Mathware & Soft Computing 5 (1998) 175-187

Refinement of a Fuzzy Control Rule Set*

Antonio González and Raúl Pérez
Depto. Ciencias Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática,
Universidad de Granada, 18071 Granada (Spain)
e-mail: {A.Gonzalez,fgr}@decsai.ugr.es

Abstract

Fuzzy logic controller performance depends on the fuzzy control rule set. This set can be obtained either by an expert or from a learning algorithm through a set of examples. Recently, we have developed SLAVE an inductive learning algorithm capable of identifying fuzzy systems. The refinement of the rules proposed by SLAVE (or by an expert) can be very important in order to improve the accuracy of the model and in order to simplify the description of the system. The refinement algorithm is based on an heuristic process of generalization, specification, addition and elimination of rules.
Keywords: theory refinement, fuzzy logic, machine learning, system modeling.

1 Introduction

One of the most successful applications of fuzzy logic has been the fuzzy control [1, 12]. A fuzzy logic controller is a rule-based system where the rules describe the experience of a skilled operator or the knowledge of control engineers. The fuzzy logic controller consists of the following basic components [9]:

- A *Knowledge Base* comprising of linguistic control rules about the controlled system. The Knowledge Base consists of two components:
  - a *Data Base* containing the definitions of the linguistic values of the variables,
  - a *Rule Base* containing the fuzzy control rules,
- a *Fuzzification Interface* which has the effect of transforming crisp values into fuzzy sets,
- an *Inference System* that uses a reasoning method,

*This work has been supported by CICYT under Project TIC95-0453

- a *Defuzzification Interface* that translates a fuzzy control action to a real control action.

The design of a fuzzy controller implies different tasks. One of the most important tasks is the obtaining of the fuzzy control rule set. A method for extracting this knowledge consists of directly extracting the expert experience from the human processes operator. However, in many cases it is not easy to express this knowledge in terms of appropriate fuzzy rules. Thus, automatic learning methods can be used as an alternative approach [5, 10, 11, 13]. In both cases of either the expert or the learning algorithm, the final set of rules can be improved by simplifying them or avoiding some possible errors. In this case, a refinement of the final set of rules can produce a high performance system. Therefore, given an incomplete and/or incorrect fuzzy control rule set and a set of consistent examples, the problem consists of modifying the initial set of rules so that it may be better adapted to the example set.

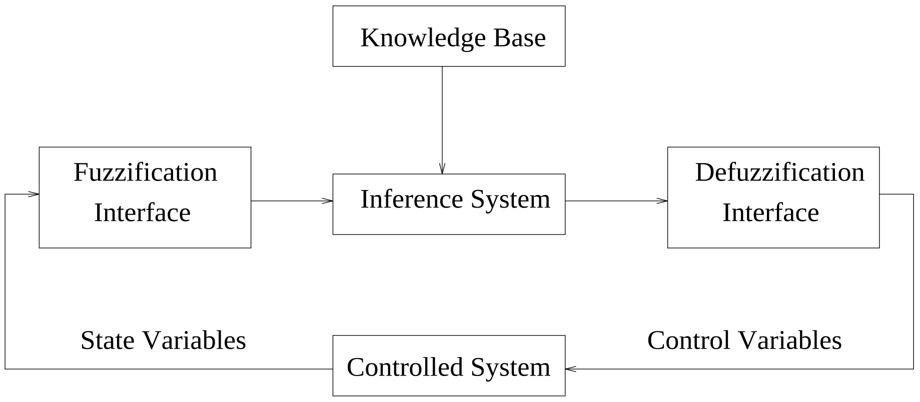


Figure 1: Fuzzy Logic Controller

In [2, 3, 4] we proposed a learning algorithm of fuzzy rules called SLAVE. This algorithm can be used for general system modeling, and therefore can be used for learning fuzzy control rules. SLAVE uses a fixed semantic for the value of the linguistic variables and a genetic algorithm to obtain the best rule in each step based on an iterative approach [8]. This genetic iterative approach produces good individual rules, and a revision of the complete set of rules may be advisable. In this case, the set of fuzzy control rules obtained by SLAVE could be partially incomplete and/or incorrect, in the same way as the expert rules can be, and the refinement algorithm can be very suitable for improving the final set of rules. Therefore, the main goal of this work is to develop a refinement algorithm for fuzzy rules obtained by either an expert, or SLAVE or any other learning algorithm.

