Optimisation of trapezoidal membership functions in a fuzzy rule system by the "bacterial algorithm" approach[♣]

János Botzheim, Botond Hámori, László T. Kóczy, Budapest University of Technology and Economics
Department of Telecommunications and Telematics[1]

***Abstract* - This paper presents a method of using the so-called "bacterial algorithm" [4, 5] for extracting the rules of a fuzzy system. The class of membership functions is restricted to trapezoidal, as it is general enough and widely used. The algorithm contains the bacterial mutation step allowing both the change of more than one membership function at one time, and the fine-tuning. Further operators are applied for optimising the number of rules in the base, eliminating ineffective rules and contract similar rules into a single one.**

***Index Terms* - Bacterial algorithm, fuzzy system, hybrid system, membership functions**

I. Introduction

In the applications of fuzzy systems [1] one of the most important task is to find the optimal rule base. This might be given by a human expert or might be given a priori by the linguistic description of the modelled system. If however neither an expert, nor a linguistic description are available, the system has to be designed by other methods based on numerical data [5]. Nature inspired some evolutionary optimisation algorithms suitable for global optimisation of even non-linear, high-dimensional, multimodal, and discontinuous problems. The original genetic algorithm (GA) was developed by Holland [2] and was based on the process of evolution of biological organisms. GA-s use three well-known operators: reproduction, crossover, and mutation. Closely related evolutionary programming (EP) was proposed by Fogel [3]. EP uses selection and mutation operators. A more recent approach is the bacterial algorithm. This includes a modified mutation operator called bacterial mutation, emulating a biological phenomenon of microbial evolution. This gives an alternative by other algorithms, because it is simpler, and it is possible to reach lower error within a short time. In Section II the bacterial algorithm for trapezoidal fuzzy systems is described. Simulation results are in Section III. Section IV concludes the paper.

II. Fuzzy rule base identification by bacterial algorithm

The algorithm is similar to that used in [4]. The main difference is that trapezoidal membership functions are used. They are described by four parameters with the four breakpoints of the trapezium. Moreover the membership functions are identified by the two indices *i* and *j*. So, the membership function $A_{ij}(a_{ij},b_{ij},c_{ij},d_{ij})$ belongs to the ith rule

[♣] Supported by the National Science Research Fund (OTKA), Grant nos. T034233 and T034212.

and the jth input variable. $B_i(a_i,b_i,c_i,d_i)$ is the output membership function of the ith rule. The relative importance of the jth fuzzy variable in the ith rule:

$$A_{ij}(x_j)=\begin{cases} \dfrac{x_j-a_{ij}}{b_{ij}-a_{ij}}, & \text{if } a_{ij}<x_j<b_{ij} \\ 1, & \text{if } b_{ij}\le x_j<c_{ij} \\ \dfrac{d_{ij}-x_j}{d_{ij}-c_{ij}}, & \text{if } c_{ij}\le x_j<d_{ij} \\ 0, & \text{otherwise} \end{cases}$$

where $a_{ij} \le b_{ij} \le c_{ij} \le d_{ij}$ must hold.

So the encoding method of a fuzzy system with two inputs and one output, see in Fig. 1.

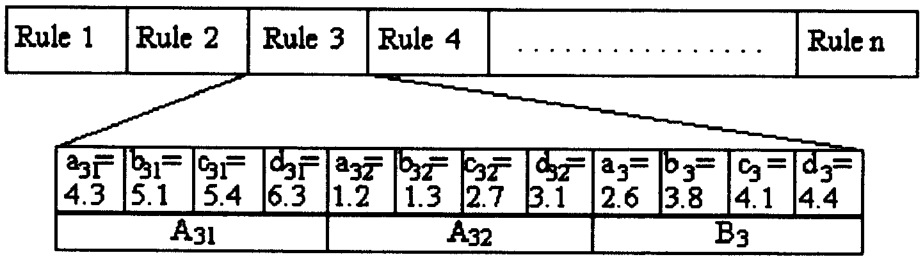

Fig. 1. Fuzzy rules encoded in a chromosome

For example, Rule 3 in Fig. 1 means:

if x_1 *is* $A_{31}(4.3,5.1,5.4,6.3)$ *and* x_2 *is* $A_{32}(1.2,1.3,2.7,3.1)$ *then* y *is* $B_3(2.6,3.8,4.1,4.4)$

where A_{ij} and B_i mean the trapezoidal membership function with the four breakpoints. The algorithm is described as follows [4, 5]

A. Generating the initial bacterium:

First the initial (random) bacterium is created. This means that all membership functions in the chromosome must be randomly initialised. The initial number of rules is N_{max}. So, in this step $(k+1)N_{max}$ membership functions are created (k is the number of input variables in the given problem, and each membership function has four parameters).

