A Study on the Discovery of Relevant Fuzzy Rules Using Pseudo-Bacterial Genetic Algorithm

Norberto Eiji Nawa, Takeshi Furuhashi, Tomonori Hashiyama, Yoshiki Uchikawa
Laboratory of Bio-Electronics, Department of Information Electronics
School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN
Tel.+81-52-789-2793, Fax.+81-52-789-3166
e-mail: {eiji,furu}@bioele.nuee.nagoya-u.ac.jp

Abstract— **This paper presents a new method for the discovery of relevant fuzzy rules using the Pseudo-Bacterial Genetic Algorithm (PBGA). The PBGA was proposed by the authors as a new approach combining a genetic algorithm with a local improvement mechanism inspired by a process in bacterial genetics, named *bacterial operation*. The presented system aims the improvement of the quality of the generated fuzzy rules, producing blocks of effective rules and more compact rule bases. This is achieved by encoding the fuzzy rules in the chromosomes in a suitable form in order to make the *bacterial operation* more effective and by using a crossover operation that adaptively decides the cutting points according to the distribution of degrees of truth values of the rules. In this paper, first, results obtained when using the PBGA for a simple fuzzy modeling problem are presented and compared with other methods. Second, the PBGA is used in the design of a fuzzy logic controller for a semi-active suspension system. The results show the benefits obtained with this approach in both of the studied cases.**

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

I. Introduction

FUZZY systems can represent non-linear systems using linguistic variables in a straightforward way when enough knowledge about the object system is available. However, when the information about the object system is incomplete or can not be easily described by an human expert, alternative methods have to be used in order for one to build a fuzzy system. 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) [2] implements a local improvement mechanism based on the genetic recombination of bacterial genetics. This operation aims the improvement of local portions of the chromosomes. The chromosomes are divided into several parts and each one of them is improved by the bacterial mutation of the PBGA. In previous works [2] the PBGA was applied for the discovery of fuzzy rules.

As stated above, the key concept of the PBGA is to improve *parts* of chromosomes. When the parameters of a fuzzy system are encoded into a chromosome, a part of chromosome can be a group of contiguous fuzzy rules or a single fuzzy rule. In both cases, the basic idea that a *part* is improved by the *bacterial operation* remains the same. The difference is that the operation will happen in different hierarchical levels of representation.

This paper presents an encoding method of fuzzy rules that facilitates the task of discovering relevant fuzzy rules by the PBGA. Moreover, the cutting points for the crossover operator are decided according to the degrees of truth values of the fuzzy rules, instead of determining them completely at random. The rationale of this special crossover operator is to induce the production of blocks of effective rules, leading to the discovery of sets of fuzzy rules with fewer irrelevant rules, i.e. unused parts of chromosomes. The motivation of the PBGA with this encoding method and the special crossover operator on the discovery of relevant fuzzy rules relies on the following premise: Genetic Algorithms (GAs) and their variants, such as the PBGA, are fitness function driven optimization methods. As long as the fitness function is improved, GA-type algorithms are not concerned with the structural complexity of the evolved fuzzy systems. Therefore, additional features have to be introduced in order to minimize the structural complexity of the evolved fuzzy systems and facilitate the task of finding relevant fuzzy rules by the GA-type algorithms. Moreover, with fewer and more useful rules, knowledge can possibly be acquired through analysis of the discovered rules. The obtained results show the benefits of this approach in this sense.

This paper is organized as follows: section II briefly describes the basis of fuzzy logic and the inference method based on fuzzy rules; section III explains the ideas underlining the hybrid fuzzy-GA systems. Section IV describes the Pseudo-Bacterial Genetic Algorithm, proposed by the authors as a GA with a local improvement mechanism and section V introduces the encoding method which enables the efficient action of the PBGA. In section VI, the results obtained for the experiments realized on the modeling of a non-linear equation and in the design of a fuzzy controller for a semi-active suspension system are presented and section VII brings the final conclusions.

II. Fuzzy Logic

Fuzzy Logic (FL) was proposed by Zadeh [14] in the 60's as a superset of the traditional bi-valued logic. In opposition to traditional logic, where the propositions can only assume two values, either "false" (0) or "true" (1), FL can deal with several 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 text on FL, please refer to [13].

