DECSAI

Department of Computer Science
and Artificial Intelligence

Genetic Algorithms and Fuzzy Logic in Control Processes

O. Cordón, F. Herrera, E. Herrera-Viedma, M. Lozano

Technical Report #DECSAI-95109
March, 1995

ETS de Ingeniería Informática. Universidad de Granada.
18071 Granada - Spain - Phone +34.58.244019, Fax +34.58.243317

Genetic Algorithms and Fuzzy Logic in Control Processes

Summary

Genetic Algorithms and Fuzzy Logic in Control Processes*

O. Cordón, F. Herrera, E. Herrera-Viedma, M. Lozano

Dept. of Computer Science and Artificial Intelligence
E.T.S. de Ingeniería Informática
University of Granada, 18071 - Granada, Spain

Abstract

In this paper we describe the genetic algorithms and fuzzy logic, focusing them as tools to model control processes and to design intelligent and automatic control systems. We describe the application of genetic algorithms to design fuzzy logic controllers, as well as the learning classifier systems and their development in a fuzzy environment, the fuzzy learning classifier systems.

Keywords: Genetic algorithms, fuzzy logic, control processes, fuzzy logic controllers, classifier systems.

1 Introduction

The assumption that all engineering system modeling can be reduced to an exact set of algebraic and differential equations has been challenged by research that recognizes that measurements, process modeling and control can never be exact for real complex processes.

There is a necessity to reach advanced control technologies able of:

- managing uncertainty and expert knowledge
- accommodating significant changes in the plant and its environment,
- incorporating techniques for learning either uncertain information, or a changing environment, and methods of combining existing knowledge with a learning process.

According to them, a problem is how to represent and compute processes that are imprecisely described or are controlled by humans without recourse to mathematical models, algorithms or a deep understanding of the physical processes involved. Fuzzy logic (FL), which may be viewed as an extension of classical logical systems, provides an effective conceptual framework for dealing with the problem of knowledge representation in an environment of uncertainty and vagueness. Among the most successful application of FL are the Fuzzy Logic Controllers (FLCs). FLCs implement an expert operator's approximate reasoning process in the selection of a control action.

Another problem is how to get ready adaptive techniques, which permit to have intelligent control systems, that is, systems involving learning or adaptation in response to changes in process parameters. FL is a powerful tool for knowledge representation in computational intelligence. On the other hand, adaptive control, learning and self-organization can be considered in a lot of cases as optimization or search processes. Genetic algorithms (GAs) are search algorithms

*This research has been supported by DGICYT PB92-0933

that use operations found in natural genetics to guide the trek through a search space. GAs are theoretically and empirically proven to provide robust search in complex spaces, offering a valid approach to problems requiring efficient and effective search.

FL and GAs are two important tools for modeling and managing intelligent and automatic control systems, which are able of supporting the above features. Each of them have different advantages, on one hand, nonlinearity and explicit knowledge expression of FL and, on other hand, learning capability, global and local search approach of GAs. Recently there are an increasing number of publications about the combination of these two topics. The integration between GA and FL may produce useful results. The application of GAs to design FLCs has been widely developed. In fact, the usefulness of the GAs in this task has been widely shown.

Here, we describe the GA and the FLCs. We introduce the FLCs and their applications. We present the GAs, some GA applications developed to control and engineering processes, and theirs applications to design FLCs. Also we approach the learning classifier systems as the usual GA paradigm in machine learning and their development in a fuzzy environment, the fuzzy learning classifier systems.

In order to do that we organize the paper as follows. Section 2 introduces the FLCs while section 3 presents the GAs. Section 4 presents the design of FLCs using GAs. Section 5 includes the description of the learning classifier systems and the fuzzy learning classifier systems. Finally some conclusions are pointed out.

2 Fuzzy Logic Controllers

2.1 Description of the Fuzzy Logic Controllers

The purpose of any controller is to look periodically the values of the state variables of the controlled system and to obtain the values associated to their control variables by means of the relationships existing between them. If those relationships can be expressed in a mathematical way, it is not too much difficult to design the controller. The problem comes when, as it happens in a lot of real world nonlinear systems with complex dynamics, there is not a mathematical model representing the existing relationships.

In the 40's and 50's, many researches proved that many dynamic systems can be mathematically modeled using differential equations. These previous works represent the foundations of the *Control Theory* which, in addition with the *Transform Theory*, provided an extremely powerful means of analyzing and designing control systems. These theories were being developed until the 70's, when the area was called *Systems Theory* to indicate its definitiveness [Mam93]. Its principles have been used to control a very big amount of systems taking mathematics as the main tool to do it during many years. Unfortunately, in too many instances this approach could not be sustained because many systems have unknown parameters or highly complex and nonlinear characteristics that make them not to be amenable to the full force of mathematical analysis as dictated by the Control Theory.

Over the last few years the application of *Artificial Intelligence* techniques has become a research topic in the domain of processes control, taking the purpose of avoid the commented drawbacks and allow to obtain efficient controllers which utilizes the human experience in a more related form than the conventional mathematical approach. In the cases in which a mathematical representation of the controlled systems cannot be obtained, the process operator should be able to express the relationships existing in them, that is, the process behavior.

Fuzzy Logic Control is the main topic of this new field known as *Expert Control*. *FLCs* initiated by Mamdani and Assilian in the work [Mam75], are now considered as one of the most important applications of the *Fuzzy Set Theory* suggested by Zadeh in 1965 [Zad65] presenting

the notion of *fuzzy set*, generalization of the ordinary set characterized by a membership function μ taking values in the interval [0,1] representing degrees of belonging to the set (not absolute belonging as in classical sets), playing a central role. FLCs are knowledge based controllers that are usually derived from a knowledge acquisition process or are automatically synthesized from a self-organizing control architecture [Bon94].

While conventional linear controllers can be viewed as a hyperplane in a N+1-dimensional space, mapping an Nth dimensional state vector to a control action, FLCs, on the other hand, typically define a non-linear mapping from the system's state space to the control space. Thus, it is possible to visualize the results of a FLC as a nonlinear control surface reflecting the process operator's prior knowledge.

A FLC is composed by a *Knowledge Base*, that comprises the information given by the process operator in form of linguistic control rules, a *Fuzzification Interface*, who has the effect of transforming crisp data into fuzzy sets, an *Inference System*, that uses them joined with the Knowledge Base to make inference by means of a reasoning method, and a *Defuzzification Interface*, that translates the fuzzy control action so obtained to a real control action using a defuzzification method. The generic structure of a FLC is shown in figure 1 [Lee90].

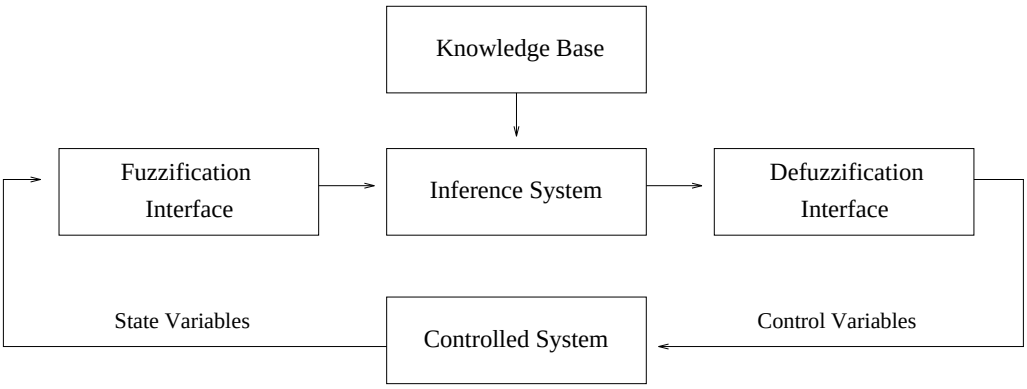

Figure 1: Generic structure of a fuzzy logic controller

The **Knowledge Base** encodes the expert knowledge by means of a set of fuzzy control rules. A fuzzy control rule is a conditional statement with the form *IF (a set of conditions are satisfied) THEN (a set of consequences can be inferred)* in which the antecedent is a condition in its application domain, the consequent is a control action to be applied in the controlled system (notion of control rule) and both antecedent and consequent are associated with fuzzy concepts, that is, linguistic terms (notion of fuzzy rule).

Thus the Knowledge Base is composed of two components, a *Data Base*, containing the definitions of the fuzzy control rules linguistic labels, that is, the membership functions of the fuzzy sets specifying the meaning of the linguistic terms, and a *Rule Base*, constituted by the collection of fuzzy control rules representing the expert knowledge. There are different kinds of rules proposed in the specialized literature regarding to the expression of the consequent. Mamdani employs rules in which the consequent is another fuzzy variable [Mam75] while Sugeno uses rules whose conclusion is a polynomial function of the inputs [Sug85]. Another kind of rules present too the consequent being a function of the input parameters. The following three rules show respectively the generic expressions of the three types commented:

$If\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ then\ Y\ is\ B$
$If\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ then\ Y = p_0 + p_1X_1 + ... + p_nX_n$
$If\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ then\ Y = f(X_1, ..., X_n)$

being the X_i and Y linguistic variables and the A_i and B fuzzy sets specifying the meaning of them.

