Proceedings of the First European Congress on Fuzzy and Intelligent Technologies, EUFIT'93, Aachen, 7-10 September 1993, pp. 1097 - 1104.

Self-Organizing and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems

Hartmut Surmann, Andreas Kanstein, Karl Goser
University of Dortmund, Faculty of Electrical Engineering
44221 Dortmund, Germany
Phone: +49 231 755 43 24, Fax: +49 231 755 44 50
Email: surmann@luzi.e-technik.uni-dortmund.de

Abstract: An automatic design method is proposed for fuzzy control and decision/diagnosis systems. This method extends traditional fuzzy systems by a learning ability without changing the fuzzy rule framework. The fuzzy rules and linguistic variables are extracted from a referential data set by a self-organizing process. A genetic algorithm is used to find optimal input/output membership functions.
Keywords: Fuzzy Control, Neural Networks, Genetic Algorithm, Automatic Design of Fuzzy Systems

I. Introduction

The transfer function of a fuzzy system (FS) is not based on a mathematical model but it is given by the definition of fuzzy rules and fuzzy sets of linguistic variables [1]. The fuzzy rules and fuzzy sets are designed on the basis of human operator's experiences, decisions and control actions. Like in conventional expert systems, the operator often cannot clearly explain why he acts in a certain way. Then an automatic design method becomes important, which is based on a set of examples for input/output relationship named *referential data* set. The methods derived from neural algorithms found in the literature can be divided into two levels. On the first level Pedrycz [2], Lin & Lee [3], and Kosko [4] use self-organizing procedures to design fuzzy systems. On the second level Lin & Lee [3], and Glorennec [5] transfer fuzzy systems in a neural structure to optimize the input/output behavior of the system. But the neural structure of the system has some disadvantages. The symbolic framework which represents structured knowledge is given up. Also its computing effort is quite high. Particularly the microelectronic realization of such networks seems to be very difficult. Due to its regular and modular structure the basic fuzzy control algorithm offers good and fast realization possibilities [6, 7]. As well as Lin & Lee, we propose a two step design method. In the first step we are searching for decision relevant situations and their consequences analogous to human learning. This will be done unsupervised with the self-organizing feature map (SOFM) [8], which is described in the second section. The information of the feature map will be used to construct an initial fuzzy rule base (section III.). In the second step of the design process a genetic algorithm is used to optimize the input/output behavior and to reduce the complexity of the fuzzy system. This is described in section four. In section V. an application of the design method on Anderson's *iris* data [9] and Box & Jenkins' *gas furnace* data [10] and a comparison to other results are given. Finally, some remarks on future research will conclude this paper.

II. Self-organization for the extraction of fuzzy points

The first step in the design of a problem which is to be solved by a fuzzy controller is the determination of the membership functions for the linguistic variables. Here the membership functions are defined with fuzzy points. These can be interpreted as the locations of clusters in the universe of a linguistic variable. Estimations of these fuzzy points will be computed with a SOFM, as described in the following.

First, we summarize the basic terminology and learning rule incorporated in SOFMs. Usually the map consists of a two dimensional array of processor elements (PEs), which are completely connected to all input nodes. At these, patterns $\vec{x} \in \mathbf{R}^n$ are presented, so each PE stores an n-dimensional weight vector $\vec{w} \in \mathbf{R}^n$. The unsupervised learning algorithm is an iterative, sequential process which finds the "best" set of weights for clusters in the universe of the learning patterns $\vec{x} \in \mathbf{X}$.

As Lin & Lee [3], we use this feature of the SOFM to estimate centers m of fuzzy points of every vector component x_j, $j = 1, \ldots, n$. In contrast to Lin & Lee, also the information given by the components of the weight vectors will be evaluated to construct the rule base. Only a one dimensional SOFM is used to compute q weight vectors, so the number of PEs equals the number of fuzzy points and the dimension of the weight vectors corresponds to the number of input and output variables. At the end of the learning process

every weight w_{ij} can be regarded as an estimator $\tilde{m}_{ij}$ for the center m_{ij} of the fuzzy point i of the linguistic variable j.

As advocated by Kohonen [8], the learning algorithm modifies the weight vectors $\vec{w}$ to reflect the structure and frequency occurring in the original input vectors. For a given input vector, the PEs compete among themselves and the winner (the PE which has the minimum distance to the input) performs the updating operation of its weight and the weights of some predefined neighbors. This update process is continued for a predefined number of learning cycles. So, the closer two input vectors $\vec{x}_a, \vec{x}_b \in \mathbf{X}$, the more evident their neighborhood property is visualised in the resulting map.

