Fuzzy-XCS: A Michigan Genetic Fuzzy System

Jorge Casillas, Brian Carse, *Member, IEEE*, and Larry Bull

***Abstract*—The issue of finding fuzzy models with an interpretability as good as possible without decreasing the accuracy is one of the main research topics on genetic fuzzy systems. When they are used to perform online reinforcement learning by means of Michigan-style fuzzy rule systems, this issue becomes even more difficult. Indeed, rule generalization (description of state-action relationships with rules as compact as possible) has received a great attention in the nonfuzzy evolutionary learning field (e.g., XCS is the subject of extensive ongoing research). However, the same issue does not appear to have received a similar level of attention in the case of Michigan-style fuzzy rule systems. This may be due to the difficulty in extending the discrete-valued system operation to the continuous case. The intention of this contribution is to propose an approach to properly develop a fuzzy XCS system for single-step reinforcement problems.**

***Index Terms*—Continuous action, genetic fuzzy systems, Michigan-style learning classifier systems, reinforcement learning.**

I. Introduction

THE Michigan-style genetic fuzzy rule-based system [1] is a machine learning system which employs linguistic rules and fuzzy sets in its representation and an evolutionary algorithm (EA) for rule discovery. It therefore combines an easily understood representation (as opposed to, for example, neural network approaches) with a general purpose search method. These systems are very useful for performing online learning, i.e., to automatically learn fuzzy rules and act on the environment at the same time that data or stimulus (or reward or payoff) are received. This fact makes Michigan-style genetic fuzzy systems ideal for reinforcement learning, adaptive systems, control, simulation of animal behavior and, in addition, data mining and knowledge discovery applications.

In order to exploit the fuzzy representation to the full, i.e., to achieve high interpretability, the ability to learn generalization is of great importance. With *generalization*,[1] we understand in this paper capability to express the state-action (antecedent-consequent) relationships as compact as possible. Generalized rules allow more compact rule bases, scalability to higher dimensional spaces, faster inference, and better linguistic interpretability. The issue of rule generalization, and the interplay between general and specific rules in the same evolving population, has received a great attention in the evolutionary learning research community (e.g., [2]).

Traditional (nonfuzzy) evolutionary learning algorithms for reinforcement learning have been "strength-based" in the sense that a rule (classifier) accrues strength during interaction with the environment (through rewards and/or penalties). This strength can then be used for two purposes: resolving conflicts between simultaneously matched rules during learning episodes; and as the basis of fitness for the EA. A completely different approach can be taken in which a rule's fitness, from the point of view of the EA, is based on its "accuracy," i.e., how well a rule predicts payoff whenever it fires. Note that the concept of accuracy used here is different from the traditionally used in fuzzy modeling (i.e., capability of the fuzzy model to faithfully represent the modeled system). Broadly speaking, and with the aim of explaining better the difference between the two measures, we could say that the strength value is the mean of the obtained payoffs while the accuracy value is the corresponding standard deviation.

This accuracy-based approach offers a number of advantages such as avoiding overgeneral rules, obtaining optimally general rules, and learning of a complete "covering map." The first accuracy-based evolutionary algorithm, called XCS, was proposed in [2] and it is currently of major interest to the research community in this field. However, no successful accuracy-based approaches have been proposed in the case of Michigan-style fuzzy rule-based systems [3]–[11]. On the contrary, all of them are strength-based.

This work aims at proposing a new approach to achieve accuracy-based Michigan-style genetic fuzzy rule-based systems. The proposal, Fuzzy-XCS, is based on XCS but properly adapted to fuzzy systems. The proposed system interacts with the environment by means of continuous actions, a point of great interest in reinforcement learning. That aspect, together with the fact of using a highly interpretable compact representation of the state-action policy that maximizes the payoff, provides our system with some competitive advantages compared with other reinforcement learning methods. An accuracy-based fuzzy rule-based system is proposed in [12]. However, it presents a number of important limitations (mainly the lack of the generalization capability and the use of integer-valued output) that are faced by our method. Furthermore, we can not forget those non-evolutionary algorithms based on fuzzy Q-learning [13], [14], which combine Q-learning (a popular reinforcement learning technique) and fuzzy rule-based systems to partition the Q-function (that estimates the future rewards of

Manuscript received November 1, 2005; revised May 29, 2006 and March 13, 2007. This work was supported in part by the Spanish Ministry of Science and Technology under Grant TIN2005-08386-C05-01 and by the European Regional Development Fund.

J. Casillas is with the Department Computer Science and Artificial Intelligence, University of Granada, Granada E-18071, Spain (e-mail: casillas@decsai.ugr.es).

B. Carse and L. Bull are with the Faculty Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol BS16 1QY, U.K. (e-mail: brian.carse@uwe.ac.uk; larry.bull@uwe.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2007.900904

[1]Do not mistake it with the generalization concept used in classification or system identification in general, i.e., the capability to successfully predict the correct answer on unknown patters that were not used during the learning process.

taking an action in a specific state) with fuzzy sets. However, fuzzy Q-learning still considers a tabular form that lacks of generality and compact knowledge representation and, in most cases, discrete action is considered.

The paper is organized as follows. Section II reflects about Michigan and Pittsburgh styles, introduces XCS, reviews Michigan-style fuzzy rule-based systems, and outlines some advantages and difficulties of an accuracy-based approach. Section III introduces the proposed Fuzzy-XCS algorithm. Section IV shows some experimental results. Finally, Section V concludes.

II. Background

A. Michigan or Pittsburgh Style in Evolutionary Learning

In this section, we summarize and discuss two alternative ways in which the EA may be applied to learn rules. These two methods, Michigan and Pittsburgh approaches, were first described as long ago as 1978 and 1980, respectively. It should be stated that both approaches are the subject of ongoing research and many significant extensions have been devised and used as the basis for successful learning systems. We begin our discussion by outlining the Michigan approach, since this was employed in the first published report of evolutionary learning system.

1) The Michigan Approach: The first Michigan-style evolutionary learning system was Cognitive System One (CS-1) devised by Holland and Reitman [15]. CS-1 maintains a population of rules with genetic operations and credit assignment applied at the level of the individual rule. Each rule in the population has an associated strength, which is used to store an accumulation of credit. The original CS-1 credit apportionment algorithm is epoch-based, where rules activated since the last payoff event share the reward collected from the environment at the next.

Since CS-1, a large number of alternative credit assignment schemes have been proposed, most notably the bucket-brigade [16] and Q-learning [17]–[19] for dealing with environments where reward may be infrequent and/or delayed. These proposed credit assignment schemes have achieved a great deal of success, although many problems regarding their use remain the focus of research.

The EA in a Michigan-style evolutionary learning system operates at the level of the individual rule with selection of parent rules for mating based on strengths (and in some cases other parameters such as rule age or relevance). In addition, rule strengths in discrete Michigan-style systems are commonly used in controlling the dynamic behavior of the rule system by forming the basis for conflict resolution between simultaneously matched rules. In the fuzzy case, some practitioners use rule strengths as weights which influence the level of contribution of rule consequents.

2) The Pittsburgh Approach: In 1980, Smith [20] published results of an alternative evolutionary learning system, LS-1, in which the unit of genetic manipulation is a suitably encoded genotype representing a complete set of rules. Credit is assigned to complete sets of rules via interaction with the environment. This typifies so-called "Pittsburgh"-style evolutionary learning systems. Since the complete rule set is the basis of credit apportionment, Pittsburgh-style evolutionary learning systems sidestep completely the potentially knotty problem of sharing out credit to individual rules. The EA in LS-1 operates at different levels: at the highest level, complete rule-sets are selected as the basis for reproduction to generate new rule-sets; at the lowest level individual rules are chosen by the EA to generate new rules. For the purposes of this discussion we are primarily concerned with the highest of these levels of reproduction. LS-1 uses variable length rule sets, and employs modified genetic operators for dealing with these variable-length, position independent (as far as phenotypic expression is concerned) genomes.

3) Differences Between Both Approaches: Clearly the role of the EA in Pittsburgh and Michigan approaches is rather different, and the distinction arises from the difference in level at which the EA is applied. Both approaches, at least in their simplest forms, suffer from distinct, known problems which arise from the different way in which the EA is applied.

The major problem in the Michigan approach is that of resolving the conflict between the individual and collective interests of single rules within the system. The ultimate aim of a evolutionary learning system is to evolve a set of co-adapted rules which act together in solving some problem. In a Michigan style system, with selection and replacement at the level of the individual rule, rules which cooperate to effect good actions and receive payoff also compete with each other under the action of the EA. Such a conflict between individual and collective interests of individual rules does not arise with Pittsburgh-style evolutionary learning systems, since reproductive competition occurs between complete rule-sets rather than individual rules.

However, maintenance and evaluation of a population of complete rule-sets in Pittsburgh-style systems can often lead to a much greater computational burden (in terms of both memory and processing time). Wilson and Goldberg [21] propose an evolutionary learning system which clusters rules into "corporations." Rules belonging to the same corporation do not compete with each other under the action of the EA and corporations form and break up under the action of a modified crossover operator. A number of successful implementations of this approach were demonstrated by Tomlinson and Bull (e.g., [22]) wherein rules which match on consecutive states can link genetically via a mutation-like operator. Such corporations also have the potential to solve sensory ambiguities within a problem, i.e., act as a form of memory for non-Markov environments [23]. Such approaches represent a middle ground between Michigan-style and Pittsburgh-style evolutionary learning systems.