Without lack of generality, in the following we consider a Rule Base constituted by m Mamdani type fuzzy control rules R_i, i = 1, ..., m, with the form:

If X_{11} *is* A_{11} *and ... and* X_{1n} *is* A_{1n} *then Y is* B_1
also
...
also
If X_{m1} *is* A_{m1} *and ... and* X_{mn} *is* A_{mn} *then Y is* B_m

As we have commented, the Knowledge Base encodes the expert known knowledge of the controlled system. So it is the only component depending on the concrete application and it makes the accuracy of the designed FLC depends directly on its composition. There are four modes of derivation of fuzzy control rules that are not mutually exclusive [Ber92, Lee90]. These modes are the following:

1. *Expert Experience and Control Engineering Knowledge.*
2. *Modeling of the Operator's Control Actions.*
3. *Based on the Fuzzy Model of a Process.*
4. *Based on Learning and Self-Organization.*

The first method is the most widely used. This method is effective when expert human operators can express that they use it to control the system in terms of control rules. The rules more usually obtained by means of this process are Mamdani type. Since they present an adequate form to represent the expert knowledge. The second method directly models the control actions of the process operator. Instead of interviewing the operator, the types of control action taken by it are modeled. The third approach is based on the developing of a model of the plant and construct a FLC to control the fuzzy model generating the fuzzy control rules of the Knowledge Base by means of the fuzzy model of the system. It makes this approach similar to that traditionally used in Control Theory. Hence, structure and parameter identification are needed. Finally, the fourth method is focused on learning. In this case, the ability to create fuzzy control rules and to modify them based on experience in order to improve the controllers performance is considered.

The **Fuzzification Interface** defines a mapping from an observed input space to fuzzy sets in certain input universes of discourse, obtaining the membership function associated to each one of the system inputs. Symbolically,

$$F = fuzzifier(x_0)$$

where x_0 is a crisp input value from a controlled system, F is a fuzzy set and $fuzzifier$ represents a fuzzification operator.

There are two main types of fuzzification:

a) *Point fuzzification*:

$$F(x) = \begin{cases} 1 \; if \; x = x_0 \\ 0 \; otherwise \end{cases}$$

b) *Approximate function:*

$F(x) = 0$ if and only if $|x - x_0| < \delta$

The **Inference System** is based on the application of the Generalized Modus Ponens (GMP), extension of the classical logic Modus Ponens, proposed by Zadeh in the way:

$$\frac{\begin{array}{l} If\ X\ is\ A\ then\ Y\ is\ B \\ X\ is\ A' \end{array}}{Y\ is\ B'}$$

The fuzzy conditional statement *If X is A then Y is B* (being X, Y linguistic variables and A, B fuzzy sets) represents a fuzzy relation between A and B defined in $\mathbf{X}\times\mathbf{Y}$, being $\mathbf{X}$ and $\mathbf{Y}$ the universes of the variables X and Y respectively. The fuzzy relation is expressed by a fuzzy set R whose membership function $\mu_R(x,y)$ is given by:

$$\forall x \in \mathbf{X}, y \in \mathbf{Y} : \mu_R(x,y) = I(\mu_A(x), \mu_B(y))$$

being $\mu_A(x)$ and $\mu_B(y)$ the membership functions of the fuzzy sets A and B respectively and I a fuzzy implication operator. The consequent B' obtained from the GMP is deduced by projection on $\mathbf{Y}$ by means of the Compositional Rule of Inference (CRI) given by the following expression in what T' is a connective:

$$\mu_{B'}(y) = Sup_{x\in X}\left\{T'(\mu_{A'}(x), I(\mu_A(x), \mu_B(y)))\right\}$$

Since the input x corresponding to the state variables of the controlled system is crisp, $x = x_0$, the application of the first type of fuzzification provokes the fuzzy set A' to be a singleton, that is, $\mu_{A'}(x) = 1\ if\ \ x = x_0$ and $\mu_{A'}(x) = 0\ if\ \ x \neq x_0$. Thus the CRI is reduced to the following expression:

$$\mu_{B'}(y) = I(\mu_A(x_0), \mu_B(y))$$

Finally, when the rules of the Knowledge Base have more than one variable in the antecedent (that is, they present the generic form $If\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ then\ \ Y\ is\ B$), $x_0 = (x_1, ..., x_n)$ and

$$\mu_A(x_0) = T(\mu_{A_1}(x_1), \mu_{A_2}(x_2), ..., \mu_{A_n}(x_n))$$

being T a conjunctive operator.

Since from each rule R_i is obtained a fuzzy set B_i' from the inference process, the **Defuzzification Interface** uses an aggregation operator G, representing the connective also of the Knowledge Base control rules, which composes them and applies a defuzzification method D to translate the fuzzy sets obtained in this way into values corresponding to the control variables of the system. So, calling S to the FLC, x_0 to the inputs value and y_0 to the crisp value obtained from the defuzzification, we have:

$$\mu_{B'}(y) = G\left\{\mu_{B_1'}(y), \mu_{B_2'}(y), ..., \mu_{B_n'}(xy)\right\}$$
$$y_0 = S(x_0) = D(\mu_{B'}(y))$$

At present, the commonly used strategies may be described as the *Max Criterion*, the *Mean of Maximum (MOM)* and the *Center of Area (COA)* [Lee90]:

- The Max Criterion takes the point at which the fuzzy set representing the fuzzy control action, B', reaches its maximum value.
- The MOM strategy generates a control action which represents the mean value of all local control actions whose membership functions reach the maximum.

- The widely used COA strategy generates the center of gravity of the fuzzy set B'.

In [Kis85] several factors were presented that have a significant influence in the FLC such as:

1. The form of the mathematical definition of the fuzzy implication in the fuzzy control rules (If ... and ... then), that is, the selection of the fuzzy implication operator I representing the fuzzy relation R.
2. The form of the mathematical definition of the sentence connective and, that is, the selection of the conjunctive operator T.
3. The form of the mathematical definition of the sentence connective also, that is, the selection of the aggregation operator G.
4. The form of the mathematical definition of composition of fuzzy relations existing in the CRI.
5. The way of defining the defuzzification operator D.

The influence of several of these factors is analyzed in [Kis85, Car93, Car95], taking as base several control applications.

2.2 Applications

During the past several years, many applications of FLCs have been developed successfully. FLCs have been proved to be superior in performance to conventional systems in many applications. It should be noted that the first industrial application was the cement kiln controller developed by the Danish cement plant manufacturer F. L. Smith in 1979 [Umb80]. Some of other more recent applications are water treatment, combustion control system for a refuse incineration plant, japanese sake fermentation control, elevator control, highway tunnel ventilation control system, automatic train operation system, container crane operation system, fully automatic washing machine, vacuum cleaner, video equipment, recuperative turboshaft engine control, locomotive wheel slip control, steam turbine cycling, power electronics control, heat exchange, warm water process control, activated sludge wastewater treatment, traffic junction, aircraft flight control, turning process, robot control, model-car parking and turning, automobile speed control, nuclear reactor control, fuzzy memory devices, fuzzy computer, welding, water purification process control, control of a liquid level rig, automobile transmission control, gasoline refinery catalytic reformer control, two-dimensional ping-pong game playing, and control of biological processes [Hir93, Bon94, Lee90, Ber92].

More complete information about FLCs can be found in [Lee90, Hel93, Ber92, Bon94].

3 Genetic Algorithms

3.1 Description of the Genetic Algorithms

GAs are general-purpose search algorithms that use principles inspired by natural population genetics to evolve solutions to problems [Hol75]. The basic idea is to maintain a population of knowledge structures that evolves over time through a process of competition and controlled variation. Each structure in the population represents a candidate solution to the concrete problem and has an associated *fitness* to determine in the process of competition which structures are used to form new ones. The new ones are created using genetic operators such as crossover and

mutation. GAs have had a great measures of success in search and optimization problems. The reason of great part of its success is their ability to exploit accumulating information about an initially unknown search space in order to bias subsequent search into useful subspaces, i.e., *their robustness*. This is their key feature, overcoat in large, complex and poorly understood search spaces, where the clasical search tools (enumerative, heuristic,..) are inappropriate, offering a valid approach to problems requiring efficient and effective search.

A GA starts with a population of randomly generated solutions, chromosomes and advances toward better solutions by applying genetic operators, modeled on the genetic processes occurring in nature. In these algorithms we maintain a population of solutions for a given problem; this population undergoes evolution in a form of natural selection. In each generation, relatively good solutions reproduce to give offsprings that replace the relatively bad solutions which die. An evaluation or fitness function plays the role of the environment to distinguish between good and bad solutions. The process of going from the current population to the next population constitutes one generation in the execution of a genetic algorithm.

Although there are many possible variants of the basic GA, the fundamental underlying mechanism operates on a population of chromosomes or individuals (representing possible solutions to the problem) and consists of three operations:

(1) evaluation of individual fitness,

(2) formation of a gene pool (intermediate population) and

(3) recombination and mutation.

The figure 2 shows the structure of a simple GA.

```
Procedure Genetic Algorithm
begin (1)
    t = 0;
    initialize P(t);
    evaluate P(t);
    While (Not termination-condition) do
    begin (2)
        t = t + 1;
        select P(t) from P(t − 1);
        recombine P(t);
        evaluate P(t);
    end (2)
end (1)
```

Figure 2: Structure of a GA

A fitness function must be devised for each problem to be solved. Given a particular chromosome, a solution, the fitness function returns a single numerical fitness which is supposed to be proportional to the utility or adaptation of the individual which that chromosome represents.

