\left\|\left(\mathbf{x}^k, y^k\right)\right\| = \left(\sum_{j=1}^{n}\left(x_j^k\right)^2 + \left(y^k\right)^2\right)^{1/2}.$$

The two groups were constructed alternately from the ordered data.

The RC gave identical results with the first and third data division (true input: 3), which was different from the one with the second division (true inputs: 1, 3). Note that in [1], the authors obtained true inputs 3, 2, and 1. We may conclude that the second division is not an appropriate choice because the data in the two groups are unbalanced and hence the cross-identification of the models is not well justifiable due to their substantial diversity. However, this example supports the judgment of the RC method indicating the importance and the impact on the results of every minor detail.

2) Sensitivity to the Trapezoid Approximation: Here, we investigate the effect of the application of the different slope versions described in Section III-A as well as the usage of different range parameters.

First, we compared the results of RC combined with different trapezoid approximation versions on the stock price prediction model. We used the three trapezoid approximation versions presented earlier in the paper. Table II shows the obtained true input variable sets.

The results are different for all versions! The first two true input sets are quite similar. The only difference is that the ninth input is substituted by the pair 4, 5. However, the steep-slope version coincides only at one place with each of the two other versions and there is no common true input for all the three versions. This might be argued by the small size of the data set, nevertheless, this example does not strengthen the applicability of the RC method. With the more robust chemical plant data set, we obtained the same result by all the different slope versions (true input: 3).

Second, we checked the RCs stability against the range parameter q [see (4)] of the trapezoid approximation on both data sets. The results are summarized in Table III.

It can be seen that the first data set is more robust also in this comparison. It gives different results only for the marginal $q = 1$ value, i.e., when all data points are considered in the trapezoid approximation. With diverse range values the RC method assigns varied true input sets to the stock price data set. The only accordance among the obtained sets appears for values 2.0 and 4.0. These results were generated by means of the smooth-slope version trapezoid approximation.

3) Final Remark on the Application of the RC Method: As it was emphasized in this section, the RC method lacks stability due to the fact that it relies strongly on the different implementation parameters. In our view, an input identification method should depend only on the data set and not on other implementation details.

This issue has another important point: the computational complexity. Theoretically, the RC method needs the construction of the whole model two times for every input candidate set evaluation (see Fig. 1). In practice, once we have built up the model for all the input candidates, we can omit certain inputs from this according to the actual input candidate set. Thus, the recalculation of antecedents becomes superfluous, which is, in fact, as we have remarked earlier, the most lengthy calculation in the model construction cycle.

TABLE II
THE TRUE INPUTS OF THE STOCK PRICE PREDICTION MODEL WITH THE USE OF RC METHOD AND DIFFERENT TRAPEZOID APPROXIMATION METHOD

Version	True input variables
smooth-slope	{2, 3, 8, 9}
average-slope	{2, 3, 4, 5, 8}
steep-slope	{4, 9}

TABLE III
THE EFFECT OF TRAPEZOID RANGE MODIFICATION ON RC METHOD

q	Chemical plant data set True input variables	Stock price data set True input variables
3.0	{3}	{2, 3, 8, 9}
1.0	{1, 3}	{2, 8}
2.0	{3}	{2, 6, 8}
4.0	{3}	{2, 6, 8}
10.0	{3}	{3, 5}

In order to fix this problem, we proposed in [14] and [15] a feature ranking method that works on fuzzy clustered output (FRFCO method). In the literature on pattern recognition, one can find numerous feature ranking and selection techniques for classification problems, but the FRFCO method offers a way to deal with systems with continuous output while maintaining the transparency and linguistic interpretability of the rule base. Based on the ranking, which is determined independently from model construction, the user can decide how many features s/he wants to take into consideration, or alternatively, one can compare the performance of the identified models using the best one, two, three, etc. features from the ranking. For more details, consult [14] and [15]. Fig. 6 shows how to modify the SY modeling in such a case.