To compute the weights, a learning rate $h(t, r(t))$ and an adaptation radius $r(t)$ which decreases exponentially with learning cycle t have to be defined, as well as the number of learning cycles $t_{\max}$ and a norm $\|\cdot\|$. The update strategy is *random but fair*, that means in one learning cycle all p input (referential) vectors $\vec{x} \in \mathbf{X}$ are trained in a random sequence. A brief specification of the learning algorithm is:

1. Select q, the norm $\|\cdot\|$, $t_{\max}$, $r(0), r(t_{\max}) \in [0, q]$, and $h(0, r(0)), h(t_{\max}, r(t_{\max})) \in [0, 1]$.
2. Initialise $w_{ij} \in \mathbf{R}$ at random, $i = 1, \ldots, q, j = 1, \ldots, n$.
3. For $t = 0, \ldots, t_{\max} - 1$; For all $\vec{x} \in \mathbf{X}$;
   (a) Calculate $d_i = ||\vec{x} - \vec{w}_i||, i = 1, \ldots, q$.
   (b) Compute the position i_1 of the winner: $d_{i_1} = \min\{d_i | i = 1, \ldots, q\}$.
   (c) Update the weights: If $\vec{w}_i \in \{\vec{w}_x \mid i_1 - r(t) \leq x \leq i_1 + r(t)\}$ then $\vec{w}_i = \vec{w}_i + h(t, r(t))(\vec{x} - \vec{w}_i)$.

In the examples (section V.), the euclidean norm ($d_i = \sum_j (x_j - w_{ij})^2$) is selected for $\|\cdot\|$. The other parameters are set as follows: $t_{\max} = 200$, $h(0) = 0.9$, $h(t_{\max}, r(t_{\max})) = 0.1$, $r(0) = q$, $r(t_{\max}) = 0.001$.

III. Fuzzy points and rules

As described above, the weight w_{ij} can be regarded as an estimator of the center of a fuzzy point ($\tilde{m}_{ij} = w_{ij}$). Since the genetic algorithm (section IV.) will optimally adjust the centers and the widths of the fuzzy points, the widths can be estimated by the first-nearest-neighbor heuristic [3]: $\tilde{\sigma}_{ij} = |\tilde{m}_{ij} - \tilde{m}_{\text{nearest } j}|/r$, with an arbitrary overlap parameter r. The width $\tilde{\sigma}_{ij}$ of the fuzzy point is only a poor estimator for the standard derivation σ_{ij}, in contrast to the weight w_{ij} of the SOFM, which is a good estimator for the center (mean value m_{ij}) of a cluster.

As it becomes clear from this description, our view of fuzzy points is statistically based. This will be explained in the following. The basic elements for the problem of practical estimation of membership functions are defined within the theory of possibility introduced by Zadeh [11]. He states that " ... — contrary to what has become a widely accepted assumption — much of the information on which human decisions are based is possibilistic rather than probabilistic in nature." Furthermore, possibility is related to probability, because " ... a lessening of the possibility of an event tends to lessen its probability — but not vice versa." Interpreting this possibility/probability consistency principle, Dubois & Prade [12, p. 258] give a mathematical criterion of the association of the possibility and probability distributions

$$\Pi(D) = \sup_{x \in D} h(x) / \sup h \geq P(D) = \int_D h(x)dx / \int_{-\infty}^{\infty} h(x)dx, \tag{1}$$

in which h is a function $h : \mathbf{R} \to \mathbf{R}^+$ representing a smoothed histogram.

As it can be seen from (1), the possibility distribution is described by a possibility density function $\pi(x) = h(x)/\sup h$, analogous to a probability distribution, which is defined by a probability density function $p(x) = h(x)/\int_{-\infty}^{\infty} h(\xi)d\xi$. The distributions differ in the norm used to evaluate the histogram. The norm $\int_{-\infty}^{\infty} p(x)dx = 1$ stresses the need of global information about the presented measure, while with $\sup \pi = 1$ only local relations are used.

membership functions can be defined simply from possibility distributions by $\mu(x) = \pi(x)$. But this way of constructing membership functions is not really suitable, because h cannot always be calculated and needs not to fulfill equation (1). Furthermore, $\pi(x)$ needs not to be convex.