The heart of a fuzzy system is the inference method, which is built on the principles of FL. The inference method is based on a set of IF-THEN rules and membership functions of the variables of the systems. As an illustration, say

that a FLC is used to control a valve of a reactor where a chemical process occurs. The input variables of the FLC are the pressure and temperature inside the chemical reactor. For example, one rule of the FLC 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 labels of membership functions. Membership functions divide the universe of discourse of the variables in several sections and usually have attached to them linguistic labels that provide the meaning of the section, such as {`LOW`, `MEDIUM`, `HIGH`} or {`SMALL`, `MEDIUM`, `BIG`}. Fuzzy inference systems have several rules like the one described above. Real input values are applied to the antecedent part of the rules (*fuzzification*), the degrees of truth of the antecedents are calculated and applied to the consequent of the rules (*inference*), the obtained outputs of the fired rules are merged (*composition*) and, finally, the result is returned to the crisp domain (*defuzzification*), in order to enable a precise actuation in the real system. Fuzzy systems can process uncertain linguistic information in a systematic way. This characteristic enable their use in many different fields. For a introductory article on fuzzy inference systems applied to control problems, refer to [10].

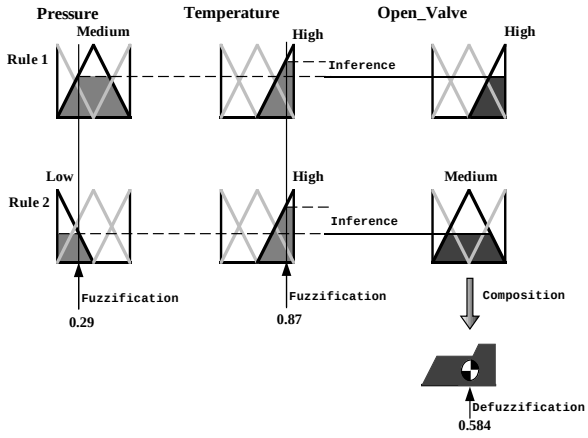

Fig. 1. Overview of the fuzzy inference process

III. Fuzzy Models and Genetic Algorithm

As illustrated before, fuzzy inference uses IF-THEN type linguistic rules that can deal with vagueness of concepts, incorporate knowledge of experts and cope with contradictions. These characteristics are especially useful for modeling systems for which there is no analytical description available but their behaviors can be described by an human expert.

The design process of a fuzzy model basically 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 time consuming process of adjustments of the fuzzy model parameters based on *trial-and-error* is necessary. In worse cases, where a human expert is not 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)[2], [6], [9], [11].

GA has been used combined with FL in the design of Fuzzy Logic Controllers (FLCs) [8], [4]. 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, which greatly relieves the burden of the human designer.

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 2):

1. *Generation of the initial population*: n chromosomes are created and evaluated;
2. *Genetic Operations*:
   (a) *Bacterial operation*: This genetic operation is applied to each chromosome one by one. Suppose there