Problems with the Pittsburgh approach have proved to be at least equally as challenging. Although the approach avoids the problem of explicit competition between rules, large amounts of computing resources are required to evaluate a complete population of rule-sets. A further problem with the approach is the small bandwidth of reinforcement information, usually a single scalar fitness value for each complete rule-set. If information about the performance of individual rules happens to be derivable from the pattern of environmental payoffs using some credit apportionment method, this information is not

explicitly exploited in the Pittsburgh approach. The disruptive threat to good collections of cooperating rules comes from a different source in Pittsburgh-style systems compared to Michigan-style systems. In the latter, competition at the level of selection and replacement of individual rules can destroy good rule associations. In a Pittsburgh-style system, although selection and replacement will automatically favor co-adapted rule-sets, crossover can be a major cause of disruption of cooperating collections of rules since the operator is blind to such associations between rules.

An elegant solution to both the problems of coarse-grained credit assignment and the disruptive effects of crossover in Pittsburgh systems is proposed by Grefenstette [24] using hierarchical credit assignment. With this method, credit is assigned to individual rules as well as to complete rule-sets. Prior to crossover, the genome encoding the rule-set is ordered so that high strength rules occupy neighboring loci on the genome. If the underlying assumption that co-adapted rules accrue similar strengths is valid, then crossover is less likely to disrupt these rule associations than if an unordered genome is employed.

B. XCS: An Accuracy-Based Michigan-Style Algorithm

Most current Michigan-style research has made a shift away from strength (payoff), after Wilson introduced XCS [2]. XCS is a Michigan-style system which uses the accuracy of rules predictions of expected payoff as their fitness. In this way a full map of the problem space is created, rather than the traditional search for only high payoff rules, with (potentially) maximally accurate generalizations over the state-action space. That is, XCS uses a genetic algorithm (GA) [25] to evolve generalizations over the space of possible state-action pairs with the aim of easing the use of such approaches in large problems through a triggered niche mechanism.

The general technique was introduced by Booker [26], who based the trigger on a number of factors including the payoff prediction "consistency" of the rules in a given niche, to improve the performance of Michigan-style systems. The scheme was motivated by observation of the disruptive effects of breeding dissimilar rules [27], i.e., the recombination of rules which match very different parts of the input space can cause the loss of useful generalizations. XCS uses a time-based mechanism under which each rule maintains a time-stamp of the last system cycle upon which it was part of a GA. The GA is applied within the current niche when the average number of system cycles since the last GA in the set is over a threshold θ_{GA}. If this condition is met, the GA time-stamp of each rule in the niche is set to the current system time, two parents are chosen according to their fitness using standard roulette-wheel selection, their offspring are potentially crossed and mutated, before being inserted into the rule-base.

When introducing XCS, Wilson [2] highlighted how the triggered niche GA leads to a tendency for accurate rules which participate in more niches than other similarly accurate rules to takeover. That is, if two rules are of equal accuracy and one is more general than the other, the more general rule will participate in more niches and therefore have more chances of reproduction—the generalization hypothesis. Wilson has described the niche GA of XCS as searching along a line in the space of possible generalizations, from completely specific to completely general, for each action "driven by a fitness measure, accuracy, that is strongly correlated with specificity" [28].

XCS can also avoid problematic overgeneral rules which receive a high optimal payoff for some inputs but are suboptimal for other, lower payoff, inputs. Since their average payoff is higher than that for the optimal rules in the latter case the overgenerals tend to displace them, leaving the system suboptimal. However, the payoffs received by overgeneral rules typically have high variance (they are inaccurate predictors) and so have low fitness in XCS. Versions of Holland's system were shown to suffer due to such rules emerging.

More recently, XCS has been shown to perform well in a number of real-world domains [29] and more formal understanding of its workings are beginning to emerge [30].

C. Related Work: Strength-Based Michigan-Style Genetic Fuzzy Systems

The first description of a Michigan-style genetic fuzzy system is given in [3]. Closely modeled on the discrete-valued Holland-style classifier system this system contains a fixed size rule-base of fuzzy rules and a fuzzy message list. Inference is carried out using a concept of "minimal messages." The system is applied to a function learning task, and credit is allocated to individual rules according to how closely each rule predicts the correct output. The payoff distribution scheme is therefore, as the author states, not pure reinforcement learning. True reinforcement learning is introduced to the system by the same author in [8].

Parodi and Bonelli [4] present a genetic fuzzy system which automatically learns fuzzy relations, fuzzy membership functions, and rule weights. The rule population consists of a fixed size list of rules. Each rule has associated with it an "output weight" which is set equal to the strength (fitness) of that rule. The rule strength therefore performs a dual function: first, it forms the basis of selection and replacement for the EA and second, it allows stronger rules to take a bigger part in decision making than weaker ones. In this system, "don't care" symbols are not allowed and the capabilities for generalization of the rule system are via learning appropriate widths of triangular membership functions.

Furuhashi *et al.* [5] propose a method using multiple stimulus-response fuzzy rules operating in tandem. Message passing between rule systems is via "crisp" values in order to control excess fuzziness, which, the authors argue, can degrade system performance. The system is applied to a simulated ship which is to reach a given target while avoiding moving obstacles. The controller is implemented as three separate, communicating rule systems, each operating at a different level of a controller hierarchy. In Nakaoka *et al.* [6], the same ship steering problem is attempted with a single rule list. This work describes problems in coverage in moving to high dimensional spaces. This is overcome by using a dual fitness—one based on environmental payoff, the other based on the accumulation of the level of activation during a simulation; this ensures selection pressure towards both high-reward rules and well-matched rules.

Velasco [9] describes a Michigan-style genetic fuzzy system designed specifically for online learning for fuzzy process control. This work introduces the concept of "limbo"—a special workspace where newly generated rules are evaluated prior to them being used to control the real process plant. The purpose of "limbo" is to test rules before insertion into the working rule list, to avoid the use of bad, newly-generated rules on the actual control system.

Ishibuchi *et al.* [10] apply a Michigan-style genetic fuzzy system to the problem of pattern classification. A fixed sized rule population is used. "Don't care" symbols are used for generalized fuzzy rules. The rule consequents are an output class (for the classification problem considered) and a certainty factor. These latter are derived using a heuristic procedure prior to fitness evaluation, and the EA operates on the rule antecedent only. Credit assignment methods are incorporated which explicitly address the classification problem. Recently, some of these authors [31] have also proposed a hybridation of both Michigan and Pittsburgh approaches by applying with a predefined probability a single iteration (rule generation and replacement) of a Michigan-style algorithm to each individual of a Pittsburgh-style algorithm (i.e., a fuzzy rule set).

The classic "competition versus cooperation" problem is directly addressed in the case of genetic fuzzy systems in Bonarini's works [7], [11] which propose ways to deal with the problem in a Michigan-style algorithm called ELF. In ELF, the EA operates on subpopulations of fuzzy rules with the same antecedent but different consequent. Bonarini's genetic fuzzy system has been applied to a wide range of learning tasks, with extensive evaluation of different reinforcement learning algorithms. Instability of generalized rules during the learning process is controlled by distributing to each rule a reinforcement normalized on the difference between the maximum and minimum reinforcement obtained by the subpopulation(s) which that rule is part of.

D. Advantages and Difficulties in "Accuracy-Based" Fuzzy Rule-Based Systems

All Michigan-style genetic fuzzy systems described previously are based on the strength. As mentioned in Section I, a completely different approach based on accuracy can be taken. Such an accuracy-based approach offers a number of advantages:

- First, it can distinguish between accurate and overgeneral rules: an overgeneral rule will have relatively low accuracy since payoff will vary according to the input states covered by the rule.
- Indeed, it has been shown (in the discrete-valued case) that the accuracy-based approach can lead to evolution of optimally general rules.
- Additionally it can maintain both consistently correct and consistently incorrect rules which allows learning of a complete "covering map."

The reason why accuracy-based genetic fuzzy systems has not successfully been proposed yet mainly resides in the great difficulty of doing that:

- First, in a traditional fuzzy rule-based system several rules fire in parallel (this is how the system achieves interpolation); credit assignment is much more difficult in the fuzzy case and it may well be that apportioning credit in proportion to a fuzzy rule's activation level is not appropriate.
- A further difficulty is measuring the accuracy of a rule's predicted payoff since (particularly early in the search) a fuzzy rule will fire with many different other fuzzy rules at different time-steps, giving very different payoffs.
- Yet another difficulty is that the payoff a fuzzy rule receives depends on the input vector—an active fuzzy rule will receive different payoffs for different inputs. This further complicates payoff predictions used as the basis for accuracy-based fitness.

III. FUZZY-XCS

This section describes the proposed accuracy-based Michigan algorithm, called Fuzzy-XCS. Fig. 1 shows a scheme of the algorithm. Basically, the online behavior consists of two cycles: action and learning.

