A Study on Fuzzy Rules Discovery Using Pseudo-Bacterial Genetic Algorithm with Adaptive Operator

Norberto Eiji Nawa, Tomonori Hashiyama, Takeshi Furuhashi, Yoshiki Uchikawa
Laboratory of Bio-Electronics, Department of Information Electronics
School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-01, JAPAN
Tel.+81-52-789-2793, Fax.+81-52-789-3166
e-mail: {eiji,tom,furu}@bioele.nuee.nagoya-u.ac.jp

Abstract— **This paper presents a new operator called *adaptive operator* for the Pseudo-Bacterial Genetic Algorithm (PBGA). The PBGA was proposed by the authors as a new approach combining a genetic algorithm (GA) with a local improvement mechanism inspired by a process in bacterial genetics. The PBGA was applied for the discovery of fuzzy rules. The aim of the newly introduced *adaptive operator* is to improve the quality of the generated fuzzy rules, producing blocks of effective rules and more compact rule bases. The new operator adaptively decides the division points of each chromosome for the bacterial mutation and the cutting points for the crossover. In order to verify the efficiency of the proposed adaptive operator, the PBGA is applied to a simple fuzzy modeling problem. The new operator actuates according to the distribution of degrees of truth values of the rules. The results show the benefits that can be obtained with this operator.**

Keywords— **Fuzzy Modeling, Genetic Algorithm, Bacterial Genetics, Hybrid Systems.**

I. Introduction

FUZZY systems can represent non-linear systems using linguistic variables in a straightforward form when enough knowledge about the object system is available. However, when the necessary knowledge is incomplete or can not be easily described, alternative methods for building a fuzzy system have to be used. Hybrid systems that utilize the methodologies of Soft Computing (fuzzy logic, neural computing, genetic computing, etc.) provide the perspective to overcome these difficulties.

The Pseudo-Bacterial Genetic Algorithm (PBGA)[1] implements a local improvement mechanism based on the genetic recombination of bacterial genetics. It is efficient in improvements of local portions of chromosomes. The chromosomes are divided into several portions and each of them is improved by the bacterial mutation of the PBGA. The PBGA was applied for the discovery of fuzzy rules. However, there was no criterion for the determination of the division points of the chromosomes.

This paper introduces an adaptive operator in the PBGA to determine the division points for the bacterial mutation and also the cutting points for the crossover operator. The adaptive operator decides the division/cutting points according to the degrees of truth values of the fuzzy rules. This special operator works effectively to discover sets of fuzzy rules with less numbers of irrelevant rules, i.e. unused parts of chromosomes.

The obtained results show the feasibility of this new operator.

II. Fuzzy Logic

Fuzzy Logic (FL) was proposed by Zadeh [2] in the 60's as a superset of the traditional bi-valued logic. In opposition to traditional logic, where the allowed values are only 0 (false) or 1 (true), FL can deal with different degrees of truth values. This ability of processing partial truth has been greatly utilized in engineering applications since then, mainly in the form of Fuzzy Logic Controllers (FLCs), Fuzzy Models and Fuzzy Expert Systems. For a more detailed introductory text on FL, please refer to [3].

The heart of fuzzy systems is the inference method based on FL. The inference method is based on a set of IF-THEN rules and membership functions of the variables. For example, one rule for a FLC of a chemical reactor could be:

```
if Pressure is MEDIUM
 and
   Temperature is HIGH
 then
   Open_Valve is HIGH
```

In the rule above, `Pressure` and `Temperature` are the antecedents of the rule (input variables), `Open_Valve` is the consequent of the rule (output variable) and {`MEDIUM`, `HIGH`, `HIGH`} are membership functions. Fuzzy inference systems have several rules like the one described above. The inference process follows the steps described below::