D. The Parameter Identification Procedure

We propose a modification of the original parameter identification procedure, which works with not a fix, but varying adjusting value f depending on the actual performance value. We set the starting adjusting value in the jth input as

$$f = \frac{\mathrm{dom}(j)}{4p_{\mathrm{steps}}} \cdot 2^{p_{\mathrm{start}}+c}$$

where $\mathrm{dom}(j) = d_{\max}(j) - d_{\min}(j)$ is the domain of the jth input, i.e., the difference between the smallest and the largest input in the given dimension, $j = 1, \ldots, n$; p_{steps} is a predefined constant (default: 3); p_{start} is to set the starting precision (default: -1) and c is an iteration counter which increases if

$$PI_{\mathrm{speed}} = \frac{PI_{\mathrm{last}} - PI_{\mathrm{actual}}}{PI_{\mathrm{last}}} < 0.1 \qquad (6)$$

that is, if the amelioration of the performance index is less than 10%. The starting value of c is zero. The parameter identification is organized in a double loop. In the inner loop, the four parameters of the trapezoid membership function of all the antecedents are sequentially adjusted with the same actual adjusting value until no further improvement can be achieved or the number of inner iterations attain a certain

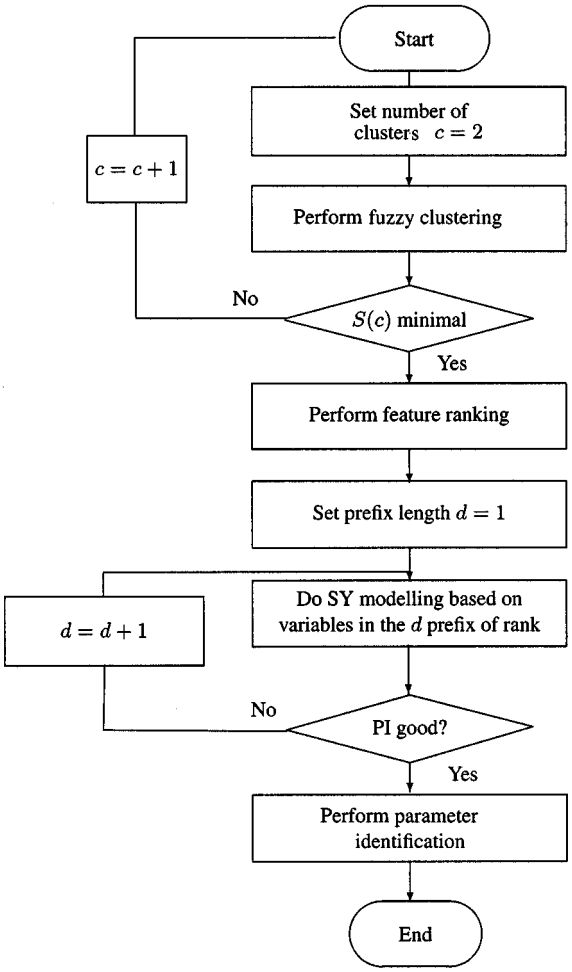

Fig. 6. Modified SY modeling after the incorporation of any (statistical) feature ranking method.

limit. Then, c is increased if (6) holds and the whole process restarts. The stopping criterion of the outer loop can be either a certain time limit or when the PI_{speed} gets smaller than a certain threshold.

We used the order p_1, p_4, p_2, p_3 for adjusting the parameters, i.e., first the support's and then the core's parameters are modified. This may affect the final performance index; the wider the starting support of an antecedent, the more space is available for finding the appropriate core length. As an experimental observation, we noticed that the support's length is inclinable to shrink and vice-versa the core's length is rather subject to widening. Therefore, in the case of steep-slope trapezoid approximation, there is not much possibility of widening the core, while this is the opposite in the case of the smooth-slope version. This may be the reason for the results of our comparative investigation on the trapezoid approximation versions versus the performance index. The results are summarized in Tables IV and V, where PI_1 and PI_2 denotes the performance index before and after parameter identification.

We can state that the smoother the starting membership function the bigger the achieved improvement can be. However, it does not mean automatically that the smooth-slope version provides the best result. For example, in the chemical plant case the average-slope version produces the best result both before and after parameter identification. Based on these two examples the best choice seems to be the average-slope version, because it gives good results according to PI, before and after parameter identification. (We should remark that for Table IV results we forced the $\{1, 2\}$ true input set due to the fact that the RC determined different inputs for the various trapezoid versions).