There are a number of ways to do selection. We might view the population as mapping onto a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, individuals are chosen using "stochastic sampling with replacement" to fill the intermediate population. The selection procedure proposed by Baker, [Bak87], and called *stochastic universal sampling* is one of the most efficient. The number of offspring of any structure is bound by the floor and ceiling of the expected number of offspring [Bak87].

After selection has been carried out, the construction of the intermediate population is complete and recombination and mutation can occur.

The crossover operator combines the features of two parent structures to form two similar offsprings. It is applied at a random position with a probability of performance, the crossover probability, P_c. The mutation operator arbitrarily alters one or more components of a selected structure so as to increase the structural variability of the population. Each position of each solution vector in the population undergoes a random change according to a probability defined by a mutation rate, the mutation probability, P_m.

The next figure illustrates the basic operations: reproduction, crossover and mutation.

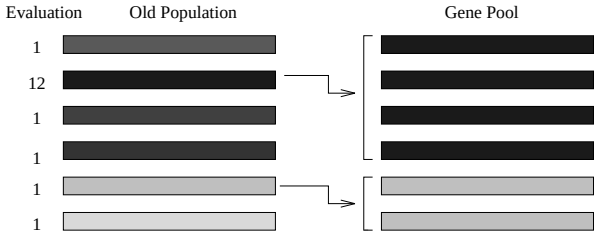

Figure 3: Evaluation and contribution to the gene pool

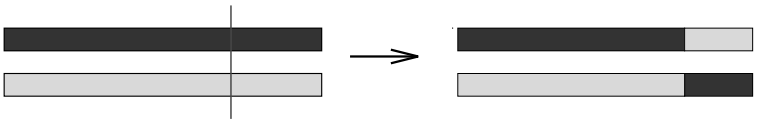

Figure 4: Recombination. One-point crossover

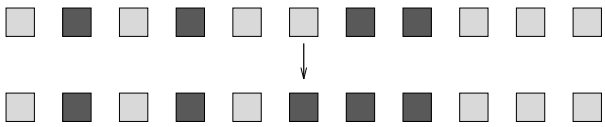

Figure 5: Mutation

It is generally accepted that a GA to solve a problem must take into account the five following components:

1. *A genetic representation of solutions to the problem,*
2. *a way to create an initial population of solutions,*
3. *an evaluation function which gives the fitness of each individual,*
4. *genetic operators that alter the genetic composition of children during reproduction, and*
5. *values for the parameters that the GA uses (population size, probabilities of applying genetic operators, etc.).*

The basic principles of the GAs were first laid down rigorously by Holland [Hol75] and are well described in many texts as [Dav91b, Gol89, Mic92].

Numerous GAs applications have been presented during the last years. Some of them can be summarize as numerical function optimization, combinatorial optimization, image processing, fuzzy logic, engineering processes, biology, artificial life, machine learning, etc. [Bel91, Män92, For93, ICEC94, Dav94].

There are many applications of GAs to learning systems [Dej88, Gre93, Gre94], a usual paradigm being that of a learning classifier system [Hol78]. The GA tries to evolve a set of *if ...*

then rules to deal with some particular situation. Learning classifier systems will be presented in section 5.

3.2 Genetic Algorithms in Control Processes

Focusing into the application of the GAs in engineering system modeling, we find a great quantity of applications. In the following we summarize some of them in order to show the potential of GAs in engineering system modeling.

- *Optimizing robot trajectories*

  In [Dav91a] is introduced a GA which successfully handles the trajectory generation of the redundant robot model described.

  The proposed GA incorporates some few new mechanisms which are necessary to make a GA amenable to the natural trajectory representation used.

  The majority of changes needed were concentrated in adopting the reproduction operator to suit the varying in length and an order dependent representation of trajectories. The results presented demonstrate the power and robustness of the trajectory-GA. It optimizes trajectories efficiently and, more importantly, reliably.

- *Parametric Design of Aircraft*

  In [Bra91] the optimizing aircraft designs when the task is posed as that of optimizing a list of parameters were discussed.

- *Air-Injected Hydrocyclone Optimization*

  In [Kar91c] the design of an air-injected hydrocyclone as a list of parameters was represented. In this approach the authors use a new operator called "simplex reproduction", showing that a GA using this operators is quite effective as a search technique for finding design parameter combinations.

- *Multiple Fault Diagnosis*

  In [Lie91] the use of a genetic algorithm for finding the most plausible combination of causes for alarms in a microwave communication system was discussed.

- *Schedule Optimization*

  In [Sys91] the application of a GA to the problem of scheduling activities in a laboratory in which each activity may affect the others in a variety of ways was described .

- *Control-Cost-Driven Evolution*

  Genetic programming (GP) is essentially a variant of GA with a different problem representation. Koza [Koz92] started from the observation that problem representation is a key issue in GAs because it is actually the coded representation of the underlying problem that a GA can manipulate.

  > "For many problems in machine learning and artificial intelligence, the most natural known representation for a solution is a hierarchical computer program of indeterminate size and shape, as opposed to character strings whose size has been determined in advance". [Koz92, p. 210].

  GP provides a way to find an approximately correct function for problems of control and optimal control for which an exact mathematical solution cannot be obtained. In the chapter 11 of [Koz92] it is demonstrated the use of GP on the well known optimal control

problem of balancing a broom, the control problem of backing up a tractor-trailer truck and an optimization problem.

Other papers about that can be found in the proceeding of the last conferences [Bel91, Män92, For93, ICE94, Dav94]. By means of this summary we have tried to show the faculty of GAs for multiple applications in ingeniering and control processes.

4 Design of Fuzzy Logic Controllers using Genetic Algorithms

When we try to design a FLC, two problems arise: first, how to establish the structure of the controller and, second, how to set numerical values of the controller parameters. The GAs have been successfully applied in these problems, learning controller structure as well as tuning controller parameters. In fact, the GAs search the fuzzy control rules (FCR) base that verify the optimality conditions specified by their fitness function according to the required features.

In the following we present some proposals according the above settled aspects.

4.1 Tuning controller parameters

A FLC contains a number of sets of parameters that can be altered to modify the controller performance. They are [Hel93]:

- the scaling factors for each variable,
- the fuzzy sets representing the meaning of linguistic values,
- the if-then rules.

Each of these sets of parameters has been used as the controller parameters to be adapted in different adaptive FLCs.

GAs have been used to modify the fuzzy set definitions, to alter the shapes of the fuzzy sets defining the meaning of the linguistic terms, to determine the membership functions that produce maximum control performance according to the inference system (fuzzy implication and conjunctive operator) and the defuzzification strategy used. That is, to tune the FCR set, in order to make the FLC behaves as closely as possible to the operator or expert behavior. This method relies on having a set of training data against which the controller is tuned.

The tuning method using GAs fits the membership functions of the fuzzy rules dealing with the parameters of the membership functions, minimizing a square error function defined by means of an input-output data set for evaluation.

Recent works have been centred on the the use of GAs altering the set definitions so that the FLC matches a suitable set of reference data as closely as possible [Kar91a, Kor93, Hes93, Var93, Her95a].

A chromosome represents one possible solution to the problem, that is, one possible FCR base . The fitness function itself depends on the task of the FLC, usually the square error can be considered, then the chromosome is tested by evaluating the training data set.

4.2 Learning controller structure

For learning the controller structure different hypotheses can be considered, either to work with a determined variables domain or to manage rules with a free structure. According to these two possible models, different GA learning processes have been proposed.

Determined variables domain

We assume that each universe, U, contains a number of referential sets having their linguistic meaning which form a finite set of fuzzy sets on U. Membership functions are defined to represent the developer's conception of the linguistic variables. For instance if X is a variable on U for temperature, then one may define A_1 as "*low temperature*", $A_i(1 < i < r)$ as "*medium temperature*" and A_r as "*high temperature*". These referential fuzzy sets are characterized by their membership functions $A_i(u) : U \rightarrow [0, 1], i = 1, ..., r$. To ensure the performance of the fuzzy model and provide an uniform basis for further study it is required that all referential sets should be normal convex and satisfy the following completeness condition:

$$\forall u \in U \ \exists j, \ 0 \leq j \leq r, \text{ such that } A_j(u) \geq \delta$$

where δ is a fixed threshold, the *completeness degree* of the universes.

Free variables structure

Rules with a free structure, without an initial fuzzy set referential, can be also considered.

The rules have the form

$$R_i : \text{IF } x_1 \text{ is } A_{i1} \text{ and ... and } x_n \text{ is } A_{in} \text{ THEN } y_1 \text{ is } B_{i1} \text{ and ... and } y_m \text{ is } B_{im}$$

where $x_1, ..., x_n$ and $y_1, ..., y_m$ are the process state variables and the control variables respectively, and $A_{i1}, ..., A_{in}$, $B_{i1}, ..., B_{im}$ are fuzzy sets in the universes of discourse $U_1, ..., U_n$, $V_1, ..., V_m$.