Instead of this, we will construct membership functions from probability density functions by $\mu(x) = \lambda p(x)$. The constant λ is calculated using the constraint $\sup \mu = 1$. Here the gaussian probability distribution

$p(x) = \varphi(x; m, \sigma^2)$ is chosen, because this distribution fits a lot of real world problems. Therefore, the membership functions used in our system are defined by

$$\mu(x) = \sigma\sqrt{2\pi}\,\varphi(x; m, \sigma^2) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right). \tag{2}$$

The advantage of this definition is that $p(x)$ and also $\mu(x)$ are described by the parameters expectation value m and variance σ^2. As explained before these parameters can be estimated by use of a SOFM.

At the left and right margin of the input linguistic variables special margin membership functions are defined. They will be derived from the normalised ($m = 0$, $\sigma^2 = 1$) membership functions

$$\mu_{\text{left}}(x) = \begin{cases} 1 & , x < 0 \\ e^{-\frac{1}{2}x^2} & , x \geq 0 \end{cases} \qquad \mu_{\text{right}}(x) = \begin{cases} 1 & , x \geq 0 \\ e^{-\frac{1}{2}x^2} & , x < 0 \end{cases}. \tag{3}$$

In a fuzzy control and decision system, fuzzy rules define the connection between input and output variables. Like the fuzzy points, the rules will be derived from the weight vectors of the SOFM. The weight vector's components w_{ij} represent fuzzy points i of linguistic variables j, so every vector $\vec{w}_i$ holds information about the connection between variables. This information will be used to construct the rules of the FS.

Here, fuzzy rule bases will be described for two basic problems, a classifier system and the simulation of a dynamic process. First, we consider a classifier problem with q classes, which is described by a referential data set with p patterns. It holds p_i patterns $\vec{r}_k^{\,i}$ ($k = 1 \ldots p_i$, $i = 1 \ldots q$, $\sum_i p_i = p$) for every class i. The classes need not to be mutually unrelated. With this referential data set a SOFM with q weight vectors is trained. Then, from every weight vector $\vec{w}_i$, $i = 1, \ldots, q$ a fuzzy rule for class i is defined:

if X_1 **is** A_{i1} **and** X_2 **is** A_{i2} **and** ...**and** X_n **is** A_{in} **then** Y_1 **is** *no* **and** ...**and** Y_{i-1} **is** *no* **and** Y_i **is** *yes* **and** Y_{i+1} **is** *no* **and** ...**and** Y_q **is** *no*.

A_{ij} is a label for the fuzzy point with center $\tilde{m}_{ij}$, Y_i is the output variable for the class i, and *yes*, *no* are labels for the fuzzy points

$$\mu_{\text{no}}(x) = \begin{cases} 2x+1 & , -0.5 \leq x < 0 \\ -2x+1 & , \quad 0 \leq x \leq 0.5 \end{cases} \qquad \mu_{\text{yes}}(x) = \begin{cases} 2x-1 & , \quad 0.5 \leq x < 1 \\ -2x+3 & , \quad 1 \leq x \leq 1.5 \end{cases}. \tag{4}$$

In contrast, the referential data set for process simulation consists of p vectors $\vec{r}$ of dimension $n = n_{\text{in}} + n_{\text{out}}$. Therefore, the weight vectors of the SOFM define n_{in} input and n_{out} output variables with q fuzzy points each. For process simulation, the rule base should be complete, i. e. for every input at least one rule fires. Therefore, the premises are conjunctive combinations of all input variables with permutated fuzzy points. This implifies that $q^{n_{\text{in}}}$ rules such as

if X_1 **is** A'_1 **and** X_2 **is** A'_2 **and** ...**and** $X_{n_{\text{in}}}$ **is** $A'_{n_{\text{in}}}$ **then** Y_1 **is** B'_1 **and** ...**and** $Y_{n_{\text{out}}}$ **is** $B'_{n_{\text{out}}}$

are defined, in which the fuzzy point A'_j is determined by $\tilde{m}_{ij}, \tilde{\sigma}_{ij}$ and the permutation of the index i.

To determine the B'_j, $j = 1, \ldots, n_{\text{out}}$ of a rule, the center points $\tilde{m}_k$, $k = 1, \ldots, n_{\text{in}}$ of the premise's fuzzy points are combined to an input vector for the SOFM. Then the most similar PE $\vec{w}_{i_1}$ is computed for this input vector. Thereby only the first n_{in} components are taken in consideration. From the output components of the winning PE $\vec{w}_{i_1}$ the centers $\tilde{m}'_j$ of the B'_j are determined as $\tilde{m}'_1 = w_{i_1, n_{\text{in}}+1}$, $\tilde{m}'_2 = w_{i_1, n_{\text{in}}+2}$, ..., $\tilde{m}'_{n_{\text{out}}} = w_{i_1, n_{\text{in}}+n_{\text{out}}}$.