1. Fuzzification: Real input values (crisp values) are applied to the antecedent part of the rules. The degree of membership is calculated for each input value and input variable;
2. Inference: The degree of truth of the antecedent of the rule is calculated and applied in the consequent. For example, let's say that the degree of membership of '`Pressure is MEDIUM` ' is 0.46 and for '`Temperature is HIGH`' is 0.87. One common inference method is the MIN method, which takes the de-

gree of truth of one rule as the minimal value of membership value of the input variables. So in the example, the degree of truth would be 0.46 and the inferred action would be '`Open_Valve is HIGH`' with membership value 0.46;

3. Composition: A fuzzy inference system is composed of several rules of the type described above. In the inference process, more than one rule can be fired at the same time (actually this is very likely to happen, due to the overlap of membership functions). For each output variable, one fuzzy subset must be calculated by composing each one of the inferred actions and the respective membership values.
4. Defuzzification: For most of applications it is necessary to obtain a crisp number after the inference process. For example, it is useless to conclude that the valve should be highly opened. In that case, a precise numerical value is necessary. The composed fuzzy subset is defuzzified in order to enable a precise action. There are several methods of defuzzification. The CENTROID method calculates the crisp value by finding the center of gravity of the composed fuzzy subset.

For a introductory article on fuzzy inference systems applied to control, refer to [4].

III. Fuzzy Models and Genetic Algorithm

Systems that use fuzzy inference have been used in a wide range of applications. Fuzzy inference uses IF-THEN type linguistic rules that can deal with vagueness of concepts and incorporate knowledge of experts. These characteristics are especially useful for modeling systems for which there is no analytical description available. Basically, the design process of a fuzzy model consists to determine:

1. the input and output variables;
2. the membership functions of each variable;
3. the fuzzy rules, and
4. the parameters in 2 and 3.

Tasks 1 to 3 are related to the definition of the the structure of the fuzzy model. Task 4 is related to the tuning of the parameters of the model.

Even for the cases where there is an expert available or when a linguistic description of the modeled system can be found, a lot of adjustments of the fuzzy models based on *trial-and-error* are necessary. If there is not an expert available nor a linguistic description, the design process of the model has to start from the scratch. In order to overcome these difficulties, automatic design methods and rule acquisition procedures for fuzzy systems have been proposed, mostly making use of hybrid systems with artificial neural networks and/or genetic algorithm(GA)[1], [5], [6], [7].

GA has been used combined with FL in the design of Fuzzy Logic Controllers (FLCs) [8], [9], [10]. The strong point of hybrid systems combining GA and FL is that almost all the tasks that comprise the design process of a fuzzy model can be accomplished automatically. The structure of the model and its parameters can be determined by the same hybrid system. In order to improve the efficiency in the discovery of effective fuzzy rules, an additional operator is introduced in this paper.

IV. PBGA

A. Bacterial Genetics

The process of bacterial recombination that inspired the PBGA is the following: Bacteria can transfer DNA to recipient cells through mating. Male cells transfer strands of genes to female cells. After that, those female cells acquire characteristics of male cells and transform themselves into male cells. By these means, the characteristics of one bacteria can be spread among the entire bacteria population.

Another analogy is possible. Bacteriophages can carry a copy of a gene from a host cell and insert it into the chromosome of an infected cell. This process is called *transduction*. By *transduction*, it is also possible to spread the characteristics of a single bacterium to the rest of the population. These genetic recombination mechanisms have configured a process of microbial evolution. Mutated genes can be transferred from a single bacterium to others and lead to a rapid evolution of the entire population.