TABLE IV
PERFORMANCE INDEXES WITH THE USE OF DIFFERENT TRAPEZOID APPROXIMATION VERSIONS ON SYNTHETICAL SAMPLE DATA SET ($y = (1 + x_1^{-2} + x_2^{-1.5})^2$)

	Smooth	Average	Steep
PI_1	0.5580	0.4110	0.3870
PI_2	0.0322	0.0355	0.0497
Improvement	17.33	11.57	7.79

TABLE V
PERFORMANCE INDEXES WITH THE USE OF DIFFERENT TRAPEZOID APPROXIMATION VERSIONS ON CHEMICAL PLANT SAMPLE DATA SET

	Smooth	Average	Steep
PI_1	$4.64 \cdot 10^5$	$7.58 \cdot 10^4$	$1.03 \cdot 10^5$
PI_2	$2.60 \cdot 10^4$	$1.06 \cdot 10^4$	$7.15 \cdot 10^4$
Improvement	17.85	7.15	1.44

IV. AN IMPROVEMENT ON RULE BASE REDUCTION

Our proposed improvement on rule base reduction can be applied effectively if there exist output clusters to which more than one identified input cluster belongs. The idea of our improvement that in such a case it is possible to merge two or more rules under certain conditions. The merged rules are of a special type to be described later.

Observe the simplest case of a possible rule merging depicted in Fig. 7. Here, the output cluster B generates two two-dimensional input clusters C_1 and C_2 and their projections coincide in one of the dimensions, while they differ in the other one. Therefore, the generated rules are

$$\begin{aligned} &\text{If } A_{11} \text{ and } A_2 \rightarrow B \\ &\text{If } A_{21} \text{ and } A_2 \rightarrow B. \end{aligned} \tag{7}$$

Here, we assume that A_{11} and A_{12} are disjoint. These rules can be merged by using special "and/or" rule type as

$$\text{If } (A_{11} \text{ or } A_{21}) \text{ and } A_2 \rightarrow B. \tag{8}$$

The introduced rule merging reduces the calculation time because it requires less time to check whether the actual observation (A^*) has positive intersection with the rule or not, simply because we do not need to twice check the intersection of A_2^* with A_2. Moreover, depending on the order of dimensions in the rule evaluation, the speed of rule checking can be further enhanced. Namely, if we first check the intersection of A_2^* with A_2 and it is zero, then it is needless to check A_1^*. Oppositely, if $A_2^* \cap A_2 \neq \emptyset$, then we have also to check the first dimension. Even in the simplest exemplified case one rule merging from (7) to (8) can result in a few percent (5%–10%) of time improvement depending on the total number of rules (usually around six).

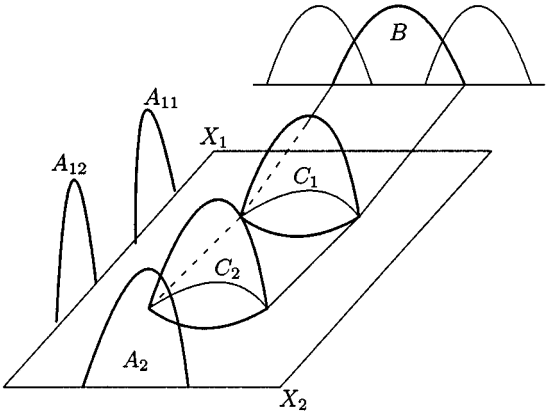

Fig. 7. The simplest case of rule merging.

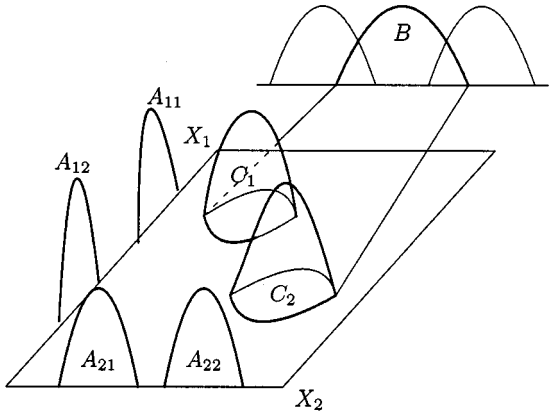

Fig. 8. Counterexample: The merging cannot be done if the input clusters differs in more than one dimensions.