- The action cycle is as follows. When a new state is received from the environment, the following sequence is performed: to generate the *match set*, to obtain *candidate subsets*, to choose the *action set* and, finally, to *infer* the output and act on the environment.
- The learning cycle starts when a reward is received from the environment. The process is done on the action set that performed the action from which this reward has been obtained. The process involves to obtain the *payoff*, to do the *credit distribution*, to *update the values* of error, prediction, fitness and experience, and to apply the *discovery component* (covering, selection, crossover and mutation) in order to generate new fuzzy rules to be included in the population.

The following subsections explain in detail the components of the algoritm.

A. Generalization Representation

First of all, a representation of fuzzy rules to allow proper generalization must be done. We propose the use of *disjunctive normal form* (DNF) fuzzy rules with the following structure:

$$\textbf{IF } X_1 \text{ is } \widetilde{A_1} \text{ and } \ldots \text{ and } X_n \text{ is } \widetilde{A_n} \textbf{ THEN } Y \text{ is } B$$

where each input variable X_i takes as a value a set of linguistic terms $\widetilde{A_i} = \{A_{i1} \vee \ldots \vee A_{il_i}\}$, whose members are joined by a disjunctive (T-conorm) operator, whilst the output variable remains a usual linguistic variable with a single label associated.

This structure uses a more compact description that allows rules with different generalization degrees. Moreover, the structure naturally supports the absence of some input variables in each rule (simply making $\widetilde{A_i}$ be the whole set of linguistic terms).

In order to use this representation in Fuzzy-XCS, we propose a binary coding scheme for the antecedent of the fuzzy rule with size equal to the sum of the number of linguistic terms used in each input variable. The allele '1' means that the corresponding linguistic term is used in the corresponding variable. For the consequent of the rule, an integer coding scheme is used where each gene contains the index of the linguistic term used for the corresponding output variable. For example, assuming we have

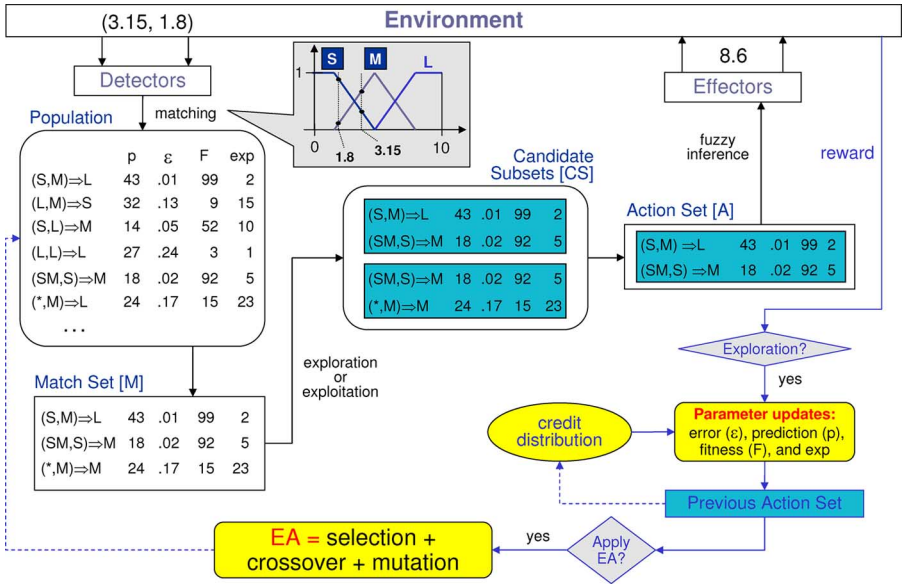

Fig. 1. Fuzzy-XCS scheme.

three linguistic terms (S [small], M [medium], and L [large]) for each input/output variable, the fuzzy rule [IF X_1 is S and X_2 is {M or L} THEN Y_1 is M and Y_2 is L] is encoded as [100|011||23].

B. Performance Component

In XCS [2], the performance component consists of three modules: match set construction, prediction array computation, and action set selection. This process has the final objective of inferring a specific action from the set of rules that matches the current state. In Fuzzy-XCS the process is different, as described as follows:

- *Match set construction* ([M]): The match set is build with all the fuzzy rules with a matching degree greater that zero for the given state.
- *Computation of candidate subsets* ([CS]): This stage might be considered as equivalent to the prediction array computation. XCS partitions [M] into a number of mutually exclusive sets according to the action of each rule. However, in real-valued output systems like Fuzzy-XCS, several "linguistic actions" (consequents) could/should be considered together. Thus, Fuzzy-XCS redefines the concept of prediction array computation.
  We can assume that what should not be accepted in our case is to have an action set with inconsistent or redundant rules, i.e. rules with equal or subsumed antecedent and different consequent (inconsistency), or rules with subsumed antecedent and equal consequent (redundancy). Note that identical rules are allowed to belong to the action set since we consider macroclassifiers [2] where the numerosity of a rule is keep instead several "physical" copies of it. Therefore, when a rule is selected, all the copies of this individual are automatically considered and used in the different calculations done in the performance and discovery components.
  Different groups of consistent and non-redundant fuzzy rules (S_i) with the maximum number of rules in each group are formed. To do that we consider a simple greedy algorithm (though an implicit enumeration algorithm could be used) as follows:

```
Let R = {R_i / R_i ∈ [M]}
Shuffle R {to avoid order-biasing}
for i = 1 to |R| do
 S_i ← {R_i}
 for j = 1 to |R| (j ≠ i) do
      if (S_i ∪ {R_j} is consistent and
      nonredundant) then
           S_i ← S_i ∪ {R_j}
      end if
 end for
end for
Remove repeated S_i
```

We perform an *exploration/exploitation* scheme with probability 0.5. On each exploitation step, only those fuzzy rules sufficiently experienced (they have been adjusted—see Section III-C2—in a degree greater or equal to θ_{exp}) are considered. The most experienced rule is considered when any one holds this condition. On exploration step, the whole match set is considered.
Using only experienced rules during exploitation steps allows the system to give the best action according to the available knowledge, while on exploration steps the system is able to learn or corroborate rules. Note that though this is a new interpretation of Wilson's XCS scheme, we are addressing the fuzzy context where we should not decide between one action or another—all fuzzy rules should act together in order to give a sound real-valued output. So, in Fuzzy-XCS the exploration/exploitation concept is moved from the action selection framework to the rule selection one.

- *Action set selection* ([A]): The action set selection chooses the consistent and non-redundant rule subset (S_i) with the highest mean prediction (considering the numerosity of each rule when computing the mean).
- *Action generation*: The action taken to interact with the environment is directly inferred from the action set. When using DNF-type fuzzy rules, special care must be taken on the inference engine. Indeed, for a proper behavior of the algorithm, it is mandatory to ensure that given two linguistically equivalent fuzzy rule sets, they are also numerically equivalent. In order to do so, we consider FATI (first aggregate, then inference) approach, the Max-Min scheme (i.e., T-conorm of *maximum* as aggregation and T-norm of *minimum* as implication operator), and the T-conorm of *maximum* as disjunction. Apart from that, T-norm of *minimum* as conjunction and *center-of-gravity* as defuzzification are used. These two latter operators could be changed without incurring in a linguistic-numeric discrepancy.

C. Reinforcement Component

The p_j (prediction), ϵ_j (prediction error), and F_j (fitness) values are adjusted by the reinforcement learning standard technique Widrow-Hoff used in XCS for each fuzzy rule C_j. (Since we only explore single-step problems, no form of temporal difference learning is required here.)

However, an important difference is considered in Fuzzy-XCS: the credit distribution among the rules must be made proportionally to the degree of contribution of each rule to the obtained output. The reinforcement distribution in Fuzzy-XCS acts on the action set [A] when an exploration step is performed. The following subsections detail the reinforcement distribution process and the adjustment of the parameters.

1) Credit Distribution: The credit distribution among the rules of the action set [A] is made by analyzing the contribution of each rule to generate the aggregated output. To do so, first the difference between the output of the individual rule and the aggregated one is computed

$$e_j = \frac{1}{m}\sum_{i=1}^{m}\frac{|b_{ij} - \widehat{b}_i|}{max_i - min_i}, \tag{1}$$

with m being the number of output variables, min_i and max_i the extremes of the universe of discourse of the ith output variable, b_{ij} the mean of the core of the fuzzy set corresponding to the ith output variable of the jth fuzzy rule, and $\widehat{b}_i$ the defuzzified aggregated output of the whole action set.

Then, a weight w_j is assigned to each fuzzy rule of the action set as follows:

$$w_j = \frac{1 - e_j}{\sum_{k=1}^{|[A]|}(1 - e_k)}. \tag{2}$$

In some cases, this distribution could be equivalent to do it proportionally to the matching degree of each fuzzy rule divided by the sum of the matching degrees, but with the proposed process we avoid some inference bias that could appear with this latter approach.

2) Parameter Updates: To adjust the parameters of each rule, firstly the P (payoff) value is computed. In the case of single-step tasks, $P = r$, with r being the external reward.