B. Bacterial mutation :

After the initialisation process the algorithm goes through several cyclic steps. First, $m-1$ copies (clones) of the rule base are generated [4]. Then a certain part of the chromosome [5] is randomly selected and the parameters of this selected part are randomly changed in each clone (mutation). Next all the clones and the original bacterium are evaluated by the error criterion. The best individual transfers the mutated part into the other individuals. This cycle is repeated for the remaining parts, until all parts of the chromosome have been mutated and tested. At the end the best rule base is kept and the remaining *m* -1 are discharged. The above procedure is repeated once more, however, with different parameters [4]. It is an important question how long is one part suffering mutation and what is the degree of the mutation (expressed as the relative size in terms of the interval). This approach allows both selecting more than one membership function and fine-tuning. The number of mutated membership functions and the mutation degree are external parameters of the bacterial mutation. If selecting more than one membership function is allowed then the local minima in the optimisation process can be avoided. Application of changing two or more membership functions at a time with fine-tuning in the first bacterial mutation and in the second step changing one membership function in the whole interval of the given variable was proposed in [4].

C. Fuzzy operators :

Bacterial mutation optimises the membership functions in the rule base. Besides, it is important to optimise the structure of the rule base. Ineffective rules must be eliminated and similar rules must be contracted. The following operators help to optimise the number of rules:

1. Annihilation

When a membership function becomes too narrow the rule using it must be deleted. The evaluation criterion (Fig. 2) is:

$$l_i \mu_i \left(\frac{a_j + b_j + c_j + d_j}{4} \right) \geq \beta l_j$$

where l_i and l_j are the lengths of the medians of the membership functions of the given input or output variables in the ith and jth rule and β is the annihilation parameter. The larger is the value of β, the more severe is the annihilation criterion.

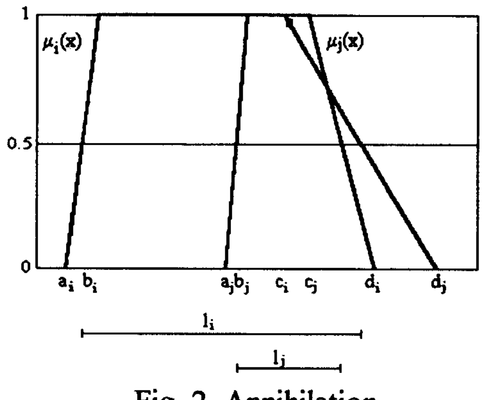

Fig. 2. Annihilation

2. Fusion

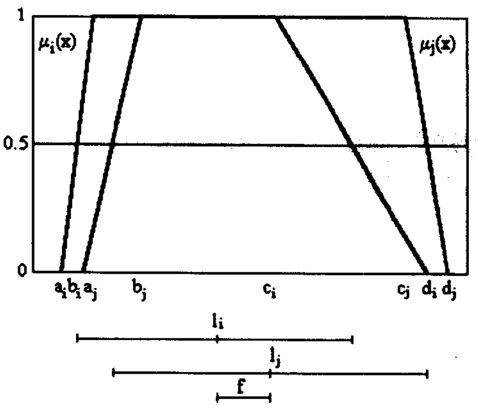

Fig. 3. Fusion

If two membership functions, belonging to the same variable are near to each other, and the difference between the lengths of their median is small enough, then they are fused in a single membership function. (Fig. 3.)There are two criteria of fusion, accordingly:

$$\left|\frac{l_i}{l_j}-1\right|<\gamma \quad \text{and} \quad |f|<\gamma,$$

where l_i and l_j are the lengths of the median of the membership functions of the given variable in the ith and jth rule, and f is the distance between the centres of l_i and l_j. To execute fusion both criteria must be satisfied, but only one parameter, g is used. The smaller is the value of g, the more severe is the criterion of fusion. The parameters of the fused membership function will be:

$$z_{fus}=\frac{z_i l_i + z_j l_j}{l_i + l_j},$$

where z stands for the four breakpoints (a, b, c and d) of a trapezoidal membership function.