In [6, 7] a refinement algorithm for classification problems was proposed. In many classification problems the consequent variable has a crisp domain and the proposed refinement algorithm was based on this property. In order to extend this refinement algorithm to fuzzy control rules, we need to adapt it so that it can deal with continuous consequent variables discretized by a fuzzy domain.

Many other algorithms try to improve the performance of a fuzzy controller by changing the semantic of the rules; that is, by changing the Data Base or changing the Rule Base or both simultaneously. The proposed refinement algorithm only changes the Rule Base and supposes, like SLAVE, a fixed semantic.

The refinement algorithm is based on an heuristic process of generalization, specification, addition and elimination of rules.

In the next section, we will briefly describe the refinement algorithm for crisp consequents. Section 3 adapts this algorithm for control problems. The behaviour of SLAVE plus the refinement algorithm (SLAVE+R) is analyzed step by step in a detailed example in the next section. Section 5 applies SLAVE+R to a well-known control problem. Finally, we show the conclusion of this work in the last section.

2 The refinement algorithm for crisp consequent

In [6] we proposed a refinement algorithm valid for fuzzy rules with crisp domain for the consequent variable. The extension to fuzzy control rules will be proposed in the next section. Let us now show the main ideas of this previous algorithm.

The refinement algorithm uses an heuristic function and a hill climbing strategy for selecting the most promising action in each step of the algorithm towards a good solution. We have considered it as a function that measures the global precision of the current rule set on the training set. Thus, in order to define this function it is necessary to describe the predictive module. The inference process begins with an ordered rule set and the classification of an example is carried out in the following way: the adaptation between the example and the antecedent part of each rule is evaluated and the class of the rule with the best adaptation is returned. If there are several rules with the best adaptation (conflict problem), the class from the rule which is lowest in the order of the rule set is returned.

Thus, it is necessary to establish a priori criterion of relevance between the rules in order to sort them. The refinement algorithm uses the same order returned by SLAVE. Basically, this criterion is as follows: the most relevant rules are those that removed the highest number of examples in the learning process. In this sense, the most relevant rules are in the first positions and the less relevant rules are in the last positions. The heuristic component of the refinement algorithm selects rules through the order previously described in the rule set. However, we have not considered a special order for variables and values. They are taken by considering the default order.

The rule model that uses the refinement algorithm is the same as that which was used in the SLAVE learning algorithm

$$\text{IF } X_1 \textit{ is } A_1 \textit{ and } \ldots \textit{ and } X_p \textit{ is } A_p \text{ THEN } Y \textit{ is } B$$

where each variable X_i has a referential set U_i and takes values in a finite domain D_i, for $i \in \{1, \ldots, p\}$. The referential set for Y is V and its domain is F. The value of the variable y is B, where $B \in F$ and the value of the variable X_i is A_i, where $A_i \in P(D_i)$ and $P(D_i)$ denote the set of subsets of D_i.

In this model, we can consider, in general, that the variables are linguistic, i.e. the domains of the variable can be described using linguistic labels, or in general fuzzy sets. The key to this rule model is that each variable can take as a value an element or a subset of elements from its domain, i.e. we let the value of a variable be interpreted more as a disjunction of elements than just one element in its domain.

Now, we can better describe each of the following steps of the refinement algorithm:

- Improve the accuracy by specifying the rules.
- Improve the completeness by generalizing the rules.
- Improve the completeness by the addition of new rules.
- Determine the relevant variables for each rule.

The previous steps are repeated until the rule set is stable. In the algorithm, the rule set is considered to be stable when the number of rules and the accuracy do not increase in two consecutive iterations.

2.1 Improving the accuracy by specifying the rules

In this step, we try to improve the capacity of prediction from an incomplete and/or incorrect fuzzy control rule set over the training set, increasing the prediction of each rule.