Then, the following adjustment process is performed for each fuzzy rule belonging to the action set:

1) Increase its experience according to its contribution degree in this action set, $\exp_j \leftarrow \exp_j + w_j$.
2) Adjust the error values ϵ_j using the standard Widrow–Hoff delta rule with learning rate parameter β $(0 < \beta \leq 1)$ toward $|P - p_j|$ considering the weights w_j computed in (2) to distribute the adjustment, i.e.

$$\epsilon_j \leftarrow \epsilon_j + \beta \cdot w_j \cdot (|P - p_j| - \epsilon_j). \tag{3}$$

The MAM (*moyenne adaptive modifee*) technique is used by adjusting ϵ_j to the weighted average of the previously computed $|P - p_j|$ values for this rule, instead of the above equation, when the corresponding rule has not been sufficiently adjusted, i.e., if $\exp_j < 1/\beta$. The previous weights (w_j) computed for this rule are stored and used for calculating the weighted average.

3) Then, adjust prediction values

$$p_j \leftarrow p_j + \beta \cdot w_j \cdot (P - p_j). \tag{4}$$

Again, weights are considered to distribute the reinforcement. MAM technique is also used here as described above during the first adjustments. Therefore, weighted average of the previous P values is computed.

4) Then, recalculate the fitness values F_j from the updated values of ϵ_j, i.e.

$$F_j \leftarrow F_j + \beta \cdot \left(k'_j - F_j\right) \tag{5}$$

with

$$k'_j = \frac{k_j}{\sum_{R_i \in [A]} k_i}, \quad k_j = \begin{cases} 1, & \text{if } \epsilon_j < \epsilon_0 \\ \alpha(\epsilon_j/\epsilon_0)^{-\nu}, & \text{otherwise} \end{cases} \tag{6}$$

where ϵ_0 is a constant controlling the tolerance for prediction error, α $(0 < \alpha < 1)$ a constant controlling the tolerance for prediction error, and ν $(0 < \nu)$ another constant controlling the rate of decline in accuracy when ϵ_0 is exceeded.

No weights w_j are considered to update the fitness since it depends on the prediction error instead of the received payoff. Due to the same reason, MAM technique is not used.

5) Finally, the mean size of the action sets where the rule has been involved (as_j) is adjusted as follows:

$$\text{as}_j \leftarrow as_j + \beta \cdot (|[AS]| - \text{as}_j). \tag{7}$$

Neither weights nor MAM technique are used for this parameter.

D. Discovery Component

The EA for Fuzzy-XCS acts only on the action set [A] when an exploration step is performed. Note that this fact involves

niche search since only fuzzy rules fired together are evolved. In order to apply an EA, the average time period since the last EA application in the action set must be greater than the threshold θ_{EA}. When applied, it selects two rules by roulette-wheel selection based on fitness, applies crossover and mutation operators with probabilities χ and μ_{chrom} (per chromosome), respectively, and inserts the offspring in the population. If the population contains the maximum number of fuzzy rules allowed, two individuals are deleted to make room. They are randomly selected proportionally to the prediction error weighted by the mean action set sizes where each fuzzy rule was involved as follows:

$$\mathrm{prob}_i = \begin{cases} \mathrm{as}_i, & \mathrm{exp}_i < \theta_{\mathrm{exp}} \\ \left(\epsilon_i / \sum_j \epsilon_j\right) \cdot \mathrm{as}_i, & \text{otherwise} \end{cases}. \tag{8}$$

The threshold θ_{exp} ensures that the error estimation is reliable since the rule has been sufficiently updated. The reason of considering as_i is mainly to induce niches in the different state spaces.

A simple two-point crossover operator that only acts on the antecedent part of the chromosomes (binary coding scheme) is considered. In case that all the genes of a variable take the allele '0' after applying crossover, a gene set to '1' in the parents is randomly selected and it is set to '1' in the son. Prediction, prediction error, and fitness values of offspring are initialized to the mean values of the parents.

The mutation randomly selects an input/output variable of the rule. If an input variable is selected, one of the three following possibilities is applied: *expansion*, which flips to '1' a gene of the selected variable; *contraction*, which flips to '0' a gene of the selected variable; or *shift*, which flips to '0' a gene of the variable and flips to '1' the gene immediately before or after it. The selection of one of these mechanisms is made randomly among the available choices (e.g., contraction can not be applied if only a gene of the selected variable has the allele '1'). If an output variable is selected, the mutation operator simply increases or decreases the integer value. Prediction, prediction error, and fitness values of mutated rules are set to the mean values of the population.

An EA subsumption is performed. Thus, if the offspring is logically contained by either of its parents and this parent is sufficiently adjusted (its experience is $\mathrm{exp}_i \geq \theta_{\mathrm{exp}}$), the offspring is not added to the pool but the parent's numerosity (i.e., the number of copies of the rule) is incremented.

When no fuzzy rules cover the state with the highest matching degree, a covering mechanism is used to include a fuzzy rule with the input linguistic term set that best matches the state and a random action. Its parameters are set as follows: $\mathrm{exp} = 0$, $\epsilon = 0$, $p = 10$, $F = 0.01$, and $as = 1$. This covering mechanism will mainly act during the early iterations (when the population starts empty) or when a new state–space is explored.

IV. Experimental Results

Some experiments have been performed to test the behavior of Fuzzy-XCS. We have developed three different experiments. A first laboratory problem to test the effectiveness of the method, some function approximation problems, and a more complex real robot simulated problem. The former problem is only included to analyze if the algorithm is able to find, not only rules with a high payoff (low approximation error) but also as general as possible to describe the relation between state and action. Although fuzzy model with a low-approximation error can be successfully obtained in this problem by supervised learning, to do moreover the fuzzy model as compact as possible (maximal generality) is by far more difficult. The robot problem allows us to verify that the proposed algorithm can be applied to complex reinforcement problems with real-valued states and actions. The following subsections show the obtained results.

TABLE I
Rule Base Used to Generate the Data Set

	X_1					X_2					Y				
	VS	S	M	L	VL	VS	S	M	L	VL	VS	S	M	L	VL
R_1	x	x				x	x				x				
R_2			x	x	x	x	x					x			
R_3	x							x	x	x			x		
R_4		x						x	x	x				x	
R_5			x	x	x			x	x	x					x

A. A Laboratory Problem With Generalization

1) Problem Specification: We have developed a laboratory problem for function approximation where the optimal fuzzy rule set is known. This allows us to analyze if Fuzzy-XCS is able to obtain it. Thus, we have generated an example data set from a previously defined fuzzy rule base with different degrees of generalization. Two input variables and one output variable are considered. A total of 576 examples uniformly distributed in the input space (24×24) were generated. Five linguistic terms are considered for each variable. Uniformly distributed triangular-shaped membership functions are used. The same inference engine used by our algorithm is considered to generate the data set, i.e., FATI approach with *minimum* as conjunction and implication, *maximum* as disjunction and aggregation, and *center-of-gravity* as defuzzification. The fuzzy rule base considered to generate the data set is shown in Table I. The surface of the function to be approximated that results of this data set construction is depicted in Fig. 2.

The reward depends inversely on the difference between the inferred and the desired output in a nonlinear way, i.e.

$$r = \frac{1 - |y - \widehat{y}|}{1 + \frac{|y - \widehat{y}|}{\delta}} \tag{9}$$

with $\delta = 0.3$, y being the output obtained by the fuzzy rule-based system, and $\widehat{y}$ the expected output according to the data set for the current state. The objective is to obtain the set of rules that best approximate the data with the highest degree of generalization, i.e., a rule base as accurate and compact as possible.

2) Obtained Results: The used parameter values are the following: maximum number of rules $= 200$, $\beta = 0.2$, $\alpha = 0.1$, $\nu = 3$, $\epsilon_0 = 0.05$, $\theta_{\mathrm{exp}} = \theta_{\mathrm{EA}} = 30$, $\chi = 0.7$, and $\mu_{\mathrm{chrom}} = 0.1$. Note that standard parameter values are used. In order to properly check the covering mechanism behavior, the population is left empty at the beginning of the algorithm.

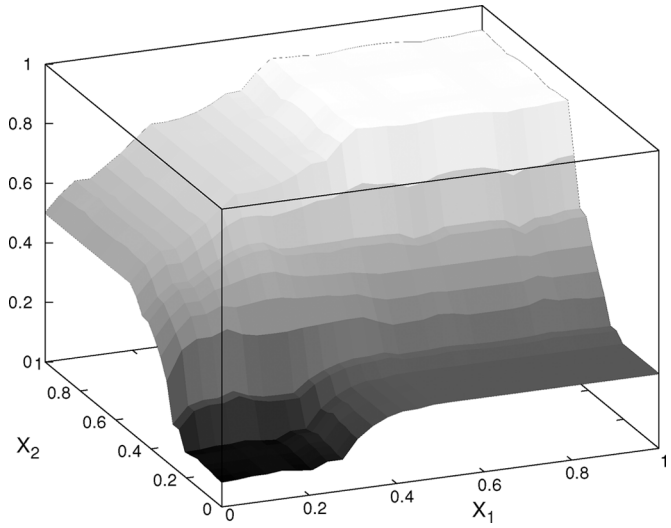

Fig. 2. Surface generated by the fuzzy rule set of Table I.