B. Algorithm Description

A similar process to the bacterial genetics is implemented in the PBGA. Its algorithm is briefly described as follows (Figure 1):

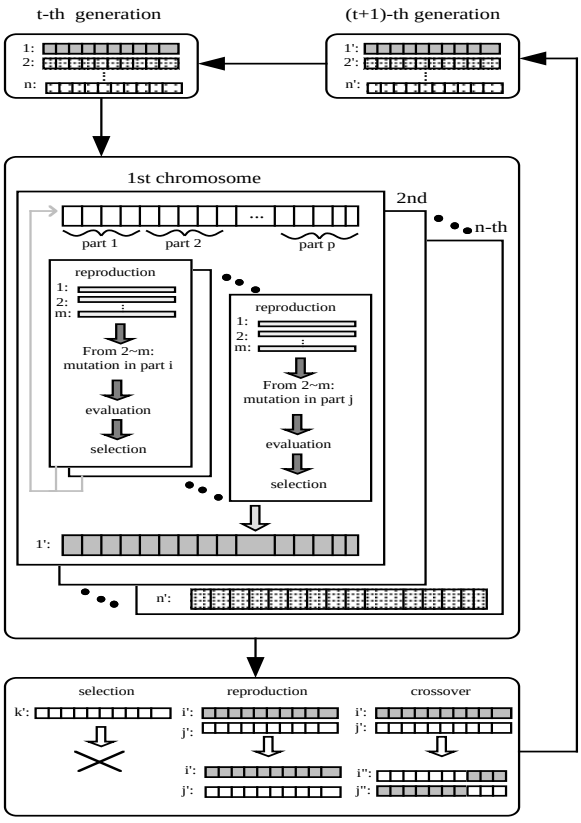

Fig. 1. Scheme of the PBGA

1. *Generation of the initial population*: n chromosomes are created and evaluated;
2. *Genetic Operators*:
   (a) *Mutation and Selection of genes*: This genetic operation is applied to each chromosome one by one. Chromosomes are arbitrarily divided in p parts (in the case of a chromosome encoding a fuzzy model, a group of fuzzy rules consitutes one *part*). One chromosome k is chosen and it is reproduced in m clones. The i-th part (randomly chosen) of m-1 clones is mutated. The elite among the m chromosomes is selected and the rest is deleted. The described process, reproduction-mutation-evaluation-selection, is repeated. The mutation is applied to another randomly chosen part, different from the already chosen ones.
   This genetic operation is applied to all the n chromosomes in the population.
   (b) *Selection and reproduction of chromosomes*: The chromosomes with lower fitness values are deleted and some randomly chosen chromosomes from the remaining group are reproduced.
   (c) *Crossover*: Chromosomes are mated and offsprings are generated by crossover.
3. *Stopping condition*: If the stopping condition is satisfied, stop, otherwise, go back to 2.

C. Adaptive Operator

In the previously described process of mutation and selection of genes, the division points that determine the parts of the chromosomes were arbitrarily fixed during the whole search process. Also, the crossover points in the chromosome were randomly chosen. In order to improve the performance of identification of fuzzy models by the PBGA, a new operator was introduced.

Fuzzy inference systems have rule bases that represent the core of the object system. In the case of a fuzzy controller, for example, the rules will describe the control algorithm of a given process. For fuzzy models, the rules are the models themselves.

When real values of input variables are applied to the antecedents of a fuzzy rule, a truth value is calculated. The truth value describes the degree of truth of the premise of a rule, therefore, the intensity that the consequent of the rule will apply.

The new operator, called adaptive operator, is used to determine the division points when the local improvement operation of the PBGA is to be applied. The probability of a certain *locus* to be selected is inversely proportional to the degree of truth value of the fuzzy rule it encodes. For each chromosome, the *moving average* of each fuzzy rule is calculated. The moving average is defined as the average of the accumulated truth values of the rules. The accumulated truth value of a rule is the sum of the truth values for each one of the entries in the training data. The moving average of the rule j, M_j can be defined as:

$$M_j = \frac{T_{j-2} + T_{j-1} + T_j + T_{j+1} + T_{j+2}}{5} \qquad (1)$$

where T_j is the accumulated truth value of the fuzzy rule j.

The accumulated truth value of a fuzzy rule is a measure of its quality. If a rule possesses a high value of accumulated truth value, it means that that rule was intensively and frequently triggered during the PBGA process. Consequently, this is an evidence of the effectiveness and utility of that rule. On the other hand, if a rule possesses a low value of accumulated truth value, this is an indication that the rule does not play an important role in the system.