1. The most relevant rule is selected.
2. Select a variable and do the following.
    - 2.1. Select an active value.
    - 2.2. This value is deactivated.
    - 2.3. If the global accuracy is improved or maintained, the modification is accepted. In all other cases, the value is activated.
    - 2.4. If there are other active values, go to 2.1.
    - 2.5. If all the values of the variable are deactivated, then this rule is removed from the rule set.
    - 2.6. If there are other unselected variables, go to 2.
3. If there are more unselected rules, select the next most relevant rule and go to 2, in all other cases the process finishes.

Furthermore, we must note that this step permits the removal of rules from the rule set. Therefore, the elimination of a rule in the rule set is considered as a particular case of specification and it is caused when all the values of an antecedent variable are deactivated.

2.2 Improving the completeness by generalizing the rules

In this second phase, the refinement algorithm tries to improve the completeness of the rules using an ordered process of generalization. In this process, new values are appended to the antecedent variables for each rule. This generalization is done when the global accuracy strictly improves, using the rule set that includes the modified rule. There can be two reasons for the increase in accuracy:

a) Unclassified examples can be correctly covered by generalizing one of the rules.

b) Examples incorrectly classified by one rule with a lesser order, can be correctly covered when a rule with a higher order is generalized.

Then, this step of the algorithm improves the completeness and the prediction of some rules from the rule set.

The procedure has a similar structure to the previous algorithm:

1. The most relevant rule is selected.
2. Select a variable and do the following.
   - 2.1. Select a inactive value.
   - 2.2. This value is activated.
   - 2.3. If the global accuracy is strictly improved, the modification is accepted. In all other cases, the value is deactivated.
   - 2.4. If there are other inactive values, go to 2.1.
   - 2.5. If there are other unselected variables, go to 2.
3. If there are more unselected rules, select the next most relevant rule and go to 2., in all other cases the process finishes.

2.3 Improving the completeness by the addition of new rules

The next step in the algorithm, consists of appending new rules to cover the examples that are not covered by any other rule and which the previous process cannot cover. This task consists in appending the most specific rule with the best adaptation for each example that is not covered in the training set.

2.4 Determining the relevant variables for each rule

This is the last step in the refinement algorithm and it tries to determine the set of antecedent variables, for each rule, that are needed to describe its class. This task uses a special type of generalization that consists of activating all the values of a variable. If the accuracy of the new rule set is equal to or better than the accuracy of the previous rule set, we can say that this variable is irrelevant for determining its class and the variable can be eliminated as a premise of the rule.

For more details about the behaviour of this algorithm see [6, 7].

3 The refinement algorithm for fuzzy control rules

The refinement algorithm for fuzzy control rules is very similar to the previous one, except for a special characteristic of such a set of rules. Usually fuzzy control rules have a control action which varies in a continuous domain. Therefore, the consequent variable has a continuous referential set and an associated fuzzy domain. The inference process is completely different. The predictive process now depends on the interaction between all the possible rules that are applicable. Therefore, the refinement algorithm uses as an heuristic the decrease in the average number of errors produced in the outputs in which a rule (or set of rules) is used.

Another important aspect of the algorithm is the inference module that determines the error. In this case, we have used a max-min method and a defuzzification based on the average of the center of gravity weighted by the adaptation between the example and the antecedent of the rule. The error is defined as

$$Error = \sum_{i=1}^{n} \frac{(y_i - y_i')^2}{2n}$$

where y_i is the correct output, y_i' is the output of the inference module and n is the number of examples.

The refinement algorithm starts with a set of ordered rules. The order is the same as that which was considered in the crisp case. Thus the structure of the refinement algorithm is as follows:

- Decrease the error by specifying the rules.
- Decrease the error by generalizing the rules.
- Decrease the error by the addition of new rules.
- Determine the relevant variables for each rule.

The previous steps are repeated until the rule set is stable again. As previously established, the rule set is considered to be stable when the number of rules and the accuracy do not increase in two consecutive iterations.