Notes on fuzzy sets and rules generation

- Computing time increases approximately linear with the number of the input patterns (Size of the referential data set).
- Storage capacity increases linear with the number of linguistic variables. The extension by a new linguistic variable can be done easily by increasing the weight vector dimension.
- A fuzzy point (set) is defined by only two parameters $\tilde{m}, \tilde{\sigma}$.
- Conventional experienced methods can be used to determine the fuzzy sets and fuzzy rules.

IV. Genetic algorithm for adaptation

The adaptation aims for an improvement of a specified system behavior in respect to a new or changing task. Therefore, adaptation is a sort of optimization or search process. In difference to the common aim of

optimization, i.e. finding the best solution, adaptation means searching for improvement. Furthermore, for adaptation of a FS a robust algorithm is needed which can cope with complex search (parameter) spaces.

There is no theoretical approach that defines an optimal FS for a given task. Therefore, adaptation is a method to compute well designed FS. A general FS is very complex. Even when parts of the structure are fixed, e.g. by a hardware implementation, a lot of free parameters remain. Especially the fuzzy sets are important for fine tuning the transfer function of the system. In contrast, the rule base is less flexible, even when the rules are weighted. Considering the number of parameters of all fuzzy sets, the search space becomes very complex.

A suitable optimization algorithm for the adaptation of a FS is the genetic algorithm (GA) introduced by Holland, which is described in detail by Goldberg [13].

Like artificial neural networks, genetic algorithms are based on a biological concept: The concept of evolution. In difference to natural evolution, GAs are limited to the analysis of one system (here a FS with specific task) at a time, which is described by a parameter set. Furthermore, the GA uses a population $\mathbf{P}$ of constant size M, i.e. a variety of M systems. The population $\mathbf{P}$ develops in discrete time steps t, and the populations $\mathbf{P}(t)$ are called *generations*. The algorithm terminates after T time steps.

The development of the population with each time step is directed by competition between population elements. The information needed is a measurement for each FS performance quality, the *fitness* Q. The objective function Q has to be a positive value which increases with system's quality. Competition between population elements is included by relating this fitness to the total fitness of the population. The related measure is called *strength*. For calculation of a new generation (genetic) operators are used which work on a coding of the systems parameter sets, named *strings*. Therefore, the population elements are completely defined by their strings A, so $\mathbf{P} = \{A_1, A_2, \ldots, A_M\}$.

Binary coded parameters are used to construct a string. Every string A will hold all fuzzy set parameters m_{ij} and σ_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, q$ of a FS (as proposed by Karr [14]). The parameters are linearly transformed to integers of l bits accuracy (e.g. $l = 8$ bits) with constant limits for every variable. Denoting the coded fuzzy set parameters $\hat{m}_{ij}$ and $\hat{\sigma}_{ij}$, a string A is constructed by adding the parameters in a row:

$$A = \hat{m}_{11}\hat{\sigma}_{11}\hat{m}_{12}\hat{\sigma}_{12}\ldots\ \hat{m}_{nq}\hat{\sigma}_{nq} = \underbrace{a_1 a_2 \ldots\ a_l}_{\hat{m}_{11}} a_{l+1} \ldots\ a_L, \qquad a \in \{0, 1\}. \tag{5}$$

The length L of a string is $L = 2lnq$.

The calculation of a new generation $\mathbf{P}(t+1)$ from $\mathbf{P}(t)$ is done by sequentially applying the operators *selection*, *mutation* and *crossing over*:

First, two strings $A, B \in \mathbf{P}$ are selected and copies A^1 and B^1 of A and B are made. The probability of the selection of a specific string is proportional to its strength.

Second, each of these two strings are eventually changed by mutation. A mutated string A^2 is calculated by changing every a_i, $i = 1, \ldots, L$ of A^1 with the probability p_{mut}. B^2 is calculated the same way.

Third, with the probability p_{cross} a crossing over of A^2 and B^2 is executed. A crossing point x, $1 < x \leq L$ is chosen at random and then new strings A^3 and B^3 are constructed as

$$A^3 = a_1 a_2 \ldots a_{x-1} b_x b_{x+1} \ldots b_L, \qquad B^3 = b_1 b_2 \ldots b_{x-1} a_x a_{x+1} \ldots a_L. \tag{6}$$

If no crossing over is performed, then $A^3 = A^2$ and $B^3 = B^2$.