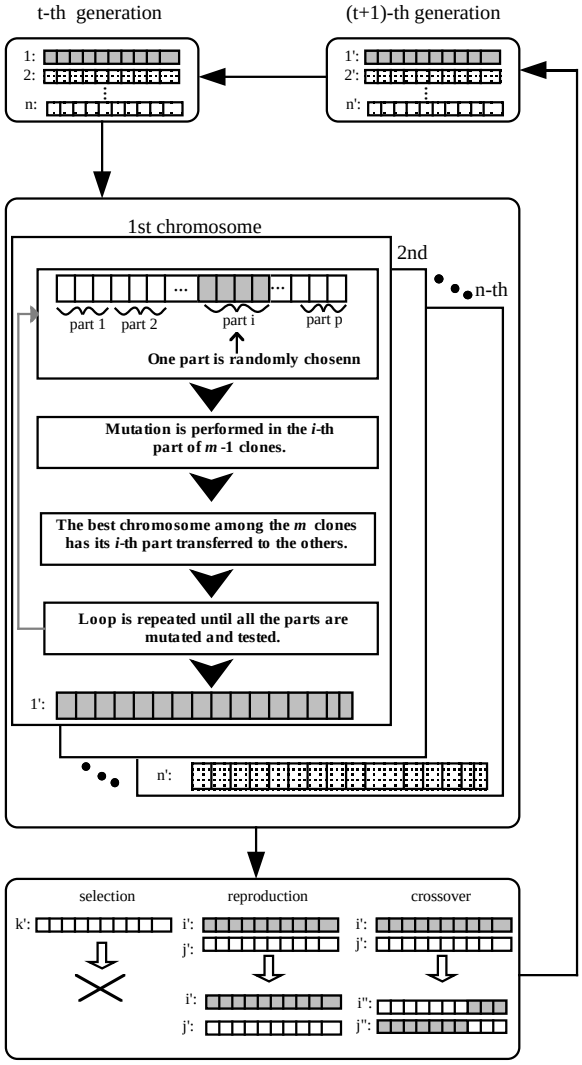

Fig. 2. Scheme of the PBGA

are p parts in a chromosome. The first chromosome is chosen and m clones are produced from it. The i-th part (randomly chosen) of m-1 clones is mutated. The elite among the m chromosomes is selected and the i-th part of the selected chromosome is transferred to the m-1 chromosomes. On this stage, the i-th part of all the clones is replaced by the i-th part of the selected chromosome. This process, mutation-evaluation-selection-replacement, is repeated. The mutation is applied to another randomly chosen part, different from the already chosen ones. When all the p parts have been tested, one chromosome from the m clones is selected to remain in the population and the other m-1 clones are deleted.

This genetic operation is applied to all the n chromosomes in the population.

(b) *Conventional genetic operations*: The chromosomes with lower fitness values are deleted and some randomly chosen chromosomes from the remaining group are reproduced. Chromosomes are mated and offsprings are generated by crossover.

3. *Stop condition*: Stop if the condition is satisfied, otherwise go back to 2.

C. Special Crossover Operator

In order to improve the performance of fuzzy system identification by the PBGA, a special crossover operator that makes use of problem domain-specific knowledge is used. Contrarily to the ordinary crossover operator that randomly decides the chromosomes cutting points, a different criteria is adopted when using the PBGA to define the parameters of a fuzzy system.

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, it indicates the intensity that the consequent of the rule will have in the overall fuzzy system output.

The special crossover operator takes into consideration the moving average of the degrees of truth values of the fuzzy rules when deciding where to cut the chromosome. 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 evaluation process. Consequently, this is an indication of the utility and possible effectiveness 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 in the chromosome that contain rules which are actually used by the system and produce blocks of these rules by sectioning the chromosomes where there is a concentration of ineffective rules. Those strings having high moving averages can be considered to have a high concentration of useful rules, since they are actually used by the system. These strings are desirable to be preserved. On the other hand, blocks

with low moving averages have higher probabilities of being mutated. This selection of crossover points works well to build blocks of fuzzy rules with high degree of utility. By minimizing the number of rules that are not used by the system it is possible to construct more compact models. Figure 3 shows a schematic of the rationale of using the moving average of the truth value of the fuzzy rules.

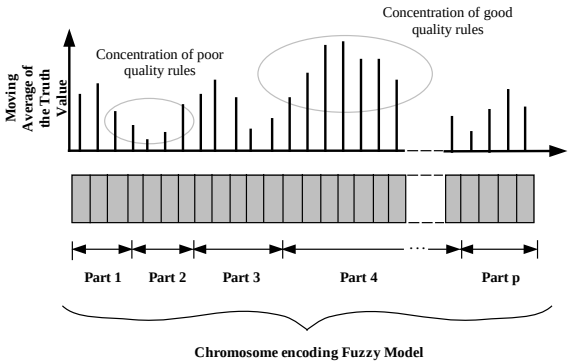

Fig. 3. Fuzzy rule truth value as a quality measure

V. Encoding Method

Encoding is an issue of fundamental importance when using genetic algorithms. In this approach, instead of using the canonical binary encoding, the real parameters of the fuzzy systems are put into the chromosomes. Each chromosome in the population encodes the rules of the rule base of the fuzzy model and the membership functions of the system variables. The chromosome lengths are variable, which means that the number of rules is also one of the parameters to be optimized by the genetic algorithm. Each of the rules 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 triangular, so their parameters are the pairs *(center, width)*. For example, the Rule 4 in Figure 4 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 located at `c` with width `w`. `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. For the bacterial operation, the parameters of a single rule or a group of contiguous rules can be considered to be a part to be optimized.