These fuzzy sets are characterized by their membership functions

$$A_{ij}(B_{ih}) : U_j(V_h) \rightarrow [0, 1], \ j = 1, ..., n, \ h = 1, .., m$$

We can consider every fuzzy set associated to a normalized trapezoidal membership function. A computational way to characterize it is to use a parametric representation achieved by means of the 4-tuple $(a^1_{ij}, a^2_{ij}, a^3_{ij}, a^4_{ij})$, $(b^1_{ih}, b^2_{ih}, b^3_{ih}, b^4_{ih})$, $j = 1, ..., n, h = 1, .., m$.

Next we describe two of the GA learning processes proposed in the literature for each one of the variables structure.

4.2.1 GA learning processes with determined variables domain

The method proposed by Karr [Kar91b]

The rule set is formed as follows. Membership functions were defined to represent the developer's conception of the linguistic variables (fuzzy sets) and these variables made the formation of the rule set a straightforward task. The selection of the decision variables and the fuzzy sets describing required a number of rules n. From all combination of antecedent labels, one action must be found via GAs. Considering seven fuzzy sets describing the control variables, the entire set of possible actions for one rule was represented as a three-bit string (000 represented action 1, 001 represented action 2, and so on). Because n rules were possible, a string of length $3n$ represents every possible rule set for the FLC.

Once an acceptable rule set was learned with a GA, the selection of high-performance membership functions with the rule set is carried out using the above described tuning process.

The method proposed by Thrift [Thr91]

The method proposed by Thrift is similar to the above proposed by Karr, except that Thrift introduced a new possible value for the consequent of rules, the label "_". The "_" symbol indicates that there is no fuzzy set entry at a position that it appears. A chromosome is formed from the decision table by going rowwise and producing a string of numbers from the code set.

In this way, during the learning process it is determined the number of rules necessary in the control process because the rules with the consequent label "_" can be eliminated.

The codification of the solutions is different of the above proposal. Each rule has assigned a gene taking integer numbers. There exist as many genes as possible combinations of the state variable labels. The range of the genes from 0 to m includes a code for the label "_" as possible value of a gene. There are particular features of the GA based on the coding strategy described above. A mutation operator changes a fuzzy code either up or down a level, or to the blank code (if it is already blank, then it chooses a non-blank code at random). The crossover operator is the standard two-point crossover.

4.2.2 GA learning processes with free rules structure

The Method proposed by Cooper and Vidal [Coo93]

In contrast to prior genetic fuzzy systems which require every input-output combination to be enumerated, they propose a novel encoding scheme which maintains only those rules necessary to control the target system.

They defined a special GA where mutations include inversion of the copied bit and the addition or deletion of an entire rule. These latter two mutations permit the size of a system's FCR base to evolve. The cycle of evaluation and reproduction continues for a predetermined number of generations or until an acceptable performance level is achieved.

The membership function for each variable is a triangle characterized by the location of its center and the half-length of its base. A single rule, therefore, consists of the concatenation of the one-byte unsigned characters (assuming values from 0 to 255) specifying the centers and half-lengths of the membership functions. The rule descriptions for a single fuzzy system are then concatenated into a single bit string where the number of rules is not restricted.

To be meaningful, the genetic paradigm requires that the rules in the two strings be aligned so that similar rules are combined with each other. Simply by combining the strings in the order they appear it does not preserve much information about either system and produces nearly random results, rather as a child system that performs in a manner similar to its parents. Therefore, before reproduction, both strings must be aligned so that the centers of the input variables match as closely as possible. The most closely matching rules are combined first, followed by the next most closely matching rules from those that remain and so on. Any rules forming a longer string that is not matched are added at the end.

The method proposed by Herrera et al. [Her95b]

The proposed learning fuzzy control rules process is based on the use of GAs under the following hypotheses:

- There is some linguistic information from the experience of the human controller but linguistic rules alone are usually not enough for designing a successfully control system or could not be available.
- There is some numerical information from sampled input-output (state-control) pairs that are recorded experimentally.

- The combination of these two kinds of information may be sufficient for a successful design of a FCR base.
- We include the possibility of not having any linguistic information and having a complete numerical information.

According to the aforementioned hypothesis a learning process is designed according to the following goals:

- to develop a generating FCR process from numerical data pairs; and
- to develop a general approach combining both kinds of information, linguistic information and fuzzy control rules obtained by the generating process, into a common framework using both simultaneously and cooperatively to solve the control design problem.

In order to reach these goals, it is proposed a methodology based on the design of the three following components:

a) a generating fuzzy rules process of desirable fuzzy rules able to include the complete knowledge of the set of examples,

b) a combining information and simplifying rules process, which finds the final set of fuzzy rules able to approximate the input-output behaviour of a real system,

c) a tuning process of the final set of rules,

all of them developed by means of GAs.

As it is possible to have some linguistic $IF - THEN$ rules given by an expert, it is used a linguistic fuzzy rules structure to represent them. On other hand, there are sampled input-output pairs and to generate the fuzzy rules covering these examples is used a free fuzzy rules structure. Then both kind of rules are combined, applying a simplified method based on a GA, and finally a tuning method is applied over the simplified set of rules.

The generating fuzzy rules process consists of a *generating method* of desirable fuzzy rules from examples using GAs together with a *covering method* of the set of examples.

- The generating method of fuzzy rules is developed by means of a real coded GA (RCGA) where a chromosome represents a fuzzy rule and it is evaluated by means of a frequency method. The RCGA finds the best rule in every running over the set of examples according to the following features which will be included in the fitness function of the GA.
- The covering method is developed as an iterative process. It permits to obtain a set of fuzzy rules covering the set of examples. In each iteration, it runs the generating method, it chooses the best chromosome (rule), assigns to every example the relative covering value and removes the examples with a covering value greater than ϵ.

Because we can obtain two similar rules in the generating process or one rule similar to another given by an expert, it is necessary to combine and simplify the complete set of rules for obtaining the final set of rules. Finally, the tuning method presented in [Her95a] is applied over the simplified set of rules.

Other methods have been proposed under different hypothesis [Bon93, Lee93a, Lee93b, Chw94, Lee94, Hof94, Sat94].

5 Learning Classifier Systems

5.1 Introduction

The man's life is surrounded by many monotonous, difficult and dangerous tasks. Many of them have been eliminated through the developments in autonomous machinery but certain tasks have remained resistant to automation. In particular, tasks that require robust adaptability have proved difficult to automate. These tasks include trouble-shooting, maintenance, navigation through poorly-defined, time-varying environments, and the control of systems that contain many initially unknown and persistently uncertain elements. Biological organisms perform effectively in these sorts of environments through processes that can be broadly described as *learning*. Due to this reason, there have been many automatic system approaches based in natural systems, e.g.: neuronal networks, genetic algoritms, evolutive computing, etc.

The aforementioned problems can be classified as *learning control problems* or *machine learning problems* in which the environment varies too rapidly or is too poorly defined for conventional or adaptive control. In such problems, a *control system (controller)* must map regions of the environmental state space onto widely different control actions. This mapping must be formed adaptively in response to performance-related environmental feedback and must be maintained in memory. More concretely, these problems are called *reinforcement learning control problems*, that is, general models of limited information learning environments where an automated system must adquire knowledge about appropiate actions through ongoing experience and whose learning process is guided by feedback about the quality of developed actions. The researches on such problems are a fundamental concern for the future of high-autonomy machines. A general reinforcement learning control system model is presented in the figure 6.

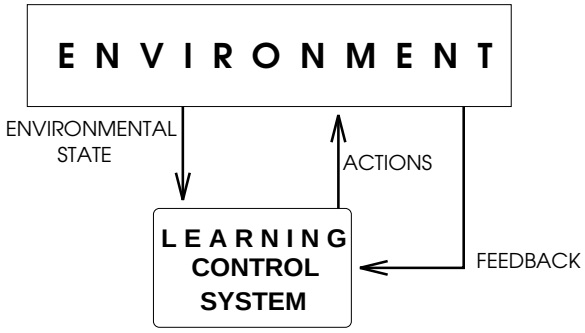

Figure 6: A general reinforcement learning model.

If we visualize a learning control system as consisting of two components, (i) a *task subsystem or performance component*, whose behavior is modified over time via learning, and (ii) a *learning subsystem* responsible for observing the task subsystem over time and effecting the desired behavioral changes. The learning control problem may be restated in terms of searching the space of legal structural changes for instances that achieve the desired behavioral changes. When search space of structural changes of learning control problem is poorly-defined and time-varying, the use of GAs is very effective and appropriate.

GAs are an alternative technique for the designer of learning control systems in which the control actions required for desired performance vary rapidly and in a way that cannot be prespecified.

According to [Dej88] there are a great variety of GA approaches to learning control problem of increasing complexity. A simpler approach is to restrict structural changes to *parameters modification* that control the behavior of performance components and to use GAs to develop a strategy to quickly locate useful combinations of parameters values. A second approach involves using GAs to *change complex data structures* (such as agendas) that control the behavior of the

task subsystem. And a third approach involves using GAs *to change the task program itself.* In this context, the task program is considered a *production system* that consists of an unordered set of rules. It represents the individual population, which is explored and exploited by GAs-based learning subsystem. Historically there are two ways to use the space of production systems to represent the population:

- *The Pitt Approach,* where each member of population represents a set of production rules and, therefore, a population is a set of rule sets [Smi80].
- *The Michigan Approach,* where each member of population represents an individual production rule and, therefore, a population is a set of rules [Hol78].