It is important that the mergeable rules should be different in exactly one dimension. Otherwise, if they differed in, e.g., two dimensions, the merged rules would cover four regions of the input space as it is depicted in Fig. 8.

The previously described rule merging can be generalized in the following way. Having m rules of the form

$$\begin{aligned} &\text{If } A_{11} \text{ and } A_2 \text{ and } \ldots \text{ and } A_n \rightarrow B \\ &\text{If } A_{21} \text{ and } A_2 \text{ and } \ldots \text{ and } A_n \rightarrow B \\ &\qquad\qquad\qquad\qquad\vdots \\ &\text{If } A_{m1} \text{ and } A_2 \text{ and } \ldots \text{ and } A_n \rightarrow B. \end{aligned} \tag{9}$$

we can merge them to

$$\text{If } (A_{11} \text{ or } A_{21} \text{ or } \ldots \text{ or } A_{m1}) \text{ and } A_2 \ldots \text{ and } A_n \rightarrow B. \tag{10}$$

where fuzzy sets A_{i1} $(i = 1, \ldots, m)$ are disjoint. The multiple rule merging is quite improbable in practice; the bigger the number of the rules and the dimension of the input, the smaller the chance to find mergeable rules in a rule base. On the other hand, however, if the surface of the input–output function of the modeled system is not strictly monotone, there is a good chance to find mergeable rules. We have to remark that this procedure is done offline, i.e., not in real time and it is a very simple search on the rules with the same output fuzzy set.

In real applications, we cannot expect that two fuzzy sets coincide precisely. Thus, it is more reasonable to investigate the similarity of the candidate fuzzy sets. There are several ways to determine the similarity of two fuzzy sets (see, e.g., [16]–[18]), but we choose a very simple technique. We consider two fuzzy sets similar if their cores overlap. In this case, we create a larger fuzzy set which includes both fuzzy sets, more precisely their common convex hull. Formally, if $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4)$ are trapezoidal fuzzy sets and $[a_2, a_3] \cap [b_2, b_3] \neq \emptyset$ is satisfied, then their convex hull that will substitute A and B in the merged rules, is defined as $C = A \oplus B = (c_1, c_2, c_3, c_4)$ (see also Fig. 9), where

$$c_i = \begin{cases} \min(a_i, b_i), & \text{if } i = 1, 2 \\ \max(a_i, b_i), & \text{if } i = 3, 4 \end{cases}.$$

To illustrate the capability of this improvement of rule base reduction we analyzed the function

$$y = \left(1 + \sin^2(x_1) + x_2^2\right)^2, \qquad (x_1, x_2) \in [-2, 2] \times [-2, 2]. \tag{11}$$

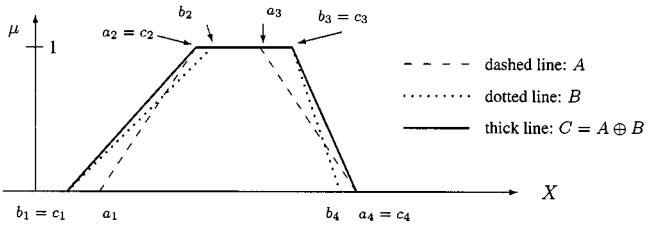

Fig. 9. The construction of $A \oplus B$.

The function is depicted on Fig. 10. We randomly selected 100 sample points.

We investigated the effectiveness of rule reduction with various parameter settings. The values of parameters t_1 and t_2 were varied to verify our algorithm, because they influence the number of the rules most significantly. The lower the values are set the more rules are generated based upon a given output cluster (see Section III-B). The number of output clusters was six, independently from the settings. The RC method found both inputs important. We obtained the following results, which were summarized in Table VI.

High parameter values (the default case): parameters t_1 and t_2 were set to the default value 0.5. In this case, the rule base consisted of seven rules before reduction. After the rule base reduction, the resulting rule base had the minimal six rules.

Medium parameter values: $t_1 = 0.3$, $t_2 = 0.3$. The number of rules was eight before reduction and six after reduction.

Low parameter values: $t_1 = 0.1$, $t_2 = 0.1$. The number of rules was 13 before reduction and nine after reduction.

Lowest parameter values: $t_1 = 0.01$, $t_2 = 0.1$. The number of rules was 14 before reduction and nine after reduction.