3. Semantic analysis

If two rules have the same antecedents but a different consequent, the membership functions of the output variable are fused in one output membership function by using the fusion described above [4].

4. Rule removal

As COG defuzzification is used, removal of a rule [4] generates large error values so it decreases the performance of the fuzzy system. A new method is proposed here where the importance of the remaining rule is increased so it represents the two identical rules in one. For example if the ith and the jth rules are identical then the jth rule is deleted and the parameters of the new output membership function of the ith rule will be calculated as the following:

$$\begin{aligned}
m_i &= 2d_i - 2a_i + c_i - b_i \\
n_i &= c_i^2 + 2c_i d_i + 2d_i^2 - 2a_i^2 - 2a_i b_i - b_i^2 \\
a_i' &= \frac{n_i - m_i^2 - c_i m_i}{2m_i + c_i - b_i} \\
d_i' &= a_i' + m_i
\end{aligned}$$

where a_i, b_i, c_i and d_i are the breakpoints of the trapezoidal membership function of the output variable and a_i' and d_i' are the new breakpoints for a_i and d_i. Parameters b_i and c_i remain unchanged.

D. Stop condition :

If the bacterium satisfies a stop condition or the maximum generation number is reached then the algorithm ends, otherwise it returns to the bacterial step.

III. Results

A. Definition of the error function :

The widely used error functions in earlier publications defining the error value are as follows [4]:

$$e_1 = \frac{1}{n} \sum_{patterns} |y - \hat{y}| \text{ or } e_2 = \frac{1}{n} \sum_{patterns} (y - \hat{y})^2,$$

where n is the number of the evaluated patterns, $\hat{y}$ is the desired output of the modeled system for an input pattern and y is the output of the fuzzy system for the same input. The main problem with this formula is that it is not normalised, so the error results cannot be compared with each other. Another formula used [5] is the following:

$$e_{rel} = \frac{1}{n} \sum_{patterns} \left| \frac{y - \hat{y}}{\hat{y}} \right|.$$

The problem with this expression is that the error depends on the actual value of the output in a non-normalised way. If the expected output value is smaller, the error seems to be larger with the same difference. In our approach we introduced a different error function in order to reach better approximation of the model:

$$e_{norm} = \frac{1}{n} \sum_{patterns} \frac{|y - \hat{y}|}{I_{\max} - I_{\min}},$$

where I_{max} is the upper and I_{min} is the lower bound of the interval of the output variable, so the error is normalised by the length of the output interval rather than the actual value of the output. This approach helps with keeping the accuracy of the approximation of the patterns always constant, not being sensitive to the absolute value of the desired output. At the same time it has the advantage of being a normalised value keeping the error within [0,1].

B. Simulation Results :

1. Target function:

To test the algorithm, the following widely used target function [4, 5] was chosen:

$$y = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)},$$

where

$$x_1 \in [1,5], x_2 \in [1,5], x_3 \in [0,4], x_4 \in [0,0.6], x_5 \in [0,1], x_6 \in [0.0,1.2].$$

The training set included 200, the validation set 500 patterns. The algorithm was running for 40 generations with 2 different bacterial operations in each and using 20 clones. Next we summarise some of the interesting points.

2. Number of changing membership functions :

It is possible that the optimisation procedure is trapped in a local optimum and further optimisation becomes impossible when only one membership function is modified in each generation in a rule. This way more than one membership functions have to be changed, as it was experienced through the previous approximation tests. In this method the number of membership functions to change is fixed. Every membership function is allowed to change in one generation. During our tests with the six input variables function two or three changes turned out to be sufficient and larger numbers of changes showed no visible effect on the result as described in Figs. 4 and 5, where β= 5, γ = 5% and the initial number of rules was 10. The values in the table are averages of 4 results.