The Pitt approach seems to be more useful for off-line environments in which more radical behavioral changes are acceptable, whereas the Michigan approach seems to be more useful in on-line and real-time environments in which radical changes in behavior cannot be tolerated.

Our study concerns third approach to the GA-based learning control problem and, more concretely, a class of general purpose reinforcement learning control systems, known as *learning classifier systems* (LCSs), inspired by the Michigan approach and whose foundations were laid by Holland in [Hol75] and later developed in [Hol76, Hol78].

5.2 Description of Learning Classifier Systems

LCSs are defined as a massively parallel, message-passing, rule-based systems that are capable of environmental interaction and reinforcement learning through credit assignment and rule discovery [Boo89]. They typically operate in environments, such as robot world, economic systems, mammalian vision systems, games as chess, and others, which exhibit some of following characteristics: perpetually novel events accompanied by large amounts of noisy or irrelevant data; continual, often real-time, requirements for action; implicitly or inexactly defined goals; and, sparse payoff or reinforcement obtainable only through long action sequences.

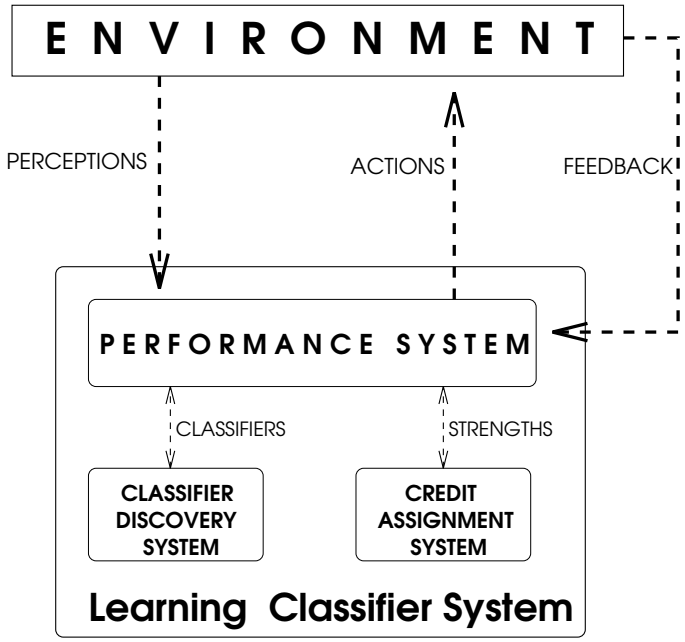

Figure 7: A general learning classifier system.

To work in such environments, LCSs are designed to absorb new information and devise sets of competing hypotheses (expresed as rules) without disturbing capabilities already acquired.

There is a considerable variety in the structural and functional details of the LCSs presented in the literature [Hol78, Gre88, Gol89, Smi91, Par93, Wil94]. The prototypical organization of a LCS is composed of the following three main parts, as it is illustrated in figure 7:

- *The Performance System.*
- *The Credit Assignment (CA) System.*
- *The Classifier Discovery System.*

5.2.1 The Performance System

This is the part of the overall system that interacts directly with the environment. Its activities are environmental interaction and messages processing. It is composed of the six basic elements:

1. *An input interface*, which consists of at least one detector that translates the current state of the environment into standard messages (external messages). Generally, all messages are required to be of a fixed length over a specified alphabet, typically k-bit binary strings.

2. *An output interface*, which consists of at least one effector that translates some messages (action messages) into actions that modify the state of the environment.

3. *A set of rules*, called *classifier list*, represented as strings of symbols over a three-valued alphabet ($A = \{0, 1, \#\}$) with a *condition/action* format. The condition part specifies the messages that satisfy (activate) the classifier and the action part specifies the messages (internal messages) to be sent when the classifier is satisfied. A limited number of classifiers fire in parallel in each cycle. Details on classifiers coding are given in [Gol89, Boo90].

4. *A pattern-matching system,* which identifies which classifiers are matched or satisfaid (matched classifiers) in each cycle of the LCS.

5. *A message list*, which contains all current messages, i.e., those generated by the detectors and those sent by fired classifiers.

6. *A conflict-resolution (CR) system,* which has two functions:

   (a) Determining which matched classifiers fire when the size of the message list is smaller than the number of matched classifiers.

   (b) Deciding which actions to choose in case of inconsistency in the actions proposed to effectors, e.g. "turn left" and "turn right".

   The CR system acts according to some usefulness measures associated to each competing classifiers, i.e. *the relevance to the current situation or specificity* and *the strength or past usefulness.* The CR system is based on an economic analogy and consists of a *bid competition* between classifiers. In this system, matched classifiers bid a certain proportion of their usefulness measures and classifier conflicts are resolved based on a probability distribution over these bids. Higher bidding classifiers are favored to fire and post their messages. If a classifier fires, it must pay out its bid. Thus, each classifier that fires risks a certain percentage of its strength with the possibility of receiving reward as compensation. No global information is maintained to suggest which classifiers compete with one another. Each classifier maintains only its own statistics, which are updated only when the classifier is active.

   Generally, the CR and AC systems make up the *complete inference engine* of LCS and i.e. their activities are interrelated as we shall see later.

A descriptive scheme of the performance system is presented in the figure 8.

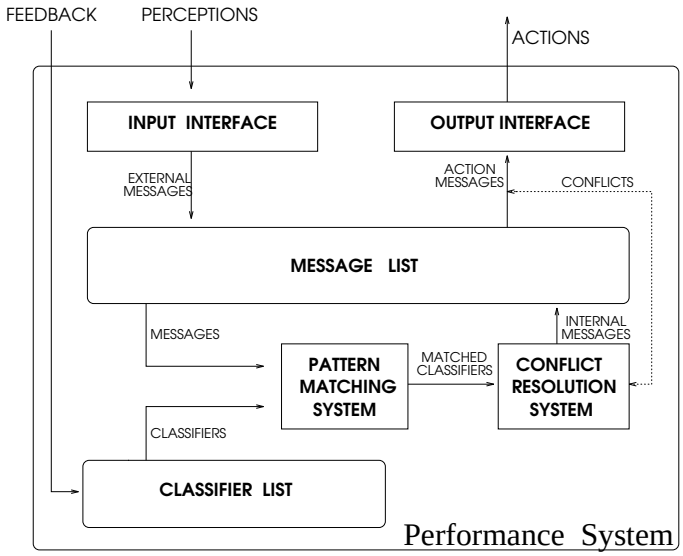

Figure 8: A general performance system.

5.2.2 The Credit Assignment System

The learning of a LCS is divided in two learning processes:

(i) the learning process developed in the credit assignment system,

(ii) the learning process developed in the classifiers discovery system.

In the first one, the set of classifiers is given and its use is learned by means of environmental feedback. In the second one, new and possibly useful classifiers can be created using past experience.

Therefore, the main task of CA system includes the activity of learnig by the modification and adjustment of conflict-resolution parameters of classifier set, i.e., their strengths. There are many different CA system scheme types proposed in the literature. The most important ones are two:

1. The traditional scheme of *the Bucket Brigade Algorithm* (BBA) [Boo82, Hol85, Boo89, Gol89], which is a local learning scheme that requires a few memory and computational requeriments. Usually, the BBA is linked to the CR system and set up the mechanism of a competitive economy known as CA/CR system. In this classical LCS CA/CR system, each satisfied classifier C_j makes a bid Bid_j,

$$Bid_j = b \cdot Str_j \cdot Spec_j$$

where b is a small constant less than 1, called *risk factor*, Str_j is C_j's strength (initialized with the same value for all classifiers), and $Spec_j$ is C_j's specificity; and i.e., a quantity expressing the classifiers relevance to particular environmental situations. The probalility that a bidding classifier C_j wins the competition is given by

$$\frac{Bid_j}{\sum_{C_l \in \mathcal{B}} Bid_l}$$

where $\mathcal{B}$ is the set of all bidding classifiers. A winning classifier C_j reduces its strength by the amount of its bid according to this expression,

$$Str_j = Str_j - Bid_j,$$

and this amount (Bid_j) is shared among those classifiers (predecessors) whose preceding activities enabled C_j to became active according to the following rule:

$$Str_i = Str_i + \frac{Bid_j}{|\mathcal{P}_j|} \quad \forall\, C_i \in \mathcal{P}_j,$$

where $\mathcal{P}_j$ is the set of predecessors, that is,

$$\mathcal{P}_j = \{\forall C_i \ : \ \exists T \in \mathcal{T}_j \ \exists m \in T \ (C_i \text{sent } m)\}$$

where $\mathcal{T}_j$ is the set of all messages tuples that satisfy C_j and m a message.

Obviously, if a classifier does not bid enough to win the competition, it pays nothing. Additionally, if an external reward is received from the environment then it is equally distributed among the classifiers that sent the effector-activating messages. In this way all classifiers directly or indirectly that are useful in achieving specific goals are rewarded although they are not active when the external reward is obtained. Some variants of BBA can be found in [Dor91,Wei91,Wil87a].