Table VI shows that the best performance index was achieved at the setting $t_1 = t_2 = 0.1$ and the worth at medium parameter values.

The time reduction depends heavily on the actual data samples and the order of the variables. In this example, we got much greater time reduction if we use variable order 2, 1 at the evaluation of each rules. This phenomenon can be explained by the observation that the number of "or" rules containing merged antecedents in the dimension two is more than in dimension one (e.g., at low parameter values these are three vs. one).

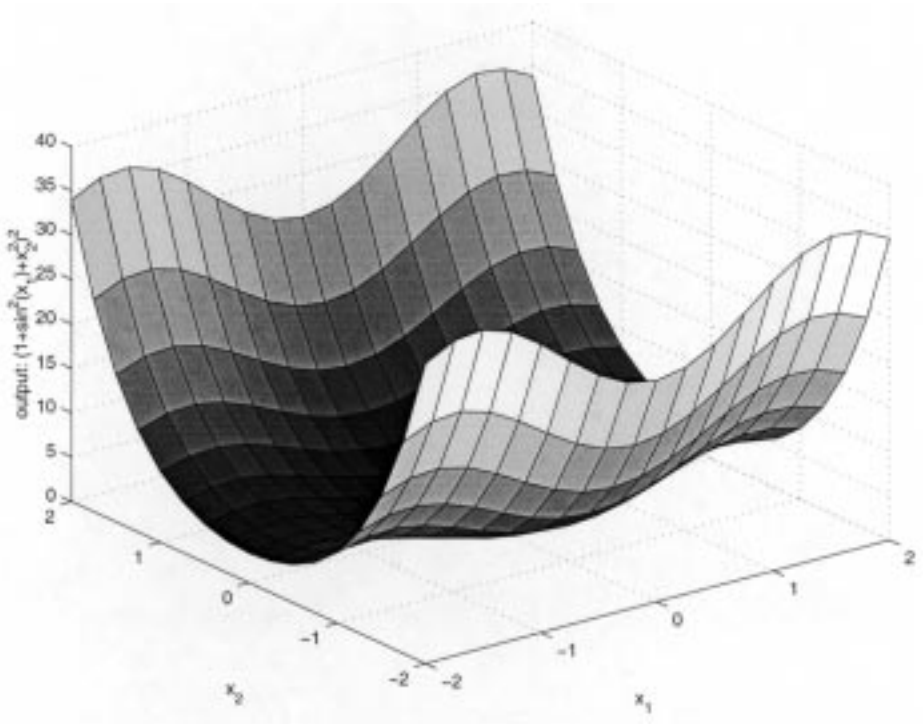

Fig. 10. Input–output relation of the function $y = (1 + \sin^2(x_1) + x_2^2)^2$. The difference between two consecutive grid points is 0.25.

TABLE VI
THE EFFECTIVENESS OF RULE BASE REDUCTION IN TERMS OF t_1 AND t_2 PARAMETER VALUES

t_1/t_2 (if differ)	0.5	0.3	0.1	0.01/0.1
Number of rules before reduction	7	8	13	14
Number of rules after reduction	6	6	9	9
PI before/after reduction (if differ)	15.2/15.5	19.6	13.6	15.2
Time reduction	4%	6%	14%	16%

V. CONCLUSION

In this paper, we clarified some vague unexplicit details of the Sugeno and Yasukawa's qualitative fuzzy modeling method. We proposed algorithms for trapezoid approximation, for the determination of the number of rules (belonging to one output cluster) and parameter identification. We investigated the reliability and stability of the RC method that is applied to determine the true inputs among input candidate variables and we found that the results were not satisfactory. Therefore, we proposed an alternative technique for ranking the input variables in [14] and [15].

Furthermore, we introduced an improvement on rule base reduction, namely, a kind of rule merging that is employable if two (or more) rules are different in exactly one input dimension. Although the applicability of this reduction is limited, in such cases where several rules can be merged, i.e., where the control surface not strictly monotone, it can reduce the elapsed time significantly.

APPENDIX

A. Proposition 1

Step 4 in the "Algorithm for determining the antecedents" is executed at most $2r|A| - r(r-1)/2$ times and at least $2|A| - r - 3 + r(r-1)/2$ times for each output cluster. Here, r denotes the number of rules generated from one output cluster.