3.1 Decreasing the error by specifying the rules

In this step, we eliminate those values which are irrelevant for the rules. A value is irrelevant if when it is eliminated from the rule, the global error is maintained or decreased. This specification process reduces the number of rules that are applicable to each example. Thus, the specification of the value of a rule means that in the rule base there are other rules whose interpolation effectively decrease the error when the influence of the first rule is decreased. Therefore, in this step, which is similar to 2.1, we achieve two important things:

- to strengthen the most relevant rules,
- and to weaken and even eliminate the less useful rules.

3.2 Decreasing the error by generalizing the rules

This step is the opposite of the previous one. Now, we try to increase the collaboration among rules by a process of generalization. The process is similar to those described in 2.2. We add a new value of a rule whenever the error is strictly decreased.

3.3 Decreasing the error by the addition of new rules

Let us suppose that in the previous step the collaboration among rules cannot be improved by generalizing the rules. In this case, a new rule can reduce the error. The solution is the same as those proposed in 2.3. However, now the process is different. Our goal is to introduce a rule or set of rules that reduces the global error. The idea is to introduce the most specific rule that covers the worst represented example, that is, for the example with the maximum individual error, we introduce the most specific rule covering this example. When there are several examples with the maximum individual error any of them is selected at random. The new rule consists of:

- in the antecedent part: the most specific antecedent covering the example,
- in the consequent part: the value of the domain that gets the least global error when the rule is included in the set of rules.

The underlying idea of this proposal is that the rules that reduce the maximum individual error have the possibility of reducing the global error, too.

This step is repeated until we have an example in which the associated rule does not reduce the global error: in this case, this rule is not introduced in the rule base.

3.4 Determining the relevant variables for each rule

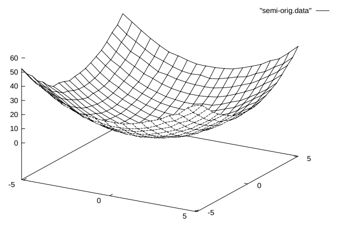
The last step of the refinement algorithm tries to determine the minimum number of variables for each of the rules that are necessary to maintain the current performance. Thus, if a variable is eliminated from the rule and the global error does not increase, we accept this elimination. This step can be interpreted as a special generalization process as in 3.2, but now affecting the complete variables instead of individual values of the variables.

4 A detailed example of the refinement algorithm

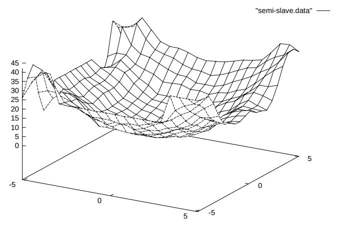
Let's suppose that SLAVE has learned the function

$$h(x,y) = x^2 + y^2$$

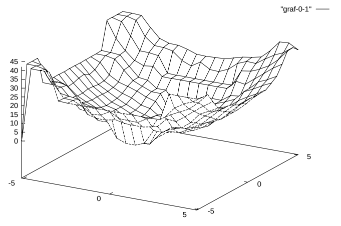
shown in Figure 2(a), randomly extracting a set of examples of this function. Previously, a set of fuzzy labels on its domain was defined for each variable. Each domain consists of seven fuzzy labels with triangular membership functions crossing at height 0.5 and distributed uniformly. The approximation obtained by SLAVE appears in Figure 2(b) and we can see how SLAVE correctly approaches the central zone of the surface, where there are enough examples, but the error is very high at the extremes of the graph.