The motivation for using the moving average of the accumulated truth value is to identify portions of good quality rules in the chromosome. Those strings having high moving averages are considered to have a high concentration of good quality rules. These strings are desirable to be preserved. On the other hand, blocks with low moving averages have higher probabilities of being mutated. The crossover points are also adaptively determined according to the moving averages of the accumulated truth values of the fuzzy rules. The lower the moving average, the higher the probability of being selected as the crossover point. This selection of crossover points works well to build blocks with effective fuzzy rules. Effective fuzzy rules are highly desired in order to build more compact models. Figure 2 shows a schematic of the adaptive operator.

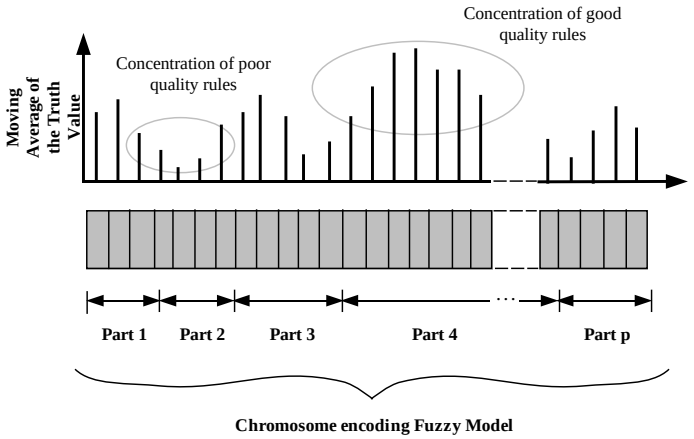

Fig. 2. Scheme of the adaptive operator

V. Encoding Method

Each chromosome in the population encodes the rules of the rule base of the fuzzy model as well as the membership functions of the variables. The lengths of the chromosomes are variable. Each rule contains information about the variables in the antecedent and consequent parts. The data about each of the membership functions are also encoded in the chromosome. In this paper, the membership functions are triangle-shaped, so their parameters are the pairs *(center, width)*. For example, the Rule 1 in Figure 3 means:

```
if X1 is T(4.238, 3.423)
 and
   X3 is T(3.428, 1.213)
```

```
and
  X4 is T(1.382, 0.381)
then
  Y  is T(12.112, 9.141)
```

where `T(c,w)` means the triangle-shaped membership function whose center is `c` and width `w` and where `X1`, `X3` and `X4` are input variables of the fuzzy model and `Y` is the output variable. This encoding gives a high degree of freedom for the PBGA, which can define the variables to be used in the rules, the rules themselves and the parameters of the membership functions.

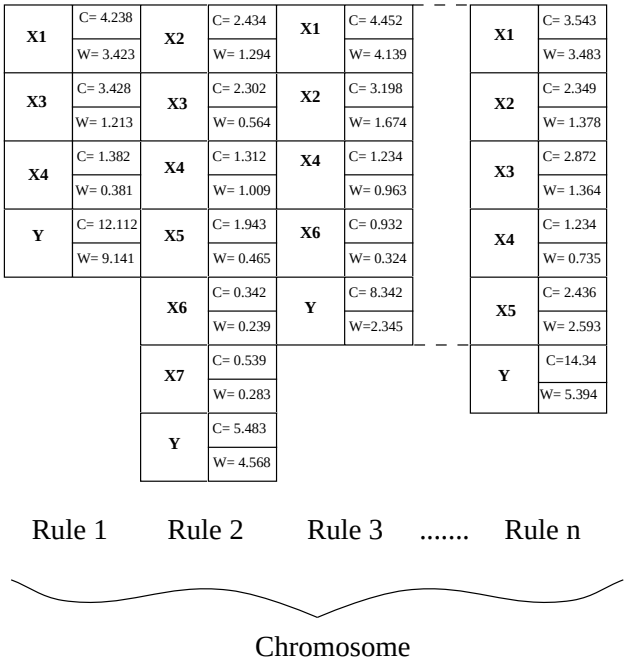

Fig. 3. Example of a Fuzzy Model encoded in a Chromosome

VI. Experiments and simulation results

In order to test the efficiency of the adaptive operator, experiments were performed to build a fuzzy model of the following function:

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

with $x_1 \epsilon [1 \ldots 5]$, $x_2 \epsilon [1 \ldots 5]$, $x_3 \epsilon [0 \ldots 4]$, $x_4 \epsilon [0.0 \ldots 0.6]$, $x_5 \epsilon [0.0 \ldots 1.0]$ and $x_6 \epsilon [0.0 \ldots 1.2]$.

The function $y(x_1, x_2, x_3, x_4, x_5, x_6)$ was modeled on 7 fuzzy variables, 6 variables on the function plus one dummy variable. 80 randomly chosen 6-tuples were used as training data entries. The population of chromosomes was fixed in 30 chromosomes during the whole search process.

The first session of simulations were performed using the following performance index (PI):

$$PI = \frac{1}{n_{entries}} \sum_{i=1}^{n_{entries}} \frac{|y_i - y_i^*|}{y_i} \qquad (3)$$

which is the average error of the inferred values of the fuzzy models built by the PBGA with the adaptive operator to the values of the training data set (y_i is the desired value, y_i^* is the inferred value and $n_{entries}$ is the number of entries in the training data set, 80).

The obtained results concerning the error of the built models are shown in Table I.

TABLE I
Error of the identified models when using PI (average of 100 runs)

	80 entries	160 entries
PBGA	23.71%	24.62%
PBGA+Adap.Op	8.04%	10.25%

From the data in Table I, it can be clearly seen that the PBGA with the adaptive operator can design better models than the PBGA alone. The column entitled "80 entries" contains the value of the error of the models when they are tested to the 80 entries of the training data. The column entitled "160 entries" contains the value of the error of the models when they are tested to the 80 entries of the training data set plus another 80 entries. This is a measure of the quality of the built models and indicates that there are no considerable changes when the model is tested with entries out of the trained set.

TABLE II
Fuzzy rules of the identified models using PI (average of 100 runs)

	Number of Rules	Non-used Rules
PBGA	26	46.30%
PBGA+Adap.Op	47	34.19%

The third column in Table II, "Non-used Rules", indicates that the ratio of the irrelevant rules included in the models built by the PBGA with the adaptive operator is lower than that of the models obtained by the PBGA only. In other words, the combination of PBGA with the adaptive operator is able to build models with more effective rules and less non-used rules. However, as it is possible to see from the second column of the Table II, "Number of Rules", the models built by the PBGA with the adaptive operator are bigger than the models built by the PBGA.

The following supposition was made: the lack of explicit indication to the PBGA to minimize the number of rules of the built fuzzy models was not allowing the construction of more compact models. Even though the effect of the adaptive operator in building active rules is present, an

additional force is necessary in order to make the PBGA build more compact models.

In the second session of simulations, the performance index was modified (PI').

$$PI' = \frac{1}{n_{entries}} \sum_{i=1}^{n_{entries}} \frac{|y_i - y_i^*|}{y_i} + \frac{NumRules}{Num_{MAX}} * \omega_r \quad (4)$$

where $NumRules$ is the number of fuzzy rules of the model, Num_{MAX} is the maximum number of rules allowed to one chromosome and ω_r is an assigned weight. Num_{MAX} was arbitrarily set to 50 and ω_r was set to 0.1 after some parameter adjusting experiments.

The newly introduced term in PI' increases the pay-off of the PBGA to models with fewer rules. Simulations were performed under the same conditions of the first session.