2. *The Profit Sharing Plan (PSP)* [Hol78,Gre88], which is a global learning scheme that typically achieves a clearly better performance than the BBA. Here, the bidding and selection of the winning classifiers is done as in BBA and the external reward Ext is distributed among sequences of active classifiers, that is, among classifiers C_j that was active at least one time during a episode (where an episode is defined as the time interval between the receipts of two succesive external rewards) according to rule

$$Str_j = Str_j - Bid_j + b \cdot Ext.$$

Therefore, the BBA is an incremental learning algorithm according to which the strengths are updated each cycle; against that, the PSP is an episodical learning algorithm according to which the strengths are only updated at the end of each episode. Some variants of PSP can be found in [Gre88,Wei92].

There exist some approaches to a synthesis of aforementioned CA schemes. For example the Grefenstette's system called RUDI [Gre88], and most recently, Gerhard Wei$\mathcal{B}$ has proposed a new CA system approach, *the Hierarchical Chunking Algorithm* (HCA) [Wei94], which approaches to both the lower computational requirements of the BBA and higher performance level of the PSP and, have been designed to solve the locality/globality dilemma (see [Wei94] for more information).

5.2.3 The Classifier Discovery System

The classifier discovery process for LCS uses AGs for its task. The system develops its learning process generating new classifiers from classifier set by means of AGs. Basically a GA selects high fitness classifiers as parents forming offsprings by recombining components from the parent classifiers. The fitness of a classifier is determined by its usefulness or strenght calculated with the CA system instead of a fitness function. In typical LCS implementations, the GA population is a portion of the classifier set consisting of those high strength ones. In order to preserve the system performance the GA is allowed to replace only a subset of the classifiers, i.e., a subset of the worst m classifiers is replaced by another of m new classifiers created by the application of the GA on the selected portion of the classifier population. As AG uses classifiers strength as a measure of fitness, it can be usefully applied to the set of classifiers only when the CA system has reached steady-state, i.e., when a classifier strength accurately reflects its usefulness. Therefore, it is applied with a lower frequency, usually between 1000 and 10000 CA system cycles. A AG basic execution cycle is as follows:

Step 1. Take the classifier set as initial population P.

Step 2. Rank individuals of P in decreasing fitness order using the strength associated to every classifier as a measure of fitness.

Step 3. Choose $2k$ individuals to be replaced among low ranked-useless-ones.

Step 4. Choose k pairs of individuals to be replicated among high ranked-useful-ones.

Step 5. Apply genetic operators to the k pairs selected at step 4, creating offspring classifiers.

Step 6 Replace the $2k$ individuals selected at step 3 with the offsprings created at step 5.

5.2.4 Basic Operation of a LCS

A LCS basic execution cycle that combines the aforementioned systems consists of the followings steps:

Step 0. Initially, a set of classifiers is created randomly or by some algorithm that takes into account the structure of the problem domain and they all have assigned the same strength.

Step 1. Allow the input interface to code the current environmental output signals as messages.

Step 2. Add all messages from the input interface to the message list.

Step 3. The pattern-matching system determines the set of classifiers that are matched by the current messages of the message list.

Step 4. The CR system resolves conflicts between matched classifiers and determines the set of active classifiers.

Step 5. Purge the message list.

Step 6. Place the messages suggested by the active classifiers on the message list.

Step 7. Allow any effectors that are matched by the current message list to submit their actions to the environment. In the case of inconsistent actions call to CR system.

Step 8. If a reward signal is present, distribute it with the CA system.

Step 9. If the CA system has reached steady-state applying GAs over classifier set.

Step 9. Return to step 1.

5.2.5 Future in LCSs Research

Nowadays there are defined many questions into LCSs research. In The First International Workshop on Learning Classifier Systems holded on October 6-9, 1992 at Johnson Space Center in Houston, Texas, were pointed out some main questions as:

1. What methods can create cooperation in LCS?.
2. What discovery mechanisms can help the GA to find and maintain useful classifier sets?.
3. What is an appropiate syntax or representation in LCS?
4. How can one insure effective credit assignment in LCS?
5. How can the LCS creates computationally useful internal message processing?

For more information see [Boo92,Smi92].

In the following section we shall study the appropiate representation problem and analize a type of LCS, called *the Fuzzy LCSs (FLCSs)*, which merge the ideas behind LCSs and fuzzy controllers, allowing to work with continous input and output variables due to the advanges of Fuzzy Sets Theory to deal imprecise information. Finally, in the last section we shall present some of the most important applications of LCSs in systems control.

5.3 Fuzzy Learning Classifier Systems

In many complex environments, e.g. in identification and adaptive control of dynamic systems, the LCSs have not had much application due in part to the limitations of the their syntax to represent continuously varying variables. A simple and promising way of dealing with this problem is through Fuzzy Sets Theory.

A Fuzzy Learning Classifier System is a genetic based machine learning system whose classifier list is a fuzzy rule base. The FLCS learns by creating fuzzy rules which relate the values of the input varibles to internal or output variables. The FLCS integrates the same elements of the LCS, but working in fuzzy environment.

Three fuzzy logic-based LCS approaches have been proposed, Valenzuela's approach [Val91a, Val91b], Parodi & Bonelli's approach [Par93] and Carse & Fogarty's approach [Car94]. In what follows, these approaches will be described.

5.3.1 The Valenzuela's Fuzzy Learning Classifier System

Valenzuela-Rendón gives the first description of a FLCS [Val91a, Val91b]. Some peculiarities of his FLCS model are:

- It operates over inputs, outputs and internal variables, and it allows them to take continuous values over given ranges.
- For each variable, n component fuzzy sets are defined so that their membership functions span the interval [0,1]. The number of these componet sets is defined by the user according to the precision required and membership functions are fixed and set by hand.
- The classifiers are fuzzy rules, similarly to fuzzy controllers. Each classifier is a binary string that encodes the membership function of the fuzzy sets defined for variables involved in the problem, so that the number of bits in a condition or an action is the number of fuzzy sets defined over the given variable. A "1" indicates that the corresponding fuzzy set is part of the condition or action.
- Each condition or action presents a non-fuzzy binary tag which indicates to which variable the condition or action is referring to.
- The FLCS syntax does not include the wildcard character #.
- There is defined a fuzzification process in the input interface of FLCSs, which fuzzifies inputs into fuzzy messages by creating minimal messages, one for each fuzzy set defined over the variable.
- Each message has an associated activity level which measures the degree of belonging of the input variable to the fuzzy set defined by the membership function represented by the message. The messages are deposited in message list.
- The pattern-matching system acts at two steps. First, the tags of the message and condition are compared; if they are the same they refer to the same variable, if so, the rest of the message and condition are compared. And second, if at least there is one position in which the condition has a "1" and the message also, then the condition is satisfied.

- When classifier conditions are satisfied, it posts a new message in the message list with an activity level proportional to the classifier own activity level. Where the activity level of a classifier is equal to the minimum of the satisfaction levels of all its conditions, being the satisfaction level of a condition equal to the maximum activity level of the messages that match this condition.

- When the output interface detects messages refered to output variables, it develops a defuzzification process, based on gravity center procedure, which translates them into real values.

- It has a CR/CA system which reassemble those of common learning classifier systems, i.e., the bucket brigade algorithm, but with a fuzzy nature.

- The classifiers that do not participate in the competition pays a living tax, (i.e. a small portion of their strength every cycle) whose purpose is to eliminate non-used classifiers.

- It uses a GA with the crossover and mutation operators to create new classifiers and therefore allows the evolution of adapted classifier set, but only the evolution of classifiers.

Valenzuela tested his FLCS in the identification of static one-input one-output systems using a stimulus-response FLCS without AC system, and he obtained good and promising results.

Some drawbacks of this approach are: (i) if the variables present many sets the classifiers can be very long; (ii) FLCS presents many parameters dependent on user; and (iii) membership functions are set and fixed by hand, which does not seem to be an easy task to perform in complex cases.

5.3.2 The Parodi & Bonelli's Fuzzy Learning Classifier System

This is a new and original approach to FLCSs that can be viewed as an extension of Wilson's Boole LCS [Wil87b] and which eliminates drawbacks of aforementioned approach. Some of its peculiarities are:

- Each variable has associated a fuzzy set, and i.e. each variable is described by a membership function. This description is variable and will evolve through genetic search. This approach suggest the use of symmetric membership functions, e.g. it chooses triangles.

- Each classifier contains the actual description of the membership functions that correspond to each input and output variable, which consists of parameters that define the associated fuzzy set. There is also an associated strength to each classifier that indicates its credibility.

- The pattern-matching system follows a process similar to a fuzzy controller process. A crisp input values vector is broadcast to the classifiers. All classifiers are activated in parallel to a different degree. The degree to which each classifier is activated is calculated by taking the minimum of the current inputs membership values with respect to the fuzzy sets present in the condition part of each classifier. All fuzzy sets of the action of each classifier are partially activated in parallel according to a proportional expresion to correpondient classifier activity degree.

- In the output interface the partially activated fuzzy sets of same output variables are combined using the weighted sum method to produce a final fuzzy set for each output variable. And finally, in order to produce numerical output values vector, each output fuzzy set is defuzzied according to fuzzy centroid procedure using the center of mass formula.

- Its CA system only works with positive rewards. It deducts a fraction of each active classifier strength and distributes a payoff quantity of the obtained reward to each active classifier strength according to a measure of goodness. This measure determines the quality of the classifiers action and the quality of the classifiers conditions for this particular input.
- Its classifier discovery system works with a GA that uses the crossover operator and a numerical creep mutation operator.
- Its main advantages are that it allows learning of membership functions as well as classifiers and it reduces the problems of classifiers representation.