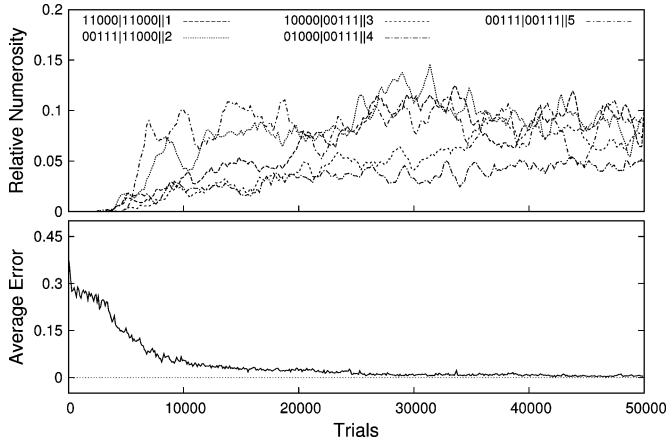

Fig. 3. Results of Fuzzy-XCS in the laboratory problem.

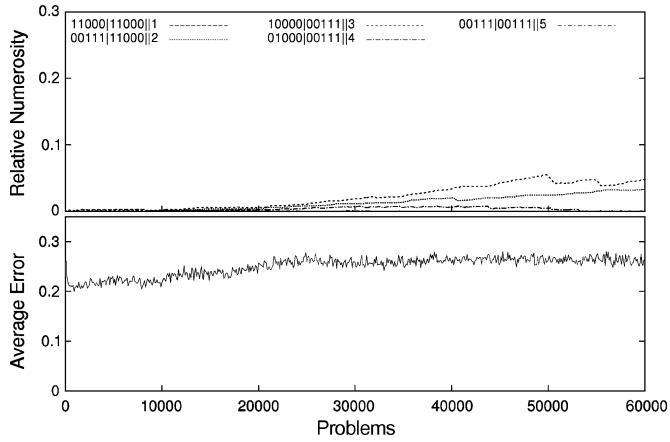

Fig. 4. Results of the Valenzuela-Rendón's algorithm [8].

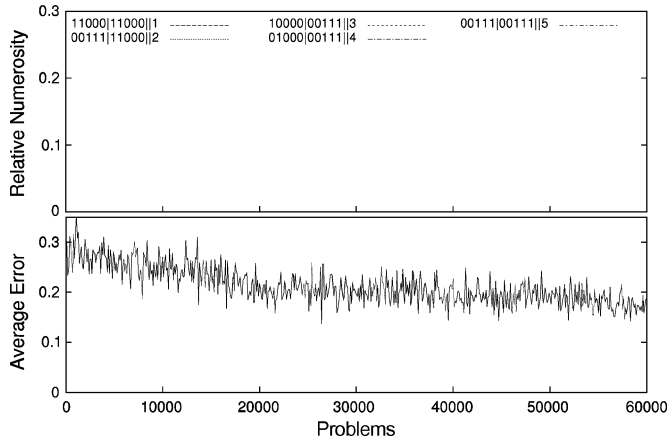

Fig. 5. Results of the Bonarini's algorithm [7].

Fig. 3 shows the average behavior of 10 runs of Fuzzy-XCS during 50 000 trials (of which 25 000 are explore trials and 25 000 are exploit ones). The upper figure depicts the relative numerosity (number of copies of the rule divided by the population size) of the five optimal fuzzy rules. It shows the capability of the algorithm to find and keep the optimal solution with maximal generality. The bottom figure depicts the mean approximation error ($|y - \hat{y}|$) of the last 50 exploit steps. It shows the capability of the algorithm to provide the appropriate action (output) to the corresponding state (input). Note that, according to this figure, the algorithm shows a very good performance in terms of generality and approximation error.

3) Comparison With Other Michigan-Style Algorithms: We have also analyzed the result obtained by two well-known Michigan-style genetic fuzzy systems, those proposed by Valenzuela-Rendón [8] and by Bonarini [7]. In this latter case, we have adapted the original proposal (designed for don't-care fuzzy rules) to DNF-type fuzzy rules. Figs. 4 and 5 shows the average results of 10 runs obtained by these two methods. It is important to remark that we have allowed these algorithms to perform 60 000 explore trials (instead 25 000 as our algorithm does) because they do not follow an exploration/exploitation scheme.

We can observe from these results that the Valenzuela-Rendon's algorithm is not able to reduce the error, i.e., to maximize the received payoff. This behavior would be related with the fact that this algorithm does not consider any selection among the matched rules to infer the action and therefore, rules with the same antecedent but different consequent will act together. Since the credit distribution is only based on the matching degree, the algorithm is not able to properly discriminate between good and bad actions for a specific state. The algorithm also presents a serious problem of overgeneralization since rules that match more frequently will accumulate more payoff and no mechanisms to avoid that is considered. Furthermore, from the upper part of Fig. 4 we can observe that the Valenzuela-Rendon's algorithm has also difficulties to find and keep rules with optimal generalization.

As regards the Bonarini's algorithm, we can observe that it obtains lower approximation errors than the Valenzuela-Rendon's one. However, its performance is still far from the one developed by our proposal. Its is curious to verify that the response of the method in each trial during the reinforcement learning process is very variant. It may be related with the fact that the algorithm chooses a rule for each subpopulation at

TABLE II
RESULTS OBTAINED BY THE ANALYZED METHODS IN THE FUNCTION APPROXIMATION PROBLEM

	Fuzzy-XCS	Valenzuela-Rendón	Bonarini	Pittsburgh-style GFS
R_1	0.7	0.0	0.0	0.1
R_2	0.9	0.0	0.0	0.2
R_3	0.8	0.0	0.0	0.0
R_4	0.7	0.0	0.0	0.2
R_5	1.0	0.0	0.0	0.1
no. of suboptimal rules	0.1	0.0	4.8	7.6
no. of erroneous rules	1.3	1.0	20.2	2.0
MSE	0.001614	0.144631	0.052516	0.001892
no. of analyzed examples	25,000	60,000	60,000	4,212,864

random for each trial. (In [11], the authors try to address it by keeping the same rules during the whole episode in multistep tasks with delayed reward.) With regard to the five optimal rules, the algorithm was not able to find anyone of them during the system learning in any of the 10 runs. Nevertheless, it is important to keep in mind that these results refers to our adaptation of the original don't-care Bonarini's proposal to DNF-type fuzzy rules.

4) Comparison With a Pittsburgh-Style Algorithm: We were also curious as to compare the performance of our online (reinforcement) learning process by a Michigan-style approach versus an off-line (supervised) learning using a simple Pittsburgh-style genetic fuzzy system. The Appendix includes a brief description of this algorithm.

In order to do this comparison, in Michigan-style algorithms it is necessary to previously filter the population because of it was used during the process for learning and acting, so there would be some bad rules that were included to learn. These rules use to have a low experience degree since bad rules should not survive for a long time. We have applied the following process to obtain a final fuzzy rule set in our Fuzzy-XCS and in the Valenzuela-Rendons's algorihtm (the original proposal does not mention anything about that). Firstly, such rules that have not been sufficiently tested (i.e., with experience lower than 30, $\exp_j < \theta_{\exp}$) are removed. Next, a iterative process is preformed until the population is empty. In each iteration, the rule with the highest prediction value of the remaining population is included in the final fuzzy rule set and removed from the population. All the rules contained in such a rule (those with subsumed antecedent and same consequent) are removed from the population. In the Bonarini's algorithm, the filter method originally proposed by him is preserved, which involves selecting the best fuzzy rule (according to the accumulated reward, i.e., strength) of each subpopulation. In the Pittsburgh-style genetic fuzzy system, the returned fuzzy rule set is the one with the best fitness in the last population that, due to the considered elitism, coincides with the best solution found during all the process.

Table II summarizes the results obtained by the analyzed methods. It shows the mean number of times (over the 10 runs performed for each algorithm) that each of the five optimal rules (according to Table I) appears in the returned fuzzy rule set. Therefore, a value of 1.0 means that the corresponding optimal rule has been obtained in all the runs. The table also shows the mean number of other suboptimal rules (i.e., those whose antecedent is subsumed by and the consequent is equal to an optimal rule) and erroneous rules included in the returned fuzzy rule set. We have also included the averaged mean square error (MSE) [see (19)] values of the fuzzy rule sets returned by the analyzed algorithms.

From that table, we can see that our Fuzzy-XCS algorithm find the five optimal rules frequently (always excepting three runs for R_1 and R_4, two runs for R_3, and one run for R_2). Moreover, the algorithm only returns one suboptimal rule (i.e., precise but not as general as possible) in the ten runs.

With respect to the two compared algorithms, they both are not able to find the optimal rules. In the case of the Valenzuela-Rendon's algorithm, it is due to the fact it has a clear tendency to overgeneralization, so the algorithm returns a fuzzy rule that outshines the rest of them. (The results without filtering the final population were also very poor: $R_1 = 0$, $R_2 = 6.6$, $R_3 = 9.7$, $R_4 = 0$, $R_5 = 0$, suboptimal $= 1.3$, erroneous $= 182.4$, MSE $= 0.093731$.) The Bonarini's algorithm at least returns some suboptimal rules that make it obtain a MSE better than Valenzuela-Rendon's, although it is worse than the one obtained by our proposal.