MF change	Rule No.	Train. err. [%]	Test err. [%]
1	5,75	4,30425	8,38225
2	4,75	5,66925	9,02325
3	6	4,8755	8,6555
4	4	6,6995	8,78875
5	3,75	6,13475	9,2765
6	4,75	5,646	8,366
7	5,75	5,948	9,28875

Fig. 4

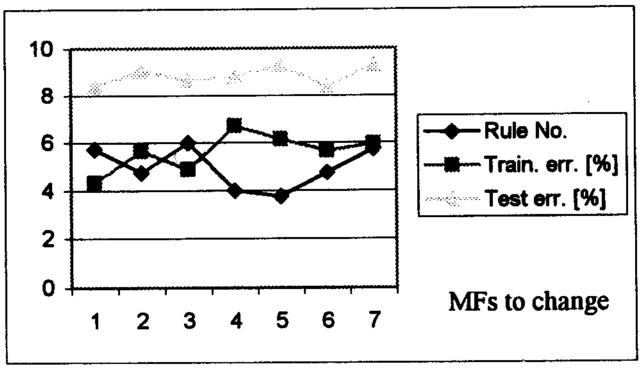

Fig. 5

We found that if the number of membership functions to be changed was increased, the approximation accuracy of the above mentioned benchmark function was not affected significantly, presumably because of the high complexity of the problem.

3. Fine-tuning:

In our tests fine-tuning the membership functions was included and we found that if a membership function remained in the rule base it meant generally that it was important for the fuzzy system even if it might not be optimal. It was possible to improve the accuracy of a fuzzy system by changing the breakpoints of the trapeziums by small random values.

4. Reducing complexity of a fuzzy system:

The use of the operations fusion, annihilation, semantic analysis, and rule removal proved to be an effective tool to optimise the size of fuzzy rule bases by reducing redundancy. Several tests were taken to identify the optimal parameters to avoid generating "too simple" rule bases and improving performance. Test results are shown in Figs. 6 and 7 γ= 5% and the initial number of rules was 10. The values in this table are averages of 4 results.

β	Rule No.	Train. err. [%]	Test err. [%]
5	4,75	5,66925	9,02325
6	5,75	5,24725	8,6675
7	7,5	4,78775	9,44525
8	9	4,455	9,311
9	8,75	4,55125	8,9235
10	8	4,261	9,01575

Fig. 6

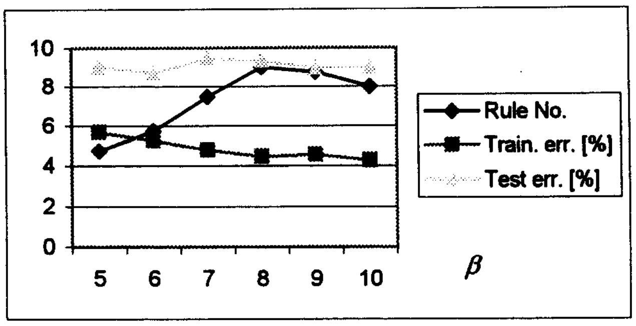

Fig. 7

With the increase of the value of β the remaining rule number increased too, because the criteria of annihilation got more severe. As the rule number in the resulting base increased, the error measured during the training phase decreased however the error measured with the validation set did not. We interpreted it that the resulting large rule base was too specific for the training set and had no advantage when applied for an independent pattern set. We concluded that it was not necessary to give β high values as it just generated more complex rule bases with the same capability.

IV. Conclusions

In this paper a new algorithm for the optimisation of the membership functions was shown. We intended not only to optimise the rules but also to reduce the size of the rule base. With trapezoidal membership functions the problem could be solved with fewer rules than with triangular ones. By applying fuzzy operators, optimal rule base size could be achieved.

References

[1] L.A.Zadeh: Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Tr. Systems, Man and Cybernetics* **3** (1973), pp. 28-44.
[2] J.H.Holland: *Adaptation in Nature and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, 1992.
[3] L.J. Fogel, A.J.Owens, and M.J.Walsh: *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.
[4] M.Salmeri, M.Re, E. Petrongari, and G.C.Cardarilli: A Novel Bacterial Algorithm to Extract the Rule Base from a Training Set, Dept. of Electronic Engineering, University of Rome, 1999.
[5] N.E.Nawa, and T.Furuhashi: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, *IEEE Tr. Fuzzy Systems* 7 (1999), pp. 608-616.