Finally A^3 and B^3 are taken into the new population. The calculation of new string pairs is repeated until the new population is filled. Then the fitness of the new population members is calculated and it can be proceeded with this population.

The parameters of the algorithm can be selected as follows. Fixed values are chosen for the algorithm's random parameters, so $p_{\text{mut}} = 0.01$ and $p_{\text{cross}} = 0.8$ (see [13]). M and T will be selected according to the system complexity.

At each time step t one is mainly interested in the performance of the best FS, i.e. in the string with the highest fitness value f. At time step $t = T$ this string is also viewed as the result of the GA. To preserve the best string of time step t from being changed by the genetic operators, this string might be copied unchanged into the next generation. This strategy called *fittest survive* does not always increase the performance of the algorithm, as it is shown in Fig. 1.

The fitness function itself depends on the task of the FS. Examples are given in section V.. For fitness calculation, a FS has to be constructed from each string and tested by evaluating the referential data set. Therefore, a lot of FS executions are needed. A discrete FS with a very fast evaluation method is used for this task (section V.c). The coding used for the GA matches the use of a discrete FS. In addition, the coding's limitations due to discretization and constant intervals are of small importance, because FSs are robust and the range of the referential data set is also limited.

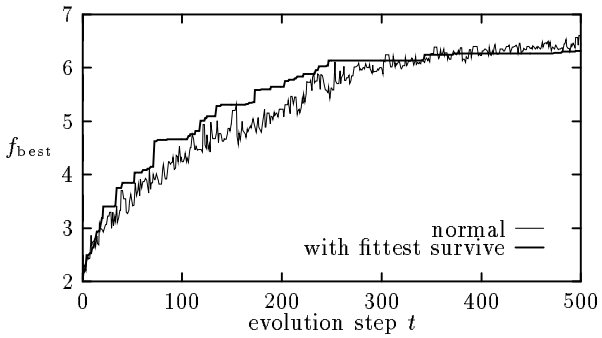

Figure 1: *Convergence of the best population element for an arbitrary example with and without the strategy fittest survive.*

model	rule array	remarks	errors J
Box & Jenkins [10]		Linear time series analysis with 6 parameters	0.196
		with an oscillation model (feedback)	0.058
Tong [19]	7×6	reduced to 19 rules	0.469
Pedrycz [2]	5×5	reduced to 20 rules	0.776
	9×9		0.32
Xu & Lu [20]	5×5	after identification	0.4555
	5×5	after adaptation	0.328
Sugeno & Yasukawa [21]	5×5	reduced to 6 rules	0.355

Table 1: *Results from literature for FSs with the gas furnace data. The column "rule array" contains the number of fuzzy sets per input variable.*

Notes on Genetic Algorithms

- GAs work with a population of competing solutions. Therefore, the algorithms can escape from local fitness maxima in the search (parameter) space.
- The genetic operators are probabilistic and are applied to coded parameter sets only. They are independent from the structure of the search space. Therefore, GAs are robust.
- Execution time grows linear with population size M, evolution steps T and length of strings L (under the assumption that FS calculation time is proportional to the number of fuzzy sets).

V. Examples

The design algorithm will be illustrated by means of two numerical examples to present the most characteristic features. For process simulation Box & Jenkins' *gas furnace* data set is used, which is a standard test for identification algorithms (Tab. 1). The data set consists of 296 pairs of input-output observations measured every 9 seconds. The input is the gas flow rate into the furnace and the output is the concentration of CO_2 in the exhaust gas.

Second, for the decision system we use Anderson's *iris* data set [15]. This data set consists of $p = 150$ (labelled) vectors (4 components) for $q = 3$ classes of iris subspecies (50 for each class). The properties of the data are well known and are used in various papers to illustrate unsupervised and supervised classifier designs. Typical error rates are $0 \ldots 5$ mistakes (resubstitution) for supervised and $10 \ldots 15$ for unsupervised designs [16].

As both design steps are influenced by random numbers, several simulations with different initial conditions were performed. Therefore, for most of the results two values are given (best and worst case). Also different composition/defuzzification methods are used (max-min, max-product and sum-product) [17, 18]. The membership functions are approximated with different numbers of lines, i. e. with 6 lines ($\mu(x) = 0$ for $x \notin [m - 2.9\sigma, m + 2.9\sigma]$), and 2 lines (triangular) ($\mu(x) = 0$ for $x \notin [m - 2.5\sigma, m + 2.5\sigma]$).