This encoding method, different from the conventional fuzzy systems, does not unify the membership functions of the variables. In other words, the membership functions of the variables in one rule are not shared with other rules. This freedom is preferred than the use of partitions determined in advance. The definition of the membership functions and the fuzzy rules are highly coupled and both determine the overall performance of the system. Therefore, it is desirable to simultaneously evolve the rules and the membership functions, minimizing the probability of arriving to a sub-optimal point due to an imperfect choice of partitions, as pointed in [5]. This encoding method increases the total number of membership functions in the system but this can be compensated by the achievement of better performances by the system.

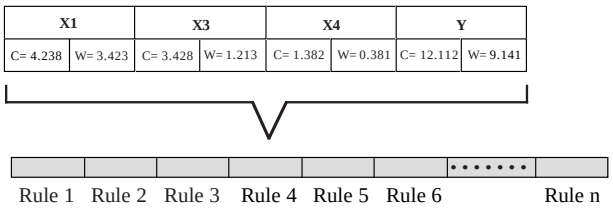

Fig. 4. Example of a Fuzzy Model encoded in a Chromosome

VI. Experiments and simulation results

In order to test the efficiency of this approach, numerical experiments were performed. The experiment was to build a fuzzy model of the following non-linear function:

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

with the variables defined in the following intervals: $x_1 \epsilon [1 \ldots 5]$, $x_2 \epsilon [1 \ldots 5]$, $x_3 \epsilon [0 \ldots 4]$, $x_4 \epsilon [0.0 \ldots 0.6]$, $x_5 \epsilon [0.0 \ldots 1.0]$ and $x_6 \epsilon [0.0 \ldots 1.2]$.

The function $y(x_1, x_2, x_3, x_4, x_5, x_6)$ has no special meaning, being just a test bed function for testing our approach. It was modeled on 7 fuzzy variables: 6 of them were the original variables, the 7^{th} was a dummy variable. The role of the dummy variable is to increase the level of difficulty of the modeling task, by adding one more dimension to the built model that nevertheless has to behave like the original function. 80 randomly chosen 7-tuples were used as training data and another 80 entries were used as test data. The population was fixed with 30 chromosomes during the whole search process. The number of clones produced in the bacterial operation was arbitrarily fixed to 4 clones. The crossover operator generates half of the new individuals; the PBGA used the special crossover operator. Tournament selection was used to complete the other half of the new population. Single rules were considered to be the parts to be improved by the bacterial operation.

The simulations were performed using the following per-

formance index (PI), which should be minimized:

$$PI = \frac{1}{n_{entries}} \sum_{i=1}^{n_{entries}} \frac{|y_i - y_i^*|}{y_i} + \frac{NumRules}{Num_{MAX}} * \omega_r \quad (3)$$

where y_i is the desired value of the model, y_i^* is the value inferred by the fuzzy model, $n_{entries}$ is the number of data entries, $NumRules$ 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. ω_r was set to 0.1 after some parameter adjusting experiments and Num_{MAX} was arbitrarily set to 50. Considering that the training data has 80 entries, the Num_{MAX} represents a minimum requirement of quality concerning the number of rules.

The first term of PI is the average error of the evaluated fuzzy model, indicating how good the estimations of the model are. The second term increases the pay-off to models with fewer rules, in order to subtly induce the PBGA to build compact models.

Comparative experiments were performed in order to test the suitability and effectiveness of the PBGA. Two algorithms besides the PBGA were used in the experiments: a GA with a radical local improvement mechanism (RLIM-GA) and a GA with a hill-climbing (HCGA).

The GA with a radical local improvement mechanism (RLIM-GA) has a similar dynamics to the PBGA; it uses the same encoding in their chromosomes and the traditional genetic operators are identical, including the special crossover operator. The PBGA and the RLIM-GA differ in the hierarchical level of representation where the bacterial operation is applied. The latter applies the bacterial operation to each one of the loci of each one of the fuzzy rules in the chromosome. It tries to radically improve each one of the parameters of the fuzzy rules.

The GA with a hill-climbing method (HCGA), as its name says, is a mixture of a GA with a hill-climbing method. The HCGA incorporates in the GA a *iterated hill-climbing* phase, as defined in [12], where the new configuration is not randomly constructed but obtained through mutation. The mutation that generates new configurations randomly modifies parameters from a random number of rules. In fact, the HCGA has the mutation operator incorporated in the the hill-climbing method. The special crossover operator was also utilized in the HCGA.

For all the cases, the number of evaluations is approximately the same, around 30000 evaluations per run.