Finally, regarding the Pittsburgh-style genetic fuzzy system, it has serious difficulties to find optimal solutions, although it is able to obtain a low MSE (almost as good as the one obtained by our algorithm) by generating suboptimal rules. That behavior was unexpected since the problem seems to be better addressed by a supervised learning algorithm. However, we realized that when the objective is not only finding a DNF-type fuzzy rule set with low error approximation but also as compact as possible, the problem becomes more difficult since this kind of fuzzy rule has a high risk of inconsistencies and redundancies when the whole rule set is coded in an individual. Of course, better results can probably be obtained with a more sophisticated Pittsburgh-style algorithm, but this experiment shows at least that the design of this kind of algorithm is not trivial.

Furthermore, the knowledge resource (respecting the number of analyzed examples) used by the Pittsburgh-style approach to obtain these results is tremendously higher than the Michigan-style ones. In this respect, it is interesting to highlight that the proposed Fuzzy-XCS finds a very good solution after analyzing only 12 500 examples (as shown in Fig. 3), which is equivalent to only 22 times the data set size, i.e, only a 44% of the number of examples needed to evaluate the initial population (with 50 individuals) of the Pittsburgh-style genetic fuzzy system.

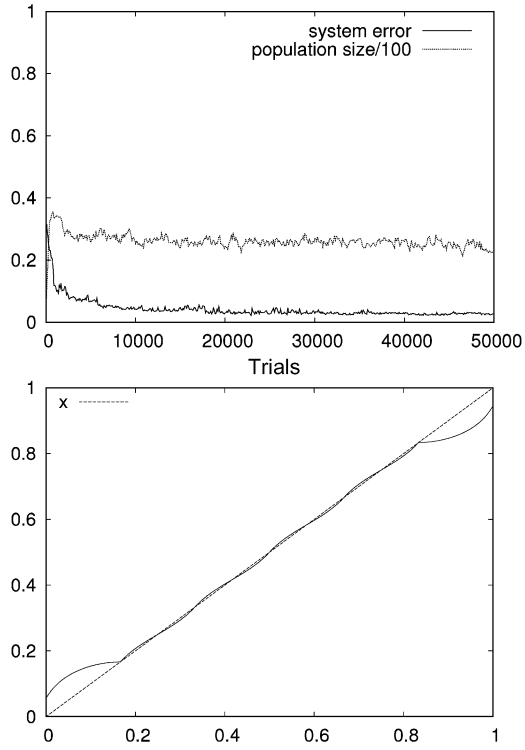

Fig. 6. Results of Fuzzy-XCS approximating the Valenzuela-Rendón's line function.

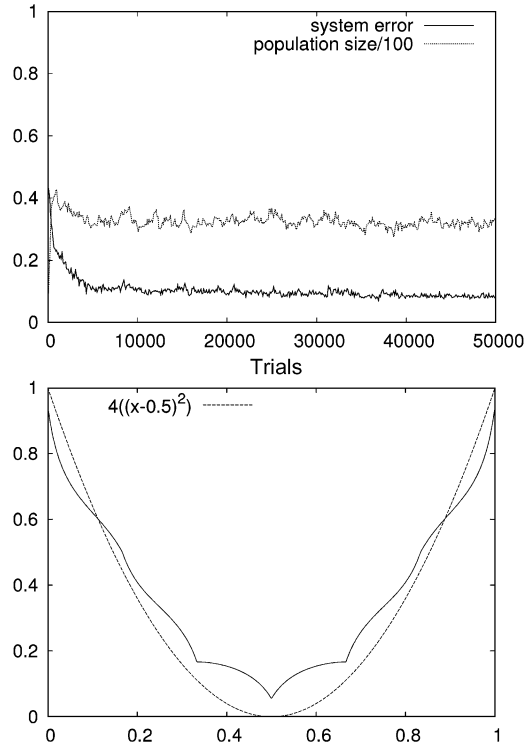

Fig. 7. Results of Fuzzy-XCS approximating the Valenzuela-Rendón's parabola function.

B. Function Approximation Problems

In that section, we experiment with the proposed Fuzzy-XCS algorithm when solving some function approximation problems. In all these experiments, the same parameter values than the ones used in the previous section are considered with the exception of the maximum number of rules, that will be 100, and the tolerance for prediction error, that will be $\epsilon_0 = 0.01$. Seven linguistic terms are used for both input and output variables. Again, ten runs are performed and average results reported.

We have considered the four following functions:

- Valenzuela-Rendón's line function [3], [8]

$$f(x) = x, \quad x \in [0, 1]. \tag{10}$$

- Valenzuela-Rendón's parabola function [3], [8]

$$f(x) = 4(x - 0.5)^2, \quad x \in [0, 1]. \tag{11}$$

- Wilson's parabola function [32]

$$f(x) = x^2, \quad x \in [0, 100]. \tag{12}$$

- Wilson's sinusoidal function [32]

$$f(x) = 100 sin\left(2\pi(x/100)\right), \quad x \in [0, 100]. \tag{13}$$

In the experiments, a dataset is generated for each function by sampling 1000 random values. At each experiment trial, a pair (x, y) is picked up at random, the x value is presented as state to the Fuzzy-XCS algorithm, and the corresponding y value used to calculate the reward [see (9)]. It is repeated 50 000 times in a exploration/explotation regime. Once the experiment has finished, the whole dataset is tested again as exploit trials (therefore, the population and the rule parameters are keep invariable) and the resulting system predictions are recorded.

Figs. 6, 7, 8, and 9 show the results obtained for each function. In each case, the upper figure shows the averaged system errors and population size during the learning process, while the bottom figure plots the system prediction values obtained at the mentioned last exploit process.

As it can be observed, the algorithm shows a good behavior approximating these functions. The error obtained in each function are related to the limits of the fuzzy inference capability since a relatively simple system is considered where only seven linguistic terms are used in each variable. More accurate results could be obtained by increasing the granularity of each variable (in fact, fuzzy systems are universal approximators [33]), but the legibility of the system would be lost and the use of fuzzy logic would not make sense. In that respect, the fuzzy approach (at least with a grid-partitioning of the input space as in Fuzzy-XCS) can not compete in prediction error terms (i.e., function approximation degree) with a real-valued learning classifier system (e.g., XCSF [32]). On the contrary, Fuzzy-XCS generates rules with a higher description capability and compactness, and needs lower population sizes than XCSF. Because of that, Fuzzy-XCS also seems to scale up better than XCSF to high-dimensional problems.

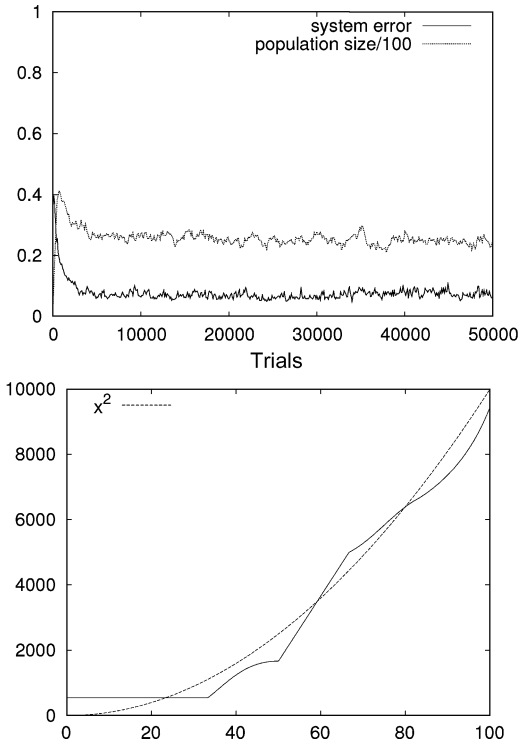

Fig. 8. Results of Fuzzy-XCS approximating the Wilson's parabola function.

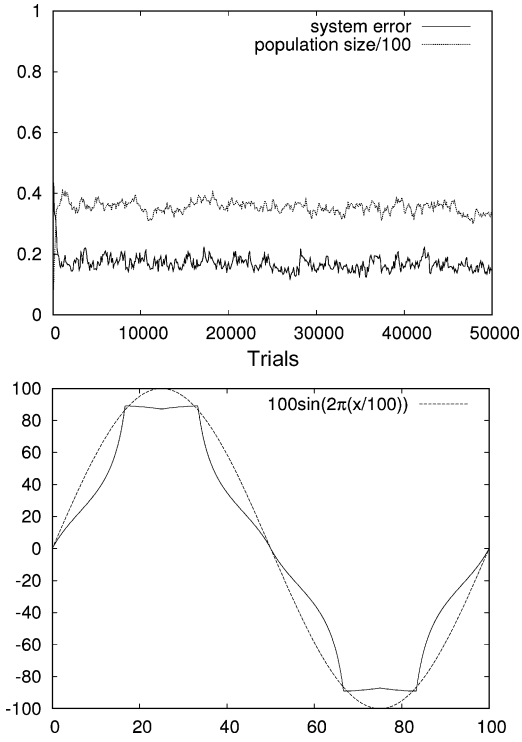

Fig. 9. Results of Fuzzy-XCS approximating the Wilson's sinusoidal function.