V.a Iris Data

Self-organizing map: For an initial FS we select $q = 3$ processor elements with $n = 4$ components and an overlap factor of $r = 2$. The resulting fuzzy classifier system (FCS) achieves $12 \ldots 15$ mistakes (tested

shape	inference	a/b	R_{max}	$1/Q_{0,c}^{292}$	R_ϕ	$\max(R_{act})$
gauss	sum-product	0.0/0.0	–	0.161 ... 0.182	9 ... 17	$20 \ldots 25^{1,2}$
gauss	sum-product	0.5/0.5	10	0.177 ... 0.216	4 ... 5	$6 \ldots 9^{2,3}$
triang.	sum-product	0.0/0.0	–	0.161 ... 0.188	9 ... 11	$15 \ldots 16^{1}$
triang.	sum-product	0.5/0.5	10	0.172 ... 0.235	5 ... 7	$6 \ldots 12^{1,2}$
gauss	max-min	0.0/0.0	–	0.208 ... 0.237	12 ... 19	$20 \ldots 25^{1}$
gauss	max-min	0.5/0.5	10	0.242 ... 0.284	4 ... 10	$8 \ldots 16^{1}$
triang.	max-min	0.0/0.0	–	0.187 ... 0.257	8 ... 13	$20 \ldots 25^{1}$
triang.	max-min	0.5/0.5	10	0.195 ... 0.26	4 ... 7	$9 \ldots 12^{1}$
gauss	sum-product	0.5/0.5	10	0.142	10	15^{4}

Table 2: *Optimization results with different inference methods and different membership function shapes for the gas furnace data, $p' = 292$, rule array 5×5^1, 5×4^2 and 4×5^2, 4×3^3, 5×3^4. The result of the last row is achieved with $p' = 234$ (80%).*

with different composition/defuzzification methods and fuzzy set shapes). A FCS which is generated supervised by statistical analysis of the referential data set achieves 5 mistakes, independently of the composition/defuzzification method and the shape of fuzzy sets.

To reduce the complexity, the excess η ($\eta = E(x-m)^4/\sigma^4 - 3$) of each input variable is computed. An input with small $|\eta|$ is not evaluated. In both examples, with $|\eta|_{\min} = 1$ only the components 3 and 4 have been evaluated. In this way the number of linguistic variables is reduced and the clarity of the fuzzy rule base (FRB) increases.

Genetic algorithm: An other problem beside the creation of coded strings (section IV.) is the definition of a suitable objective function. A suitable objective function for the classification problem is the sum of the hamming distances of the class membership values a_j and the nominal values (coded with $l = 8$ bits, $2^l - 1 = 255 \hat{=} true$, $0 \hat{=} false$).

$$Q_{0,d} = \frac{255\ p}{\sum_{i=1}^{p} q_i} \quad \text{with} \quad q_i = \begin{cases} ((255 - a_i) + \sum_{j=1, j\neq i}^{n_{out}} a_j)^2 & \text{, if } a_i \text{ failed} \\ (255 - a_i) + \sum_{j=1, j\neq i}^{n_{out}} a_j & \text{, else} \end{cases} \tag{7}$$

Independent of the shape and composition/defuzzification method, the resulting FS achieves two mistakes after $T = 200$ time steps with a population size of $M = 200$. The initial population is calculated with the centers and widths of the fuzzy points and a random variance of these parameters. The best result (one mistake) can be achieved evaluating the components 1, 3 and 4 of the input vector.

V.b Gas Furnace Data

Self-organizing map: In this example, the task is to build a rule based model from the referential data set which identifies the process. Particular for the process control, the referential data set can contain the observations and the control actions of a human operator. As well as the literature (Tab. 1), we choose two input variables ($n_{in} = 2$: $x(t-\tau_1), y(t-\tau_2)$) and one output variable $y(t)$. $x(t-\tau_1)$ denotes the input at the time $t - \tau_1$ and $y(t-\tau_2)$ the output of the process at $t - \tau_2$. With $\tau_1 = 4$ and $\tau_2 = 1$, the referential data set contains $p = 296 - \tau_1 = 292$ elements.

For the self-organizing identification, we select $q = 5$ fuzzy sets for each input/output variable. For that, the medium square distance of the FC output to the referential data is $1/Q_{0,c}^{292} = 0.88 \ldots 0.89$ (equation 8). Simulations with different numbers of input vectors p' (all, 80%, 60%, 40% of p, random chosen) during the training phase show results of $1/Q_{0,c}^{292} = 0.6 \ldots 1.1$.