TABLE III
ERROR OF THE BUILT MODELS USING PI'(AVERAGE OF 100 RUNS)

	80 entries	160 entries
PBGA	29.95%	29.88%
PBGA+Adap.Op	9.88%	12.52%

Table III shows that when using PI' the difference in the performance between the models built by the PBGA and the models built by the PBGA with the adaptive operator was enlarged.

TABLE IV
FUZZY RULES OF THE BUILT MODELS USING PI' (AVERAGE OF 100 RUNS)

	Number of Rules	Non-used Rules
PBGA	13	33.16%
PBGA+Adap.Op	13	14.02%

Table IV shows that the number of rules of the built models was drastically reduced with the performance index PI'. Moreover, the percentage of non-used rules of the fuzzy models decreased considerably for both cases, but more to the case where the adaptive operator was present.

Comparing the data on tables I and III, it is possible to confirm that the number of rules of the model and the quality of the built models comprise a balance problem. This is an intuitive idea: using more rules it is possible to build models with lower errors and vice-versa. In the extreme case, this would mean to build one rule for each one of the entries in the training data set, which would lead to a zero-error model. However, this model suffers from a lack of generality. The precision of the model is desirable to be achieved with less number of rules.

The simulation results show the effects of the adaptive operator. For both the cases, PBGA with and without the adaptive operator, the number of rules diminishes considerably. However, the decrease in the number of rules for the models built by the PBGA is followed by a considerable increase in the error of the models but the same not happen for the case where the PBGA with the adaptive operator is used. This shows the benefit brought by the adaptive operator for building effective fuzzy rules. The combination of PBGA with adaptive operator and PI' also brings the best percentage of non-used rules in the obtained models, which is a confirmation of the action of the adaptive operator.

VII. Conclusions

This paper presented a new adaptive operator for determining the division points of chromosomes for bacterial mutation and the cutting points for crossover. The effects of the new operator introduced into a Pseudo-Bacterial Genetic Algorithm were shown in experiments building a fuzzy model of non-linear equation.

The adaptive operator had an effective action for increasing the percentage of active rules in the built models. This led to more compact fuzzy models with better quality rules. Operators like the one presented here should be used to combine forces with the dynamics of GA type algorithms in order to build better quality models. GAs are basically fitness function-oriented search methods, therefore, blind for any other aspect that is not explicitly considered on the fitness function. Consequently additional mechanisms are necessary in order to construct not only good fuzzy models in terms of performance but also compact models and with optimized rules.

References

[1] Takeshi Furuhashi, Yujiro Miyata, Ken Nakaoka, and Yoshiki Uchikawa, "A new approach to genetic based machine learning for efficient finding of fuzzy rules," in *Lecture Notes in Artificial Intelligence*, vol. 1011, pp. 173–189. Springer-Verlag, 1995.

[2] Lotfi Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[3] T. Terano, K. Asai, and M. Sugeno, *Fuzzy systems theory and its applications*, Academic Press, 1992.

[4] C.C. Lee, "Fuzzy logic in control - fuzzy logic controller - parts i & ii," *IEEE Transactions on Systems, Man and Cybernetics*, pp. 404–435, 1992.

[5] Shin ichi Horikawa, Takeshi Furuhashi, and Yoshiki Uchikawa, "On fuzzy modeling using fuzy neural networks with the backpropagation algorithm," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 801–806, 1992.

[6] 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.

[7] M.A. Lee and R. Saloman, "Hybrid evolutionary algorithms for fuzzy system design," in *Proceedings of IFSA'95*, 1995, vol. 1, pp. 269–272.

[8] C.L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proceedings of the 4th International Conference on Genetic Algorithms*, 1991, pp. 450–457.

[9] F. Hoffmann and G. Pfister, "A new learning method for the design of hierarchical fuzzy controllers using messy genetic algorithms," in *Proceedings of IFSA'95*, 1995, vol. 1, pp. 249–252.

[10] Tomonori Hashiyama, Takeshi Furuhashi, and Yoshiki Uchikawa, "A study on finding fuzzy rules for semi-active suspension controllers with genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation 95*, 1995.