Parodi & Bonelli tested their model with the same examples used by Valenzuela, and they obtained better results.

5.3.3 Caser & Fogarty's Fuzzy Learning Classifier System

Brian Carse & Terence C. Fogarty proposed in [Car94] a new FLCS using the Pittsburgh model. Some of its peculiarities are:

- In this FLCS genetic operators and strength asignment apply over classifier sets rather than over individual classifiers.
- The classifier sets representation is based on Parodi & Bonelli's representation, which allows the discovery component the learning of membership functions as well as fuzzy relations (classifiers).
- It includes variable length classifier sets.
- Its discovery component involves a numerical creep mutation operator and a new crossover operator based on weak positional dependence.

They tested also its system with the same examples used by Valenzuela, and obtained better results than Parodi & Bonelli's system and than Valenzuela's sytem.

The three aforementioned approaches have exhibited good results, but they have not been enough developed yet. Many issues remain open for research, see [Val91b, Par93, Car94].

5.4 Applications of LCSs in Control Processes

There exist many useful cases of LCSs application in control of real systems where they show good results. In this section we shall briefly comment some of main applications of LCSs in systems control.

In 1983 Goldberg applied a LCS to the control of two engineering systems: the pole-balancing problem and a natural gas pipeline-compressor systems. The simulations were in the stimulus-response format and with conventional structure of a LCS. For more information see [Gol83].

In robotic, Dorigo has developed a learning control system of a real robot, with great success. He has designed the system according to a parallel distributed model of learning based on LCSs and parallel machines (transputer networks). He decomposes the complex learning control problem of a robot in many simpler learning control problems. Each one of them is assigned to a LCS. Each component of LCS is implemented on a transputer network, each LCS is implemented as a subset of transputer nerwork and thus, the whole learning control system of robot is implemented as a set of transputer network subsets. In this way, the proposed system implements both a low-level parallelism within the structure of a single LCS (among its component systems) and a high-level type of concurrency, which allows various LCSs to work together by means of appropiate coordination strategies of competition and cooperation. These systems presents the following advantages:

- Increasing speed and flexibility of learning control system.
- Allowing to manipulate large sets of rules and therefore to face very complex learning control problems.
- Allowing to build more modular and efficient learning control systems.

For more information see [Dor92a, Dor92b, Dor93].

An FLCS application in control has been done by Furuhashi et al. in [Fur93]. They propose a method for suppressing excessive fuzziness in the FLCS for fulfilling complex tasks. The method uses multiple stimulus-response type Valenzuela's FLCSs. Simple simulations for learning to steer a simulated ship are done, showing its effectiveness in fulfilling a control task and for acquiring complex control rules.

Others interesting applications and studies can be found in [Wil83, Wil85, Gre87].

6 Conclusions

We have focused this paper on the description of GAs and FLCs as tools to model control processes. We have shown some GA applications to the design of FLCs, and the learning classifier systems and fuzzy learning classifier systems as tools for intelligent and adaptive control.

References

[Bak87] Baker, J.E., Reducing Bias and Inefficiency in the Selection Algorithm. Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1987, 14-21.

[Bel91] Belew, R.K., Booker, K.B. (Eds.), Proc. of the Fourth Int. Conf. on Genetic Algorithms, San Diego, 1991.

[Ber92] Berenji, H., Fuzzy Logic Controllers. In [Yag92], 69-96.

[Bon93] Bonarini, A., ELF: learning incomplete fuzzy rule sets for an autonomous robot. Proc. First European Congress on Fuzzy and Intelligent Technologies, Aachen, 1993, 69-75.

[Bon94] Bonissone, P.P., Fuzzy Logic Controllers: An Industrial Reality. In Computational Intelligence: Imitating Life, J.M. Zurada, R.J. Marks II, C.J. Robinson (Eds.), (IEEE Press, 1994) 316-327.

[Boo82] Booker, L.B., Intelligent Behavior as an Adpatation to the Task Environment. Doctoral Dissertation. Department of Computer and Communication Science, University of Michigan, Ann Arbor, 1982.

[Boo89] Booker, L.B., Goldberg, D.E., Holland, J.H., Classifier Systems and Genetic Algorithms. Artificial Intelligence 40 (1989) 235-282.

[Boo90] Booker, L.B., Representing Attribute-Based Concepts in a Classifier System. Proceeding of the First Workshop on Foundations of Genetic Algorithms, Morgan Kaufmann, 1990, 115-127.

[Boo92] Booker, L.B., Viewing Classifier Systems as an Integrated Architecture. Paper presented at The First Conference on Learning Classifier Systems, Houston, Texas, 1992.

[Bra91] Bramlette, M.F., Bouchard, E.E., Genetic Algorithms in Parametric Design of Aircraft. In [Dav91b], Chapter 10, 109-123.

[Car93] Cárdenas, E., Castillo, J.C., Cordón, O., Herrera, F., Peregrín, A., Influence of Fuzzy Implication Functions and Defuzzification Methods in Fuzzy Control. Busefal 57 (1993) 69-79.

[Car95] Cárdenas, E., Castillo, J.C., Cordón, O., Herrera, F., Peregrín, A., Applicability of T-norms in Fuzzy Control. Busefal 61 (1995) 28-37.

[Car94] Carse, B., Fogarty, T.C., A Fuzzy Classifier System Using the Pittsburgh Approach. In [Dav94], 260-269.

[Cas93] Castro, J.L., Delgado, M., Herrera, F., A learning method of fuzzy reasoning by ms. Proc. First European Congress on Fuzzy and Intelligent Technologies, Aachen, 1993, 804-809.

[Chw94] Chwee Ng, K., Li, Y., Design of Sophisticated Fuzzy Logic Controllers using Genetic Algorithms. Proc. of the Third IEEE Int. Conf. on Fuzzy Systems, Orlando, 1994, 1708-1712.

[Coo93] Cooper, M.G., Vidal, J.J., Genetic Design of Fuzzy Controllers. Proc. Second Int. Conf. on Fuzzy Theory and Technology, Durham, 1993.

[Dav91a] Davidor, Y., Genetic Algorithms and Robotics. A Heuristic Strategy for Optimization. World Scientific, London, 1991.

[Dav91b] Davis, L., Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, 1991.

[Dav94] Davidor, Y., Schwefel, H.P., Männer (Eds.), Parallel Problem Solving from Nature - PPSN III, Springer-Verlag, 1994.

[Dej80] De Jong K., Adaptive System Design: A Genetic Approach. IEEE Transactions on Systems, Man, and Cybernetics 10 (1980) 556-574.

[Dej88] De Jong K., Learning with Genetic Algorithms: An Overview. Machine Learning 3 (1988) 121-138.

[Dor91] Dorigo, M., New Perspectives about Default Hier Formation in Learning Classifier Systems. Technical Report No. 91-002. Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[Dor92a] M. Dorigo, Using transputers to increase speed and flexibility of genetics-based machine learning systems. Microprocessing and Microprogramming J. vol. 34 (1992) 147-152.

[Dor92b] M. Dorigo, ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems. Technical Report No. 92-011, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[Dor93] M. Dorigo and U. Schnepf, Genetics-based machine learning and behavior base robotics: A new synthesis. IEEE Transactions on Systems, Man, and Cybernetics, 23, 1 (1993) 141-153.

[For93] Forrest, S. (Ed.), Proc. of the Fifth Int. Conf. on Genetic Algorithms. University of Illinois at Urbana-Champaign, 1993.

[Fur93] Furuhashi T., Nakaoka K., Morikawa K., Uchikawa Y., Controlling execessive fuzziness in a fuzzy classifier system. Proceedings of the Fifth International Conference on Genetic Algorithms, 1993, 635.

[Gol83] Goldber, D.E., Computer-aided gas pipeline operation using genetic algorithms and rule learning, Ph.D. Dissertation, University Microfilms No. 8402282, University of Michigan, Ann Arbor, MI (1983).

[Gol89] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York, 1989.

[Gre86] Grefenstette, J.J., Optimization of Control Parameters for Genetic Algorithms. IEEE Transactions on Systems, Man, and Cybernetics 16 (1986) 122-128.

[Gre87] Grefenstette, J.J., Genetic Algorithms and Their Applications, Erlbaum, Hillsdale, NJ, 1987.

[Gre88] Grefenstette, J.J., Credit Assignment in Rule Discovery Systems based on Genetic Algorithms. Machine Learning 3 (1988) 225-245.

[Gre93] Grefenstette, J.J. (Ed.), A Special Issue of Machine Learning, 13 (1993) n. 2-3.

[Gre94] Grefenstette, J.J. (Ed.), Genetic Algorithms for Machine Learning. Kluwer Academic, Boston, 1994.

[Har93] Harris, C.J., Moore, C.G., Brown, M., Intelligent Control. Aspects of Fuzzy Logic and Neural Networks. World Scientific, London, 1993.

[Hel93] Hellendoorn, H., Driankov, D. & Reinfrank, M., An Introduction to Fuzzy Control. Springer-Verlag, 1993.

[Her93] Herrera, F., Lozano, M., Verdegay, J.L., Genetic Algorithm Applications to Fuzzy Logic Based Systems. Proc. of the 9yh Polish-Italian and 5th Polish-Finnish Symposium on Systems Analysis and Decision Support in Economics and Technology, Warsaw, 1993, 125-134.