The obtained results concerning the errors of the built models are shown in Table I. The errors are averages of 50 runs for each of the methods.

From the data in Table I, it can be clearly seen that the PBGA can design better models than the other two methods. The column entitled "train set" contains the value of the error of the models when they are tested to the 80 entries of the training data. The column entitled "test set" contains the value of the error of the models when they are tested to the 80 entries of the test data set.

The bacterial operation was used in both the RLIM-GA and the PBGA. However, the level of representation where the bacterial operation actuates is different: in the RLIM-GA the bacterial operation is applied in the rule parameters level, i.e. the parameters that define each one of the fuzzy rules are tried to be optimized. The bacterial operation considers each one of the parameters as the part to be optimized. In the case of the PBGA, the bacterial operation is applied in the rule level; the rule is "seen" as the entity to be optimized; the definition of "part" goes up one level. The difference is subtle, but it is visible from the results in Table I. Also, the natural expectation that the RLIM-GA approach would suffer the illness of overtraining to the train data is confirmed: the results of the RLIM-GA degrade the most when the test data is used.

TABLE I
ERROR OF THE BUILT MODELS(AVERAGE OF 50 RUNS)

	train set	test set
HCGA	23.96%	24.17%
RLIM-GA	13.61%	23.62%
PBGA	7.50%	13.41%

Table I confirms that the choice of the representation level that the local improvement mechanism will actuate is critical. For instance, the hill-climbing method of the HCGA can be considered to be actuating in the entire fuzzy system at once. The choice of locally optimizing in the fuzzy rule level of representation by the PBGA brought the best results. This was possible in this case because the hierarchical levels are very clear within the chromosome.

TABLE II
FUZZY RULES OF THE IDENTIFIED MODELS(AVERAGE OF 50 RUNS)

	Number of Rules	Non-used Rules
HCGA	17.82	43.78%
RLIM-GA	30.34	17.02%
PBGA	18.34	26.57%

The column "Number of Rules" of Table II shows the average of the number of rules obtained by the models constructed by each one of the methods. The column "Non-used Rules" shows the average of fuzzy rules that do not fire when the test data is applied. From Table II it can be seen that the HCGA and the PBGA develop, in average, models with approximately the same number of rules. However, recalling from Table I, the models built by the PBGA perform considerably better than the ones built by the HCGA. Moreover, the percentage of non-used rules, i.e. rules that are not fired at all when the test data is applied, is the best when the RLIM-GA is used. This is an interesting phenomenon: from data in Table I, the results of the RLIM-GA degrade the most in terms of error for the test data. The rules generated by the RLIM-GA fire more often when unknown data is applied, but it does not imply that the rules are meaningful or effectively contribute to

the inferred output value of the model. This can be clearly seen when comparing the performances of both RLIM-GA and PBGA from Table I, where the performance of models built by the PBGA outperforms the ones built by the RLIM-GA.

From the data in Tables I and II, one plausible explanation can be drawn. The performance index that evaluates the chromosomes has one term that measures the performance of the built models and another term that takes in consideration the number of rules in the model. This is based on the idea that the models should not only be good concerning the errors of the outputs but also parsimonious concerning the number of rules. As the errors of the models are considered to be more important than the parsimony in the number of rules, a weight adjusts the trade off between the two terms. The data in tables I and II show the efficiency of the methods in fulfilling both goals. Although the HCGA and the PBGA achieve similar number of rules, the PBGA is more efficient in building better models. On the other hand, the RLIM-GA build models that perform better than the HCGA in terms of the error of the models but face some difficulty to build parsimonious models in terms of number of rules. Among the three methods, the PBGA is the one that best fulfills the two goals expressed in the performance index. This can be considered an effect that emerges from the combination of the characteristics of the algorithm.

TABLE III
STANDARD DEVIATIONS OF THE MODELS (AVERAGE OF 50 RUNS)

	Error		Number of
	train	test	rules
HCGA	0.070	0.060	11.93
RLIM-GA	0.085	0.111	11.69
PBGA	0.011	0.017	6.56

Table III shows the standard deviation of the data obtained in the simulation results. It can be seen that the variance of the results is less when using the PBGA.