Genetic algorithm: For process simulation, the reciprocal of the mean square error is a suitable objective function:

$$Q_{0,c}^{p'} = \begin{cases} 0 & \text{, if one } a_i \text{ undefined} \\ p' / \sum_{i=1}^{p'} (a_{ref,i} - a_i)^2 & \text{, else} \end{cases} \tag{8}$$

The GA generates new membership functions for a fuzzy process simulation system (FPSS). Compared to the literature (Tab. 1), this system achieves the best results (referring to the input/output values). The analysis of the internal states of the fuzzy controller (FC) shows that for every input vector a lot of rules

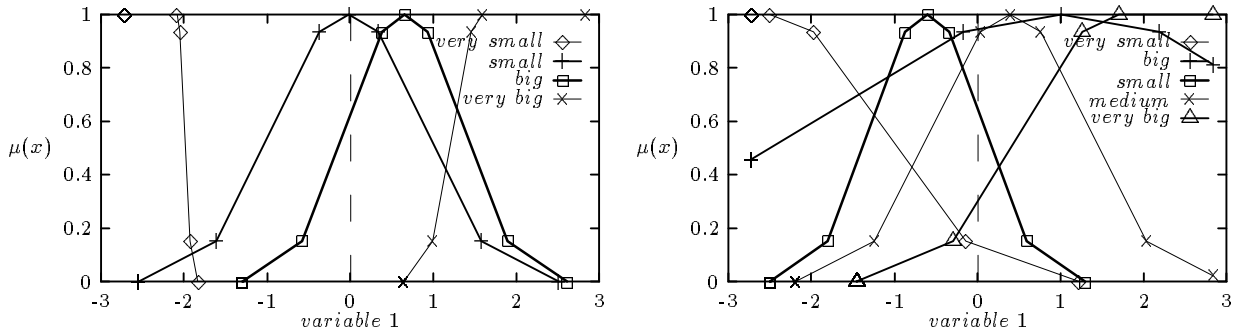

Figure 2: *Membership functions of the first variable after optimization, with and without entropy reduction.*

are activated and that the overlap and position of the membership functions does not correspond to human expectation and definitions (Fig. 2). So, the next step is to modify the objective function to reduce the number of activated rules. We define the *entropy* of a FS by the average number of activated rules:

$$R_\phi = \frac{\sum_{i=1}^{p'} R_{act,i}}{p'} \tag{9}$$

To decrease the entropy of a FS and the overlap of the membership functions, we define a nominal number R_{max} of activated fuzzy rules and modify the objective function by:

$$Q = \begin{cases} \dfrac{Q_{0,c}^{p'}}{(\frac{R_{act}}{R_{max}}-1)a+(\frac{R_\phi}{R_{max}}-1)b+1} & , R_{max} < R_{act} \\ \dfrac{Q_{0,c}^{p'}}{(\frac{R_\phi}{R_{max}}-1)b+1} & , \text{else} \end{cases} \tag{10}$$

$R_{max} \in \mathbf{N}$: nominal number of activated rules, $R_{act} \in \mathbf{N}$: actual number of activated rules, $a, b \in \mathbf{R}$.

If limitations such as a hardware restriction of a FS exists a factor $a > 0$ punishes a FS which activates too many rules for one input (local). The factor b works more global and punishes the FS if the average number of activated rules is too big. The results due to the input/output performance can be regarded in Tab. 2 ($R_{max} = 10$). An example of the membership functions is given in Fig. 2.

Notes on decreasing the entropy of a fuzzy system

- Takes hardware limitations in consideration.
- Increases the process performance.
- Limits the width of the membership functions and, if $\sigma = 0$, reduces the complexity (number of rules and membership functions).
- Corresponds to human expectations.

V.c Hardware requirement

A general purpose simulator for the proposed approach for SOFM, GA and FS has been written in C and C++ language and runs on a PC-486 and Sun SPARCstation. For a local area network with different workstations a distributed version of the SOFM and the GA is implemented. The computation time for the GA was about 50 minutes for the gas furnace data ($p = 292, T = 500, M = 500$) with 5 sun workstations ($\approx$ 33 MIPS$_\phi$, workstation utilization: 50%) and about 10 minutes for $p = 292, T = 200$ and $M = 200$.

In addition, a fuzzy development system [22] offers several possibilities including process simulation, fuzzy rule debugging and 2D/3D visualisation. A general purpose ANSI C-Code generator with a rule matching speed of 0.3...1.0 million fuzzy rules/second (PC-486/33 MHz) and 1.0...2.3 MFRPS (sparc10 70 MIPS) is also implemented (gas furnace simulation).