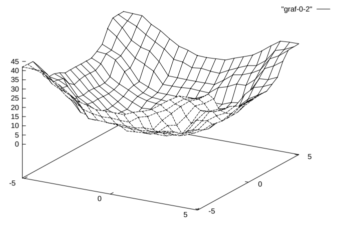
(a) Function $h(x, y) = x^2 + y^2$



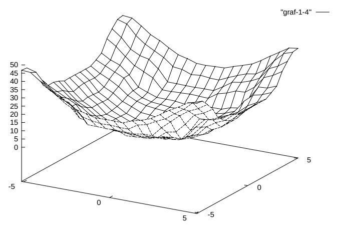
(b) Approximation obtained by SLAVE



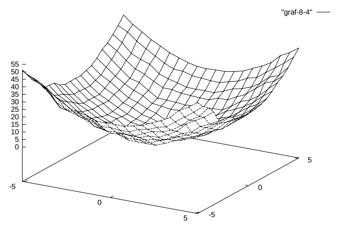
(c) Approximation after step 1



(d) Approximation after step 2



(e) Approximation after step 3



(f) Approximation after refinement

Figure 2: An example of refinement algorithm

The refinement module attempts to correct the approximation obtained by SLAVE using the error measurement. In Figure 2(c), the surface after the specification process is shown. At first glance, this graph presents a worse approximation on the original because a steep slope appears on the surface, but taking the average error, this approximation is better. These steep slopes are due to the elimination of the bad rules which result in some examples not being well covered by the rule set.

From this step, we can see the evolution of the refinement algorithm as a process of smoother approximation on the original function. In Figure 2(d), the control surface is improved using the generalization process where the examples badly covered in the previous step, are covered to a better degree by the generalization of the existent rules.

Figure 2(e) shows the behaviour of the rule set after the step 3. The criterion used for introducing new rules consists in selecting a rule that reduces the maximum individual error. With this criterion, we can see that the approximation of the extremes of the graph are more similar to the original graph.

For this problem, the refinement algorithm is repeated 8 times until the termination condition is satisfied. Figure 2(f) shows the approximation returned by this module. This surface is also similar to the original surface and the improvement on the surface obtained by SLAVE is high.

5 Applying SLAVE to a Control Problem

A traditional control problem used for testing the behaviour of any new controller is the pendulum problem. The pendulum problem consists of, given an example set that describes the system's performance, learning the rule set that permits the pendulum to be controlled. On the assumption of $|\theta| << 1$ (radian) the nonlinear differential equation that leads to the behavior of the pendulum is managed by the equation:

$$m\frac{l^2}{3}\frac{d^2\theta}{dt^2} = \frac{l}{2}(-f + m\ g\ sin\theta - k\frac{d\theta}{dt})$$

where $k\frac{d\theta}{dt}$ is an approximation of the friction strength.

This system can be described using two state variables θ (angle) and ω (angular speed) and the control variable f (force). A pendulum weighing 5 kg and 5 m long has been considered in a real simulation, applying the force to the center of gravity, for a constant time of 10 ms. With these parameters the universe of discourse of the variables are as follows:

$$\theta \in [-0.277, 0.277] \qquad \omega \in [-0.458, 0.458] \qquad f \in [-1593, 1593].$$

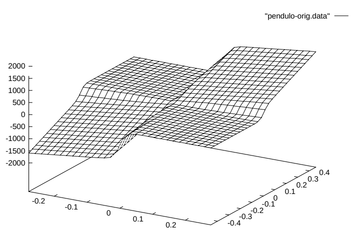
Experimentally, we have obtained two example sets using the previous restriction: the first one contains 213 examples and it is used for learning (training set) and the second one contains 125 examples and it is used for testing the behavior of the learned rule set (test set). These example sets have been obtained from the previous equation on two different initial conditions:

a) θ=-0.277 and ω=0

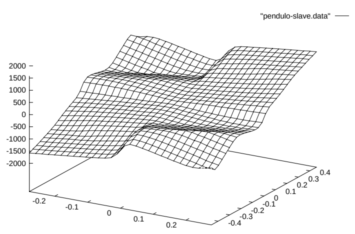
b) θ=0.277 and ω=0.

Figure 3a represents the control surface on the previous conditions.

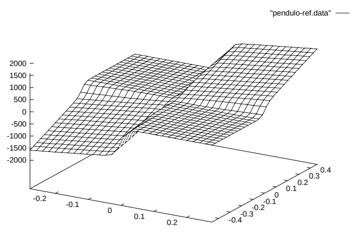
The first step for working with the SLAVE learning system, consists of discretizing the range of variables using fuzzy labels. Figure 4 shows the labels used for each variable.