C. Realistic Mobile Robot Problem

This latter problem is by far more difficult to solve than the previous ones since it is a real online learning process. The problem involves giving a robot the capability to learn a set of fuzzy rules to implement the *wall-following behavior*. This behavior is usually implemented when the robot is exploring an unknown area, or when it is moving between two points in a map, generally in indoor environments. A good wall-following controller is characterized by three features: to maintain a suitable distance from the wall that is being followed, to move at a high velocity whenever possible, and finally to avoid sharp movements, making smooth and progressive turns and changes in velocity.

The behavior is not specially difficult and has been successfully addressed by some supervised-learning-based methods (e.g., [34]). What really makes this problem difficult is to perform online learning. That is, when the controller is designed at the same time the robot interacts with the environment. To do that, we will consider a reward that the robot will receive every time it performs an action. The reward is calculated from the sensorial information, thus making the problem more realistic. However, it makes the problem more difficult since it adds uncertainty, noise, and sensor reading errors. Of course, the problem is faced with real-valued states and actions.

Because of the online character of the problem, the sequence of states can not be previously defined since it depends on the actions performed during the process. This means that the robot will not be able to explore the whole input space in a reasonable time, even never if the actions do not allow it. This makes more difficult to generate a complete map and provokes a high risk of getting stuck in loops. Moreover, the robot is not reset on an original position when a number of steps is reached but it is continuously interacting with the environment. An additional difficulty is the fact that applying the same action in the same state can drive the robot to different new states. It involves the robot could receive different values of reward for the same state/action combination. Finally, the best reward that can be obtained in a specific state may be significantly lower than the maximum if any possible action drive the robot to an optimal situation (exact right distance, maximum velocity, and parallel orientation) in the next control cycle.

1) Input and Output Variables: We have employed the Nomad 200 software to simulate the behavior of the robot. This robot is provided by a ring of 16 ultrasound sensors. The four considered input variables are calculated from this information as follows [34]. Two of the input variables are the relative right-hand distance (RD) and the distance quotient (DQ), which are calculated as

$$\text{RD} = \frac{\text{right-hand distance}}{d_{\text{wall}}} \tag{14}$$

$$\text{DQ} = \frac{\text{left-hand distance}}{\text{right-hand distance}}. \tag{15}$$

DQ shows the relative position of the robot inside a corridor, which provides with information that is more relevant to the problem than simply using the left-hand distance. A high value for DQ means that the robot is closer to the right-hand wall, while a low-value indicates that the closer wall is the left-hand one. In this case the robot should approach to the right-hand wall, although the right-hand wall is closer than the reference

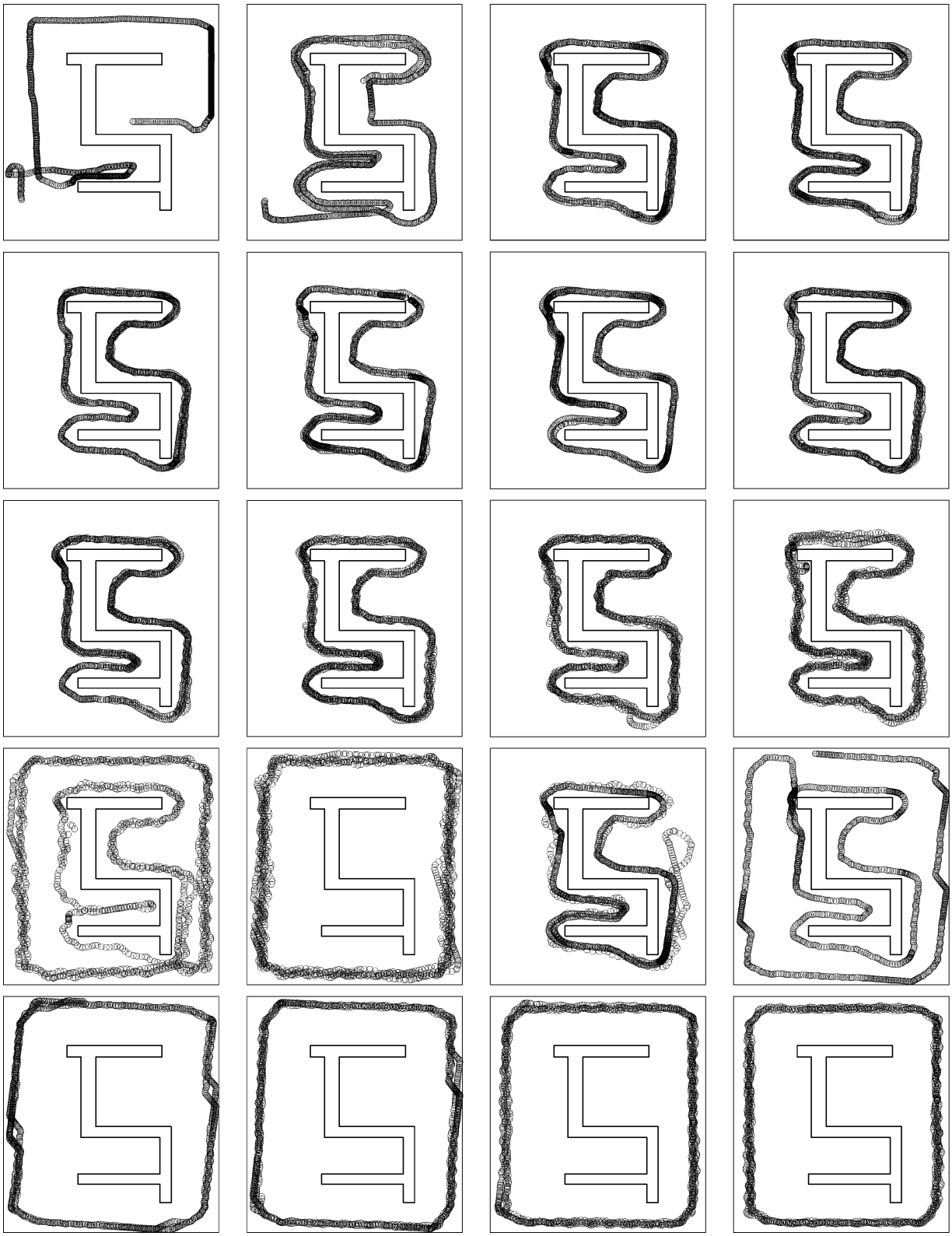

Fig. 10. Path sequence (from top to bottom, from left to right) followed by the robot during the online learning performed by the Fuzzy-XCS algorithm.

distance. The other input variables are the relative linear velocity of the robot (RLV)

$$\text{RLV} = \frac{v_r}{v_{\max}} \tag{16}$$

where v_r is the real linear velocity of the robot; and the orientation of the robot with respect to the wall it is following (θ_{wall}). A positive value of the orientation indicates that the robot is approaching to the wall, whilst a negative value means the robot is

moving away from the wall. The output variables are the linear velocity (LV) and the angular velocity (AV).

All the information used to calculate distances and orientations is obtained from the ultrasound sensors of the robot. Data are processed using the *distributed perception* [35], and for this reason the sensors are grouped in different sets. Distances are measured as the minimum distance of a set of sensors (obviously, the set of sensors is different for RD and DQ). θ_{wall} will be a weighted sum of the orientation of each sensor in the set, giving more weight to those sensors that detect closer obstacles:

$$\theta_{\text{wall}} = \frac{\sum_{i=1}^{n_u} angle_i \cdot \left(1 - \frac{d_i}{\max_{ud}}\right)}{\sum_{i=1}^{n_u} \left(1 - \frac{d_i}{\max_{ud}}\right)} \tag{17}$$

where n_u is the number of sensors in the set, angle$_i$ is the angle of sensor i, d_i the measured distance of this sensor, and $\max_{ud}$ is the maximum distance an ultrasound sensor can measure.

The universe of discourse is, for some variables (RD, DQ, and θ_{wall}), a reduced version of the real universe of discourse that contains those values of the variable that are meaningful for learning. For example, high values of distance are not useful during learning, because for all of them the robot will execute the same action. Therefore, it is enough to include only a few high values in the universe of discourse. The following universes of discourse are considered: RD $\in [0, 4]$, DQ $\in [0, 3]$, $\theta_{\text{wall}} \in [-45, 45]$ degrees, RLV $\in [0, 1]$, LV $\in [0, 51]$ cm/s, and AV $\in [-30, 30]$ degrees/s.

2) Reward: The reward is computed as follows:

$$R(\text{RD},\text{LV},\theta_{\text{wall}}) = 1 - \left(\alpha_1 \frac{|\text{RD}-1|}{3} + \alpha_2 |\text{LV}-1| + \alpha_3 \frac{\theta_{\text{wall}}}{45}\right). \tag{18}$$

The weights α_1, α_2, and α_3 (with $\alpha_1 + \alpha_2 + \alpha_3 = 1$) are set as follows: $\alpha_1 = 0.8$, $\alpha_2 = 0.15$, and $\alpha_3 = 0.05$. These weights indicate how much important the deviation in the value of a variable is with respect to the deviation of other variables. The highest weight has been assigned to the distance, as small variations of RD with respect to the reference distance should be highly penalized. An intermediate weight is associated to velocity and, finally, the least important contribution is for the orientation of the robot.