Experiments were also performed with the PBGA with common single-point crossover, where the cutting point is randomly decided. Simulation conditions were exactly the same as the previous experiments. Table IV shows the obtained results for the error of the models and Table V shows the results about the obtained rules. From Table IV it is possible to see that the PBGA with common crossover operator outperforms the HCGA and RLIM-GA, but does not perform better than the PBGA with the special crossover operator. This is a strong indication of the effect of taking the degree of truth value in consideration, as a criterion for deciding the cutting points of chromosomes. The difference becomes more dramatic when comparing the results of the test data. The models built by the PBGA with common crossover degrade more than all the other methods, but surprisingly, the percentage of rules that do not fire when the test data is applied is very low. This apparent contradiction shows that although most of the fuzzy rules fire when the test data is applied, this has no direct relation with the "quality" of the output value of the model. The number of rules of the models built by the PBGA with the common crossover exceeds the results obtained by the other methods. Apparently, the special crossover plays an important role when building parsimonious models.

TABLE IV
ERROR OF THE MODELS BUILT BY THE PBGA WITH COMMON CROSSOVER (AVERAGE OF 50 RUNS)

train set	test set
9.93%	37.99%

TABLE V
FUZZY RULES OF THE MODELS IDENTIFIED BY THE PBGA WITH COMMON CROSSOVER (AVERAGE OF 50 RUNS)

Number of Rules	Non-used Rules
33.94	8.11%

A. Fuzzy Logic Controller for Semi-active Suspension System

A.1 Semi-active Suspension System

Semi-active suspension systems aim to improve the ride comfort and the drivability of cars, in the same way as active control systems. However, semi-active systems are simpler, more reliable and considerably cheaper when compared to active systems.

The basic principle of the semi-active control systems is to actuate in the damping coefficient of the shock absorber(damper), according to the current ride conditions. This is achieved by controlling the electro-magnetic valve inside the absorber. The seminal work in the field of semi-active suspension systems was proposed by Karnopp [7] and its objective was only to reduce the vertical acceleration of the body of the car.

The system uses a PBGA to build FLCs for a semi-active suspension system. Figure 5 shows the simulation model of the semi-active suspension system. K is the stiffness of the suspension and C_d is the damping coefficient of the shock absorber.

The dynamics of the simulation process is as follows. Each chromosome in the population encodes a FLC. All the chromosomes are tested in the model of the suspension system and receives a fitness value from the *Performance Evaluator*, which gives a grade according to the performance index that is being used. Basically, the controllers are supposed to minimize some physical variables of the system, for example, *Body Vertical Position, Body Vertical Velocity, Body Vertical Acceleration, Tire Vertical Position, Tire Vertical Velocity and Tire Vertical Acceleration.* This process is repeated within the dynamics of the PBGA.

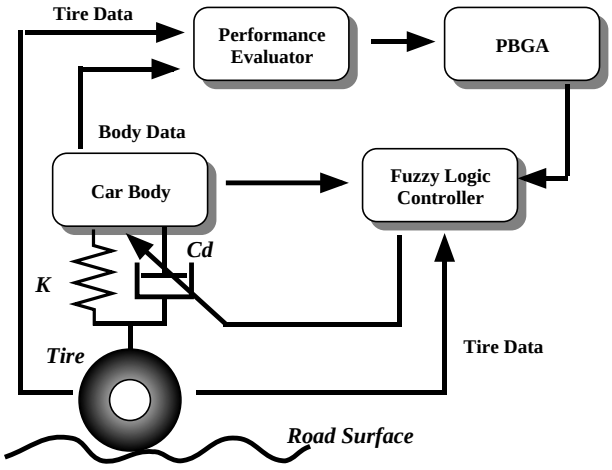

Fig. 5. Semi-active suspension model

One point that should be remarked is that the fuzzy controllers generated possess a multi-objective character, since the performance index was set to minimize multi-objective functions.

A.2 Simulation Results

In this experiment, a PBGA variant was used and its performance compared to other methods. The variant includes a new operator, called *adaptive operator*. This operator defines the parts of the chromosome where the bacterial operation will be performed. In the fuzzy modeling experiment, the parts were the fuzzy rules themselves. In this case, the parts are groups of contiguous rules. This was done to cope with the long simulation times involved, that disabled any tentative of applying the bacterial operation in each fuzzy rule of the chromosomes.