Proof: The determination of the antecedents for an output cluster is a sequence of decisions: whether the two actual rules can be merged or should be separated. Let k_i be the number of successful rule fusions between the $i-1$th and ith rule separation. Hence, we have

$$k_1 + 2k_2 + 3k_3 + \cdots + r\,(2|A| - (k_1 + k_2 + \cdots + k_{r-1} - 1)) \tag{12}$$

where

$$1 \le k_i \le N - \sum_{j=1}^{i-1} k_j \text{ and } \sum_{i=1}^{r-1} k_i < 2|A| - 1.$$

It is obvious that the later a rule separation is executed, the less conditional rule merging (or rule comparison) is needed. Therefore, in worst case $k_i = 1$ for all $i = 1, \ldots, r-1$, and, from (12), we obtain

$$2r|A| - r - (r-1)k_1 - (r-2)k_2 - \cdots - k_{r-1} = 2r|A| - \frac{r(r-1)}{2}.$$

In the best case when the rules are separated as late as possible, then $k_1 = 2|A| - r - 2$ and $k_i = 1$ for all $i = 2, \ldots, r-1$, so expression (12) is equal to

$$2|A| - r - 2 + 2 + 3 + \cdots + r = 2|A| - r - 3 + \frac{r(r-1)}{2}.$$

If we estimate from above $2|A|$ and r by N then the maximum number of executaion of Step 4

$$N^2 - \frac{N(N-1)}{2} = \frac{N(N+1)}{2} = O(N^2).$$

■

REFERENCES

[1] M. Sugeno and T. Yasukawa, "A fuzzy logic based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Jan. 1993.

[2] L. T. Kóczy and K. Hirota, "Rule interpolation in approximate reasoning based fuzzy control," in *Proc. of 4th IFSA World Congr.*, R. Lowen and M. Roubens, Eds., Brussels, Belgium, 1991, pp. 89–92.

[3] ——, "Approximate reasoning by linear rule interpolation and general approximation," *Int. J. Approx. Reason.*, vol. 9, pp. 197–225, 1993.

[4] P. Baranyi, "Fuzzy Information Reduction Techniques in Control Algorithms," Ph.D. (in Hungarian), Tech. Univ. Budapest, Budapest, Hungary, 1999.

[5] D. Tikk and P. Baranyi, "Comprehensive analysis of a new fuzzy rule interpolation method," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 281–296, June 2000.

[6] V. Cross and T. Sudkamp, "Geometric compatibility modification," *Fuzzy Sets Syst.*, vol. 84, no. 3, pp. 283–299, 1996.

[7] D. Dubois and H. Prade, "Gradual rules in approximate reasoning," *Inform. Sci.*, vol. 61, pp. 103–122, 1992.

[8] I. B. Turkşen and Z. Zhong, "An approximate analogical reasoning approach based on similarity measures," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 1049–1056, Dec. 1988.

[9] L. T. Kóczy, "Computational complexity of various fuzzy inference algorithms," *Ann. Univ. Sci. Budapest, Sect. Comp.*, vol. 12, pp. 151–158, 1991.

[10] L. T. Kóczy and K. Hirota, "Size reduction by interpolation in fuzzy rule bases," *IEEE Trans. Syst., Man, Cybern.*, vol. 27, pp. 14–25, Feb. 1997.

[11] J. Ihara, "Group method of data handling toward a modeling of complex system IV" (in Japanese), *Syst. Control*, vol. 24, pp. 158–168, 1980.

[12] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[13] Y. Fukuyama and M. Sugeno, "A new method of choosing the number of clusters for fuzzy c-means method" (in Japanese), in *Proc. 5th Fuzzy System Symp.*, 1989, pp. 247–250.

[14] D. Tikk and T. D. Gedeon, "Feature ranking based on interclass separability for fuzzy control application," in *Proc. Int. Conf. Artificial Intelligence in Science and Technology (AISAT'2000)*, V. Karri and M. Negnevitsky, Eds., Hobart, Tasmania, Australia, 2000, pp. 29–32.

[15] D. Tikk, T. D. Gedeon, and K. W. Wong, "A feature ranking algorithm for fuzzy modeling problems," in Trade-Off Between Accuracy and Interpretability in Fuzzy Rule-Based Modeling, ser. Series of Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Physica-Verlag.