(a) Control surface
of the pendulum problem



(b) Control surface
obtained by SLAVE



(c) Control surface
obtained by SLAVE+R

Figure 3: Control surfaces

We want to know the behavior of this refinement algorithm under different knowledge bases for this problem. Therefore, we use the following fuzzy rule set:

- **SLAVE**. This is the knowledge base obtained by SLAVE.
- **WM**. This is the knowledge base obtained using the Wang and Mendel algorithm [14].

For the experiment, we have studied the behavior of the previous knowledge bases and their combination with the refinement algorithm.

Table 1: The knowledge base obtained by WM

1	If θ is ZR and ω is ZR Then f is ZR
2	If θ is PS and ω is PS Then f is PS
3	If θ is PM and ω is PM Then f is PM
4	If θ is NS and ω is PS Then f is ZR
5	If θ is NM and ω is PM Then f is ZR
6	If θ is PS and ω is NS Then f is ZR
7	If θ is PS and ω is NM Then f is ZR
8	If θ is NS and ω is NS Then f is NS
9	If θ is NM and ω is NM Then f is NM

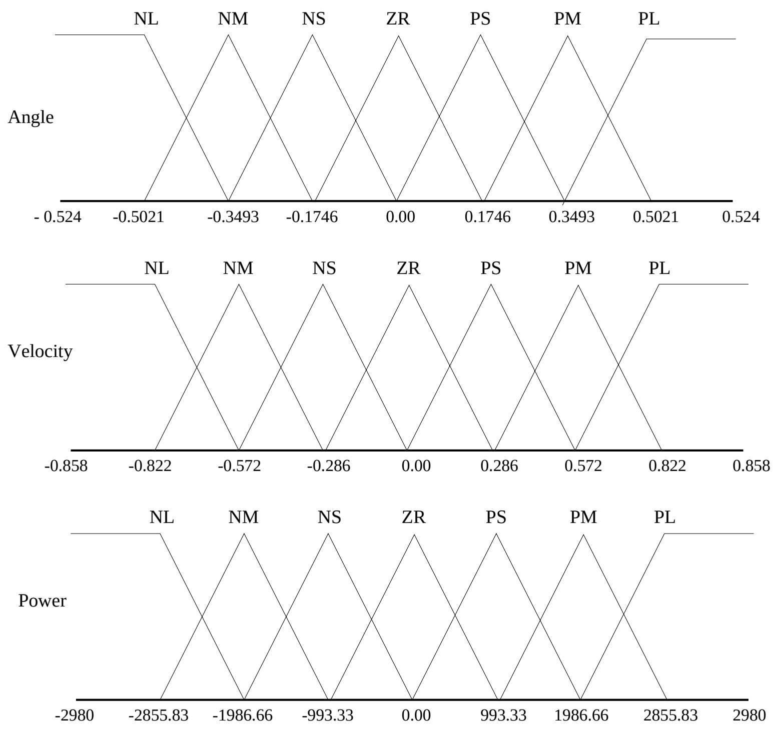


Figure 4: Pendulum problem domains

The set of rules obtained by WM, SLAVE and SLAVE+R are shown in Tables 1, 2 and 3 respectively. It is important to note the simplicity of the set of rules obtained by SLAVE+R.

Table 2: The knowledge base obtained by SLAVE

1	If θ is greater than or equal to ZR and ω is lesser than or equal to ZR Then f is ZR
2	If θ is lesser than or equal to NS and ω is NS Then f is NS
3	If θ is greater than or equal to PS and ω is PS Then f is PS
4	If θ is lesser than or equal to NS and ω is greater than or equal to ZR Then f is ZR
5	If θ is lesser than or equal to NM and ω is lesser than or equal to ZR Then f is NM
6	If θ is greater than or equal to PM and ω is greater than or equal to ZR Then f is PM

Table 3: The knowledge base obtained by SLAVE+R

1	If θ is NS and ω is NS Then f is NS
2	If θ is PS and ω is PS Then f is PS
3	If ω is ZR Then f is ZR
4	If θ is NM or NS and ω is NM Then f is NM
5	If θ is PS or PM and ω is PM Then f is PM

Table 4 shows the number of rules and the error obtained using the test example set with the previous knowledge bases and the refinement algorithm. In all the cases, we have taken the max-min inference system and the average of the center of gravity weighted by the matching value as a defuzzification method and the minimum operator as t-norm.