The adaptive operator determines the division points of each chromosome according to the moving average of the degree of truth value of the fuzzy rules. Similarly to the special crossover operator, the probability of a certain rule to be selected as a frontier point between two parts is inversely proportional to the moving average of the degrees of truth values of the fuzzy rule. The number of division points determines the intensity of the bacterial operation in the genes of the chromosomes, in this case, the parameters of the fuzzy rules. In the experiments here described, the number of division points was fixed during the whole process.

Simulations were performed using PBGA with adaptive operator and simple GA. The first approach used a simple GA (as defined in [3]) and the fuzzy controllers encoded in the chromosomes used *fixed membership functions*, defined *a priori*. Every input and output variables had its universe of discourse divided 7 partitions, usually named as *Positive Big, Positive Medium* and so on. The task of the simple GA was basically to define the rule base of the FLCs over the predefined membership functions. 100 chromosomes constituted the population and the process ran through 100 generations. The performance index (PI) used was:

$$PI = peak(Pos_{car}) + peak(Vel_{car}) + peak(Acc_{car}) \quad (4)$$

The system tries to minimize the peak values of the parameters of the car body (Vertical Position, Velocity and Acceleration). In doing so, expectedly higher levels of ride comfort would be achieved. No term concerning the number of rules of the fuzzy controller was introduced in the performance index this time. The system showed to be very sensitive to the introduction of terms not directly related to the performance of the controller. Finding the proper setting of multi-objective fitness functions when using evolutionary computation methods is a critical problem. Refer to [1] for a more detailed text on this issue.

The average of the best FLC in the population in 10 runs is shown in Figure 6, under the name of *Simple GA with fixed MF*.

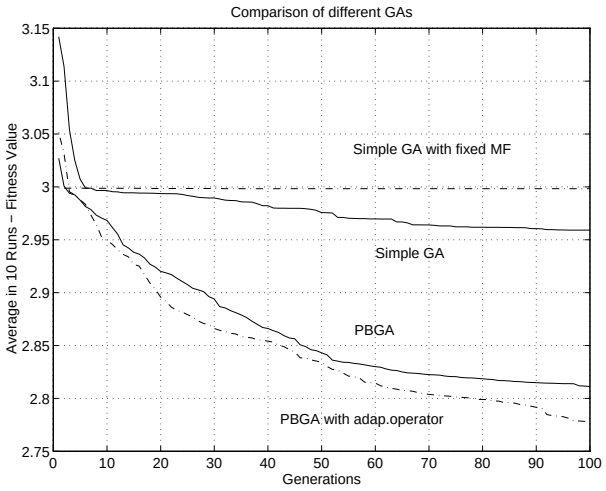

Fig. 6. Obtained performances using different GAs

The improvements obtained by that approach for this system were very small. The vertical axis on the graph in Figure 6 represents the fitness value utilized. Fitness value equal to 3 can be understood as the same performance of the controller designed by Karnopp in [7]. The first approach was not able to evolve FLCs that differ considerably from controllers presented in [7].

The second approach also used a simple GA. However, the membership functions of the variables in the FLCs were not defined *a priori*. The chromosomes in this second approach used the same encoding method presented in section V. It had the possibility of defining the membership functions and rule base of FLCs simultaneously. The obtained performance is the curve over the label *Simple GA*.

The third approach used the PBGA, with an ordinary crossover operator and with chromosomes equally divided in 3 parts. It is possible to see that the performance is significantly better than the two approaches presented above. The last approach was the PBGA with the adaptive operator dividing the chromosomes in 3 parts too (and special

crossover operator), that achieved the best performance among all.

TABLE VI
BEST AND WORST OBTAINED RESULTS (PI) IN 10 RUNS

	Best	Worst
PBGA	2.75	2.89
PBGA+Adap.Op	2.67	2.89

Table IV shows the best and worst results obtained in 10 runs by the PBGA and the PBGA with the adaptive operator. From Figure 6 it can be seen that the fuzzy controllers built by the PBGA perform better than the Karnopp controller and the fuzzy controllers derived by the simple GA. Moreover, although the difference between the performances of the PBGA and the PBGA with the adaptive operator apparently is not dramatic, the PBGA with the adaptive operator worked well to find out better rules under these simulation conditions with limited room for improvement.