[16] S.-M. Chen, M.-S. Yeh, and P.-Y. Hsiao, "A comparison of similarity measures of fuzzy values," *Fuzzy Sets Syst.*, vol. 72, no. 1, pp. 79–89, 1995.

[17] C. P. Pappis and N. I. Karacapidilis, "A comparative assessment of measures of similarity of fuzzy values," *Fuzzy Sets Syst.*, vol. 56, no. 2, pp. 171–174, 1993.

[18] R. Zwick, E. Carlstein, and D. Budescu, "Measures of similarity among fuzzy concepts: A comparative analysis," *Int. J. Approx. Reason.*, vol. 1, pp. 221–242, 1987.

Domonkos Tikk was born in Hungary in 1970. He received the B.Sc. and M.Sc. degrees in computer science from Eötvös Loránd Science University, Budapest, Hungary, and the Ph.D. degree from the Budapest University of Technology and Economics, Hungary, in 1993, 1995, and 2000, respectively.

Since 1998, he has been Research Assistant at the Budapest University of Technology and Economics, Hungary. Since 2000, he has been a Postdoctoral Research Fellow at the same institute. He was Visiting Scholar at Murdoch University, Perth, Australia, in 2000, and at Chonbuk National University, Chonju, Korea, in 2001. His research interests include fuzzy and neural network techniques, text mining, and clustering/classification methods.

Dr. Tikk is a Member of Hungarian Society of the International Fuzzy System Association (IFSA).

György Biró was born in Hungary in 1973. He received the M.Sc. degree in electrical engineering from the Technical University of Budapest, Hungary, in 1999. He is currently working toward the Ph.D. degree at the Eötvös Loránd Science University, Budapest, Hungary.

Since 1998, he has been Research Assistant at the Computer Automation Research Institute, Budapest, Hungary. His research interests include data mining with fuzzy and neural network techniques.

Tamás D. Gedeon received the B.Sc. (Hons.) and Ph.D. degrees from The University of Western Australia, Crawley, in 1981 and 1989, respectively, and the Graduate Diploma in management from the Australian Graduate School of Management, Kensington, Australia, in 1994.

He currently holds the position of Chair in Information Technology and is Head of the School of Information Technology and Director of the Interactive Media Institute at Murdoch University, Perth, Australia. His research is focused on the development of automated systems for information extraction and for the synthesis of the extracted information into humanly useful information resources, primarily using neural network and fuzzy logic methods.

Dr. Gedeon is the Regional Editor of the *International Journal of Systems Research* and *Information Science*, and is Past-President of the Asia Pacific Neural Network Assembly.

László T. Kóczy (M'92) was born in Hungary in 1952. He received the equivalent of the M.Sc. degree in electrical engineering, the M.Sc. degree in control engineering, and the Ph.D. degree in engineering, all from the Technical University of Budapest, Hungary, in 1975, 1976, and 1977, respectively.

Since 1976, he has been with the Department of Telecommunications, Technical University of Budapest, where he is currently a Professor. He was a Visiting Professor at the Dalian Maritime University, China, in 1990 (intensive summer term), the Pohang Institute of Science and Technology, Korea, in 1992, the Tokyo Institute of Technology, Yokohama, Japan, in 1993 and 1994 (holding the Fuzzy Theory Chair), the J. Kepler University, Linz, Austria, in 1994 (part-time), and the University of Trento, Italy, in 1995–1996 (part-time). His research interests include fuzzy systems, especially reasoning, modeling and control, image understanding, graphs and hardware components, telecommunications networks and control, and genealogy and heraldry.

Dr. Kóczy is the Elected President of the International Fuzzy Systems Association, a Founding Member of the EURO Working Group on Fuzzy Sets, and Associate Editor of *Mathware* and of *Fuzzy Systems and Artificial Intelligence*. He is a Fellow of the Hungarian Academy of Engineering.

Jae Dong Yang received the B.E. degree in computer engineering from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 1985 and 1991, respectively.

He is currently a Professor of the Department of Computer Science at Chonbuk National University, Chonju, Korea. His research area includes fuzzy information retrieval, fuzzy reasoning, automatic document categorization, and databases.