Figure 3b represents the control surface obtained by SLAVE and Figure 3c the control surface obtained by SLAVE+R.

The Wang-Mendel result has been obtained from the learning algorithm proposed in [14] and using the same fuzzy discretization as we used in SLAVE.

Table 4: Table of results

Rule Database	Error	Number of Rules
WM	7.45	9
SLAVE	492.16	6
WM+R	6.92	6
SLAVE+R	1.21	5

6 Concluding Remarks

In this paper we have proposed an algorithm for the refinement of fuzzy control rules. In this process, the semantic of the rules has not changed. The algorithm begins with an incomplete and/or incorrect fuzzy rule base that represents the initial knowledge (where this set can be empty) and a set of consistent examples, the refinement problem consists of modifying this initial set of rules in order to better adapt it to the example set. The algorithm is based on a process of specification and generalization. The initial set of rules can be obtained from an expert or from a learning algorithm like SLAVE.

In the examples we have shown how the evolution of the refinement algorithm is a process of smoother approximation on the original system. SLAVE+R has been shown to be an improved version of SLAVE also for control problems.

A very interesting combination that will be investigated in future works is the definition of a refinement algorithm capable of modifying the semantic of the rules.

References

[1] Bonissone, P.P., Fuzzy Logic Controllers: An Industrial Reality. In Computational Intelligence: Imitating Life. IEEE Press pp.316-327 (1994)

[2] González A., Pérez R., Verdegay J.L., Learning The Structure of a Fuzzy Rule: a Genetic Approach, Proc. EUFIT'93 vol. 2 814-819 (1993). Also Fuzzy System and Artificial Intelligence, 3 n.1 57-70, (1994).

[3] González, A., Pérez, R., Structural learning of fuzzy rules from noisy examples. Proc. FUZZIEEE/IFES'95, Yokohama, vol.III, pp. 1323-1330 (1995).

[4] González, A., Pérez, R., Completeness and Consistency conditions for learning fuzzy rules, Technical Report #DECSAI-95103 (1995), also to appear in Fuzzy Sets and Systems.

[5] González, A., Pérez, R., A Learning System of Fuzzy Control Rules. In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 202-225, (1996).

[6] González, A., Pérez, R., A refinement algorithm of fuzzy rules for classification problems, Proceedings of the IPMU'96, vol. 2, pp. 533-538 (1996).

[7] González, A. Pérez, R., Aplicación de un Sistema de Refinamiento de Reglas a Problemas de Clasificación. Memoria del V Congreso Iberoamericano de Inteligencia Artificial. Editorial Limusa. pp. 20-29, (1996).

[8] González, A. Herrera, F., Multi-stage Genetic Fuzzy Systems Based on the Iterative Rule Learning Approach, to appear in Mathware and Soft Computing (1997).

[9] Lee, C.C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Parts I and II. IEEE Transactions on Systems, Man and Cybernetics, 20, pp. 404-435 (1990)

[10] H. Ishibuchi, T. Murata, A Genetic-Algorithm-Based Fuzzy Partition Method for Pattern Classification Problems. In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 555-578, 1996.

[11] L. Magdalena, J.R. Velasco, A Learning System of Fuzzy Control Rules Based on Genetic Algorithms. In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 172-201, 1996.

[12] Mamdani, E.H., Assilian, S., An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies, 7, pp. 1-13 (1975)

[13] Sugeno, M., Tanaka, K., Successive identification of a fuzzy model and its applications to prediction of a complex system, Fuzzy Sets and Systems 42, pp. 315-334, (1991).

[14] Wang, L., Mendel, M., Generating fuzzy rules by learning from examples, IEEE Transactions on systems, man and cybernetics, vol.22 n.6, pp.1414-1427 (1992)