Figure 7 shows an example of an evolved FLC for the semi-active suspension system. The six input variables are, respectively from the left, the car body vertical position and velocity, tire vertical position and velocity, car body vertical acceleration and tire vertical acceleration. The rightmost variable is the output variable, namely the damping coefficient of the shock absorber. By analyzing the obtained rules, the formidable possibility of acquiring knowledge of an unknown system is opened. For example, the first rule in the top row can be read as IF $T_{velocity}$ is Small and $T_{acceleration}$ is Positive Big THEN $DampCoef$ is Small, where $T_{velocity}$ is the tire vertical velocity, $T_{acceleration}$ is the tire vertical acceleration and $DampCoef$ is the damping coefficient of the shock absorber.

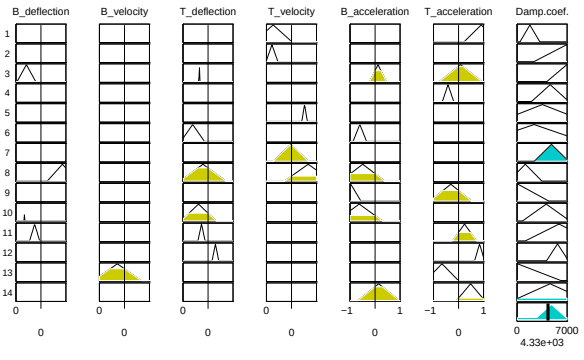

Fig. 7. Example of a FLC obtained by the PBGA

VII. CONCLUSIONS

This paper presented a Pseudo-Bacterial Genetic Algorithm (PBGA) performing the task of designing fuzzy rule bases. The PBGA implements a local improvement mechanism based on a process that occurs in the bacterial genetics level. The aim of this approach is to produce blocks of relevant and effective fuzzy rules. The importance of discovering relevant fuzzy rules for a knowledge domain with no experts is that it opens the possibility of knowledge discovery.

The performance of the PBGA was shown in experiments for building a fuzzy model of non-linear equation and a fuzzy controller for a semi-active suspension system. When designing fuzzy systems, the PBGA makes use of a special crossover operator that takes into account the degrees of truth values of the fuzzy rules when deciding the cutting points. Additionally, the effect of considering the degrees of truth values of the fuzzy rules when adaptively determining the parts for the bacterial operation was examined.

The results show that the strategy adopted by the PBGA of improving parts of chromosomes using the bacterial operation is beneficial. The encoding method used for the chromosomes was highly suitable for the bacterial operation to work. Also, the results show that the choice of the level of representation where the bacterial operation will actuate is critical for the performance of the system. The PBGA was able to design fuzzy systems that ally good performance with fewer rules, and it outperformed the other methods concerning the error, in the case of the fuzzy model experiments, and the performance index in the case of the design of a semi-active suspension controller.

REFERENCES

[1] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part i: A unified formulation. *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):26–37, 1998.

[2] T. Furuhashi, Y. Miyata, K. Nakaoka, and Y. Uchikawa. A new approach to genetic based machine learning for efficient finding of fuzzy rules. In *Lecture Notes in Artificial Intelligence*, volume 1011, pages 173–189. Springer-Verlag, 1995.

[3] D. E. Goldberg. *Genetic Algorithms in seach, optimization, and machine learning*. Addison-Wesley, 1989.

[4] 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*, volume 1, pages 249–252, 1995.

[5] A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2):129–139, 1995.

[6] S. Horikawa, T. Furuhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, 3(3):801–806, 1992.

[7] D. Karnopp, M. J. Crosby, and R. A. Harwood. Vibration control using semi-active force generators. *ASME Journal of Engineering for Industry*, 96(2):619–626, 1974.

[8] C. L. Karr. Design of an adaptive fuzzy logic controller using a genetic algorithm. In *Proceedings of the* 4^{th} *Int. Conference on Genetic Algorithms*, pages 450–457, 1991.

[9] 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*, 1196:274–288, 1989.

[10] C. C. Lee. Fuzzy logic in control - fuzzy logic controller - part i & ii. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–435, 1990.

[11] M. A. Lee and R. Saloman. Hybrid evolutionary algorithms for fuzzy system design. In *Proceedings of IFSA '95*, volume 1, pages 269–272, 1995.

[12] H. Muhlenbein. Evolution in time and space - the parallel genetic algorithm. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan Kaufmann, 1991.

[13] T. Terano, K. Asai, and M. Sugeno. *Fuzzy Systems Theory and its Applications*. Academic Press, 1992.

[14] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–35, 1965.