VI. Conclusion

Several conclusions may be drawn from these investigations. The proposed model introduces the self-organizing capabilities of neural networks and the optimization capabilities of genetic algorithms into a

fuzzy control and decision system and provides high-level human-understandable fuzzy rules. The fuzzy rule framework of the proposed system is not changed in contrast to neural fuzzy architectures [3, 5]. This allows a high rule matching speed and offers the possibility of a microelectronic realization [6, 7] of a general purpose adaptive fuzzy controller. Furthermore neural networks tend to hike in local optima, while the genetic algorithm reduces this possibility. Two examples of the design system demonstrate two major applications (fuzzy controller, fuzzy decision). Future research will focus on microelectronic realization of adaptive fuzzy controllers as well as different applications.

Acknowledgements: This work has been supported partly by the VW-Stiftung Hannover and the german department for research and technology BMFT, contract number 413-5893-01 IN 103 B/O. The authors would like to thank our students Kai Heesche and Martin Bieroth for their assistance.

VI. List of References

[1] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 28 – 44, 1973.

[2] W. Pedrycz, "An identification algorithm in fuzzy relational systems," *Fuzzy Sets and Systems*, vol. 13, pp. 153 – 167, 1984.

[3] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, vol. 40, pp. 1320–1336, 1991.

[4] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs NJ: Prentice Hall, 1991.

[5] P.-Y. Glorennec, "A Neuro-Fuzzy Controller with variable geometry," in *Proceedings of FUZZ-IEEE '92, San Diego*, 1992.

[6] H. Surmann, T. Tauber, A. Ungering, and K. Goser, "Architecture of a fuzzy-controller based on field programmable gate arrays," in *2nd International Workshop on Field-Programmable Logic and Applications*, 1992.

[7] A. Ungering, K. Thuener, and K. Goser, "Architecture of a PDM VLSI Fuzzy Logic Controller with Pipelining and Optimized Chip area," in *Proceedings of IEEE-FUZZ 1993, San Francisco*, 1993.

[8] T. Kohonen, *Self-Organization and Associative Memory*, ch. 5. Self-Organizing Feature Maps. Berlin: Springer, 1984.

[9] J. C. Bezdek, "Numerical taxonomy with fuzzy sets," *Journal of Mathematical Biology*, vol. 1, pp. 57 – 71, 1974.

[10] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco CA: Holden Day, 1976.

[11] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3 – 28, 1978.

[12] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, vol. 144 of *Mathematics in Science and Engineering*. Boston: Academic Press, 1980.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley, 1989.

[14] C. L. Karr, L. M. Freeman, and D. L. Meredith, "Improved fuzzy process control of spacecraft autonomous rendezvous using a genetic algorithm," *SPIE Intelligent Control and Adaptive Systems*, vol. 1196, pp. 274–288, 1989.

[15] E. Anderson, "The Irises of the Gaspe Peninsula," *Bulletin of the American Iris Society*, vol. 59, pp. 2 – 5, 1935.

[16] J. Bezdek, E. C.-K. Tsao, and N. R. Pal, "Fuzzy kohonen clustering networks," in *Proceedings of FUZZ-IEEE '92, San Diego*, pp. 1035 – 1043, 8-12.3.92.

[17] W. J. M. Kickert and E. H. Mamdani, "Analysis of a fuzzy logic controller," *Fuzzy Sets and Systems*, vol. 1, pp. 29 – 44, 1977.

[18] I. Togai InfraLogic, *FC110 DFP Digital Fuzzy Processor, Operation Manual*. 30 Corporate Park, Suite 107, Irvine, CA 92714: Togai InfraLogic, Inc, 1991.

[19] R. M. Tong, "The evaluation of fuzzy models derived from experimental data," *Fuzzy Sets and Systems*, vol. 4, pp. 1 – 12, 1980.

[20] C.-W. Xu and Y.-Z. Lu, "Fuzzy model identification and self-learning for dynamic systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-17, pp. 683 – 689, 1987.

[21] M. Sugeno and T. Yasukawa, "Linguistic modeling based on numerical data," in *Proceedings of IFSA '91 Brussels: Computer, Management & Systems Science*, 1991.

[22] H. Surmann, K. Heesche, M. Hoh, K. Goser, and R. Rudolf, "Entwicklungsumgebung für Fuzzy-Controller mit neuronale Komponente," in *VDE-Fachtagung Technische Anwendung von Fuzzy-Systemen*, pp. 288 – 297, 1992.