[Her94] Herrera, F., Lozano, M., Verdegay, J.L., Generating Fuzzy Rules from Examples using Genetic Algorithms. Proc. of the Fifth Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, 1994, 675-679.

[Her95a] Herrera, F., Lozano, M., Verdegay, J.L., Tuning Fuzzy Controllers by Genetic Algorithms. To appear in International Journal of Approximate Reasoning, 1995.

[Her95b] Herrera, F., Lozano, M., Verdegay, J.L., A Learning Process for Fuzzy Control Rules Using Genetic Algorithms. Technical Report DECSAL-95108, Dept. of Computer Science and A.I., University of Granada, Spain, 1995.

[Hes93] Hessburg, T., Lee, M., Takagi, H., Tomizuka, M., Automatic Design of Fuzzy Systems using Genetic Algorithms and its Application to Lateral Vehicle Guidance. SPIE's Int. Symposium on Optical Tools for Manufacturing and Advanced Automation, Vol. 2061, Boston, 1993.

[Hir93] Hirota, K., Industrial Applications of Fuzzy Technology. Springer-Verlag, 1993.

[Hof94] Hoffmann, F., Pfister, G., Automatic Design of Hie Controllers using Genetic Algorithms. Proc. of the Second European Congress on SoftComputing and Intelligent Technologies, Aachen, 1994, 1516-1522.

[Hol75] Holland J.H., Adaptation in Natural and Artificial Systems. Ann arbor: The University of Michigan Press, 1975 (The MIT Press, London, 1992).

[Hol76] Holland, J.H., Adaptation. In: R.Rosen, F.M. Snell (Eds), Progress in Theorical Biology, 4. New York: Plenum, 1976.

[Hol78] Holland, J.H., Reitman, J.S., Cognitive Systems Based on Adaptive Algorithms. In: D.A. Waterman, F. Hayes-Roth (Eds), Pattern-Directed Inference Systems. New York, Academic Press, 1978.

[Hol85] Holland, J.H., Properties of the Bucket Brigade Algorithm. Proceedings of the First International Conference on Genetic Algoritms and Their Applications, Pittsburgh, PA: Erlbaum 1985, 1-7.

[Hol86] Holland, J.H., Holyoak, K.J., Nisbett, R.E., Thagard, P.R., Induction. Processes of Inference, Learning, and Discovery. The MIT Press, London, 1986.

[ICEC94] Proc. of the First IEEE Conference on Evolutionary Computation, Orlando, 1994.

[Kar91a] Karr, C., Genetic Algorithms for Fuzzy Controllers. AI Expert, February 1991, 26-33.

[Kar91b] Karr, C., Applying Genetic Algorithms to Fuzzy Logics. AI Expert, March 1991, 38-43.

[Kar91c] Karr, C.L., Air-Injected Hydrocyclone Optimization via Genetic Algorithms. In [Dav91b], Chapter 16, 222-236.

[Kis85] Kiszka, J., Kochanska, M. & Sliwinska, D., The influence of Some Fuzzy Implication Operators on the Accuracy of a Fuzzy Model - Parts I and II. Fuzzy Sets and Systems, 15 (1985) 111-128, 223-240.

[Koz92] Koza, J., Genetic Programming. On the Programming of Computers by Means of Natural Selection. The MIT Press, London, 1992.

[Lee90] Lee, C.C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Parts I and II. IEEE Transactions on Systems, Man and Cybernetics 20 (1990) 404-435.

[Lee93a] Lee, M.A., takagi, H., Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques. Proc. of Fifth Int. Conf. on Genetic Algorithms, Urbana-Champaign, 1003, 1993, 76-83.

[Lee93b] Lee, M.A., Takagi, H., Embedding Apriori Knowledge into an Integrated Fuzzy System Design Method Based on Genetic Algorithms. Proc. of The Fifth IFSA World Congress, Seoul, 1993, 1293-1296.

[Lei94] Leitch, D., Probert, P., Context Dependent Coding in Genetic Algorithms for the Design of Fuzzy Systems. Proc. of IEEE/Nagoya University WWW on Fuzzy Logic and Neural Networks/Genetic Algorithms, Nagoya, 1994.

[Lie91] Liepins, G.E., Potter, W.D., A Genetic Algorithm Approach to Multiple-Fault Diagnosis. In [Dav91b], Chapter 17, 237-250.

[Mam75] Mamdani, E.H. Assilian, S., An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies 7 (1975) 1-13.

[Mam93] Mamdani, E.H., Twenty years of Fuzzy Control: Experiences Gained and Lessons Learnt. Fuzzy Logic Technology and Applications. R.J. Marks II (Eds.), IEEE Press, 1993, 19-24.

[Män92] Männer, R., Manderik, B., Proc. of the Second Conference on Parallel Problem Solving from Nature, 2, Brussels, 1992.

[Mic92] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York, 1992.

[Nom92] Nomura, H., Hayashi, I., Wakami, N., A Learning Method of Simplified Fuzzy Reasoning by Genetic Algorithm. Proc. of the Int. Fuzzy Systems and Intelligent Control, Lousville, 1992, 236-245.

[Par93] Parodi, A., Bonelli, P., A New Approach to Fuzzy Classifier System. Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, 1993, 223-230.

[Sat94] Satyadas, A., Krishnakumar, GA-optimized Fuzzy Controller for Spacecraft Attitude Control. Proc. of the Third IEEE Int. Conf. on Fuzzy Systems, Orlando, 1994, 1979-1984.

[Smi80] Smith, S.F., A Learning System Based on Genetic Adaptive Algorithms. Doctoral Dissertation, Department of Computer Science, University of Pittsburgh, PA, 1980.

[Smi91] Smith, R.E., Default Hierarchy Formation and Memory Exploitation in Learning Classifiers Systems. Ph. D. thesis, The University of Alabama, Tuscaloosa, Al, 1991.

[Smi92] Smith, R.E., A Report on The First International Workshop on Learning Classifier Systems, Houston, Texas, 1992.

[Sug85] Sugeno, M., An Introduction Survey of Fuzzy Control. Information Science 36 (1985) 59-83.

[Sur93] Surmann, H., Kanstein, A., Goser, K., Self-Organizing and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems. Proc. First European Congress on Fuzzy and Intelligent Technologies, Aachen, 1993, 1097-1104.

[Sys91] Syswerda, G., Schedule Optimization Using Genetic Algorithms. In [Dav91b], Chapter 21, 332-349.

[Thr91] Thrift, P., Fuzzy Logic Synthesis with Genetic Algorithms. Proc. of the Fourth Int. Conf. on Genetic Algorithms. Morgan Kaufmann, 1991, 509-513.

[Umb80] Umbers, I.G., King, P.J., An Analysis of Human-Decision Making in Cement Kiln Control and the Implications for Automation. International Journal of Man-Machine Studies 12 (1980) 11-23.

[Val91a] Valenzuela-Rendón, M., The Fuzzy Classifier System: Motivations and First Results. Parallel Problem Solving from Nature II, (Springer Verlag, Berlin, 1991) 330-334.

[Val91b] Valenzuela-Rendón, M., The Fuzzy Classifier System: A Classifier System for Continuously Varing Variables. Proc. of the Fourth Int. Conf. on Genetic Algorithms, San Mateo, CA: Morgan Kauffmann, 1991, 346-353.

[Var93] Varsek, Urbancic, T., Filipic, B., Genetic Algorithms in Controller Design and Tuning. IEEE Transactions on Systems, Man, and Cybernetics 23 (1993) 1330-1339.

[Wei91] Wei$\mathcal{B}$, G., The Action-Oriented Buchket Brigade. Technical Report FKI-156-91. Institut für Informatik, Technische Universität München, 1991.

[Wei92] Wei$\mathcal{B}$, G., Learning the Goal Relevance of Actions in Classifier Systems. Proceedings of the 10th European Conference on Artificial Intelligence, 1992, 430-434.

[Wei94] Wei$\mathcal{B}$, G., The Locality/Globality Dilemma in Classifier Systems and an Approach to its Solution. Technical Report FKI-187-94. Institut für Informatik, Technische Universität München, 1994.

[Wil83] Wilson, S.W., On the retinal-cortical mapping. Int. Journal Man-Manc. Stud. 18 (1983) 361-389.

[Wil85] Wilson, S.W., Knowledge growth in an artificial animal. Proceedings of a International Conference on Genetic Algorithms and Their Applications, Pittsburgh, 1985, 16-23.

[Wil87a] Wilson, S.W., Hierarchical Credit Allocation in a Classifier System. In Genetic Algorithm and Simulated Annealing, L. Davis (Ed.), (CA: Morgan Kaufmann, Los Altos, 1987) 104-105.

[Wil87b] Wilson, S.W., Classifier Systems and the Animat Problem. Machine Learning Journal 2 (1987) 199-228.

[Wil94] Wilson, S.W., ZCS: A Zeroth Level Classifier System. Evolutionary Computation Vol. 2, No 1 (1994) 1-18.

[Yag92] Yager, R., Zadeh, L.A. (Eds.), An Introduction to Fuzzy Logic Applications in Intelligent Systems. Kluwer Academic, Boston, 1993.

[Zad65] Zadeh, L.A., Fuzzy Sets. Information and Control 8 (1965) 338-353.