3) "Inborn" Knowledge: For a better behavior of the robot, a simple knowledge has been previously provided. We could say that it will be its inborn knowledge. When the reading of the ultrasound sensors get a very low value (25.4 cm), i.e., the robot is very near to an obstacle, its velocity is automatically decreased in order to avoid the impact, or to reduce its damage if it is done. Furthermore, if the robot is physically in contact with an obstacle (it is detected from the bumper senors), it automatically turns to the nearest free angle (detected by the ultrasound sensors). This is the only knowledge we provide to the robot before starting the learning.

4) Obtained Results: In this experiment, the maximum number of rules is set to 1000. The rest of parameter values are equal to the ones used in the previous problems, with $\epsilon_0 = 0.05$. The population is randomly initialized at the beginning of the algorithm.

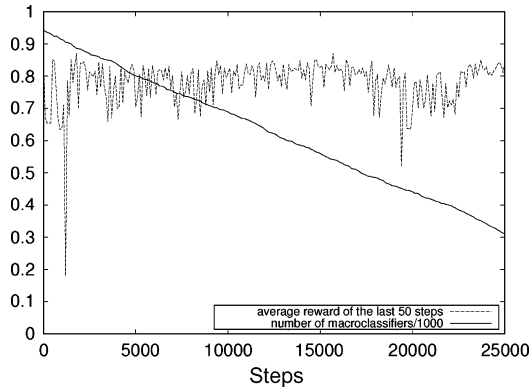

Fig. 11. Results of Fuzzy-XCS in the robot problem.

We have applied the algorithm in a specific wall configuration. It includes different situations that the robot usually faces during navigation: straight walls of different lengths, followed and/or preceded by a number of concave and convex corners, etc. Fig. 10 depicts the path followed by the robot during the online learning process for 25 000 control cycles. It has been divided in different pictures (each one containing 1,250 control steps) to properly represent the temporal sequence. The set of figures must be read from top to bottom and from left to right. Therefore, the first row represents the first 5000 control cycles, the second row the control cycles between 5001 and 10 000, and so on. The robot trajectory is represented by circular marks. A higher concentration of marks indicates lower velocity.

According to this figure, we can observe how the robot is able to face the problem implementing a right-hand wall-following behavior. Note also the real-valued action performed by it. Due to the exploration feature of some actions inferred by the algorithm, the robot is sometimes guided to a bad behavior that, nevertheless, is able to solve after several iterations.

Furthermore, Fig. 11 shows the obtained results of performance and numerosity by the Fuzzy-XCS algorithm for 25 000 steps. The solid line represents the number of different fuzzy rules of the population divided by the population size. The dashed line represents the average reward received by the robot in the last 50 steps.

As we can observe, the robot gets a proper reward degree. It is important to remark that in this problem we do not know the maximum attainable reward from a specific state (usually less than 1) and that different parts of the environment (e.g., corners) are more difficult to be solved. It makes the plot of the average reward to be far from 1. For example, the low reward degree obtained around trial 19 500 corresponds to the change of the robot between following the central wall to follow the perimeter wall (see row 4, column 4, in Fig. 10). With respect to the number of different fuzzy rules contained in the population, it is observed a gradual decreasing that shows the convergence of the algorithm by reducing the number of fuzzy rules improving their generalization capability (compactness) without lack of performance. Although the robot is far from behaving optimally, the results obtained in this difficult problem show promising possibilities of the proposed algorithm.

V. Concluding Remarks

This paper has presented a proposal to properly develop an accuracy-based Michigan-style fuzzy rule-based system for continuous state and action. Its main advantages are, compared with the most of genetic fuzzy systems, its capability to perform online learning and, compared with other Michigan-style genetic fuzzy systems, its capability to obtain maximal generalization, i.e., representation of the fuzzy rule set as compact as possible.

The algorithm design is inspired with the well-know XCS algorithm for non-fuzzy rules. The fitness function is based on the estimation of the performance prediction error in order to look for robust (in the sense of the received reward) fuzzy rules. Niche search is also considered.

Promising results of the proposal have been obtained in some function approximation problems and a realistic robot simulation online learning. Future work involves investigating the behavior of the proposal in other reinforcement problems with continuous actions and multi-step tasks with immediate reward.

Appendix
Pittsburgh-Style Genetic Fuzzy System Used in the Experiments

The Pittsburgh-style genetic fuzzy system used for comparison in the experiments of Section IV-A consists of the following components. A *generational approach* with direct replacement (offspring replaces the corresponding parents) is considered. An *elitism* component that ensures the best chromosome survival of the previous generation is applied. The *fitness* is the MSE

$$f(S) = \frac{1}{N}\sum_{i=1}^{N}\left(S(x^i) - y^i\right)^2 \tag{19}$$

with S being the evaluated fuzzy system, N the data set size, and (x^i, y^i) the ith input-output pair of the data set. We have to say that we also tested with other version were the fitness is a tradeoff between MSE and the number of rules in order to penalize fuzzy rule set with low generalization. However this version tended to obtain overgeneral fuzzy rule sets with a lower performance and the results were discarded in this paper.

The same *coding scheme* than the proposed Fuzzy-XCS (Section III-A) for each fuzzy rule is used, but taking into account that now a chromosome is composed of a set of rules. *Variable-length chromosome size* is considered. The population is randomly initialized. *Binary tournament selection* is used.

The *crossover* operator randomly chooses a cross point between two fuzzy rules at each chromosome, and exchanges the right string of them. Therefore, the crossover only exchanges complete rules, but it does not create new ones since it respects rule boundaries on chromosomes representing the individual rule base. In the case that inconsistent rules appear after crossover, the ones whose antecedent is logically subsumed by the antecedent of a more general rule are removed. Redundant rules are also removed.

The *mutation* operator works in a similar way to the one proposed for our Fuzzy-XCS algorithm in Section III-D. In the same way, specific rules appeared after mutation are subsumed by the most general ones and redundant rules are removed. Note that, although the fitness is guided only by approximation error, the fact of subsuming rules in crossover and mutation provides the system with a generalization orientation.

The parameter values used in the experiments and shown in Table II were as follows: population size $= 50$, number of generations $= 200$, crossover probability $= 0.7$, and mutation probability (per chromosome) $= 0.1$. The same inference engine used in the rest of experiments of this paper (i.e., FATI approach, Max-Min scheme, and center-of-gravity as defuzzification) is considered.

References

[1] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: World Scientific, 2001.

[2] S. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[3] M. Valenzuela-Rendón, "The fuzzy classifier system: A classifier system for continuously varying variables," in *Proc. 4th Int. Conf. Genetic Algorithms*, San Mateo, CA, 1991, pp. 346–353.

[4] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *Proc. 5th Int. Conf. Genetic Algorithms*, San Mateo, CA, 1993, pp. 223–230.

[5] T. Furuhashi, K. Nakaoka, and Y. Uchikawa, "Suppression of excess fuzziness using multiple fuzzy classifier systems," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Piscataway, NJ, 1994, pp. 411–414.

[6] K. Nakaoka, T. Furuhashi, and T. Uchikawa, "A study on apportionment of credits of fuzzy classifier systemfor knowledge acquisition in large scale systems," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Piscataway, NJ, 1994, pp. 1797–1800.

[7] A. Bonarini, "Evolutionary learning of fuzzy rules: Competition and cooperation," in *Fuzzy Modelling: Paradigms and Practice*, W. Pedrycz, Ed. Norwell, MA: Kluwer Academic, 1996, pp. 265–284.

[8] M. Valenzuela-Rendón, "Reinforcement learning in the fuzzy classifier system," *Expert Systems With Applications*, vol. 14, pp. 237–247, 1998.

[9] J. Velasco, "Genetic-based on-line learning for fuzzy process control," *Int. J. Intell. Syst.*, vol. 13, pp. 891–903, 1998.

[10] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern.—Part B: Cybern.*, vol. 29, pp. 601–608, 1999.

[11] A. Bonarini and V. Trianni, "Learning fuzzy classifier systems for multi-agent coordination," *Inf. Sci.*, vol. 136, pp. 215–239, 2001.

[12] D. Gu and H. Hu, "Accuracy based fuzzy Q-learning for robot behaviours," in *Proc. 12th IEEE Int. Conf. Fuzzy Syst.*, 2004, pp. 1126–1131.

[13] P. Glorennec, "Fuzzy Q-learning and dynamical fuzzy Q-learning," in *Proc. 3rd IEEE Int. Conf. Fuzzy Syst.*, Orlando, FL, 1994, pp. 474–479.

[14] P. Glorennec and L. Jouffe, "Fuzzy Q-learning," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, 1997, pp. 659–662.

[15] J. Holland and J. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-Directed Inference Systems*, D. Waterman and F. Hayes-Roth, Eds. San Diego, CA: Academic, 1978, pp. 313–329.

[16] J. Holland, "Properties of the bucket brigade algorithm," in *Proc. First Int. Conf. Genetic Algorithms and Their Applications*, Hillsdale, NJ, 1985, pp. 1–7, Lawrence Erlbaum Associates.

[17] G. Roberts, "Dynamic planning for classifier systems," in *Proc. Fifth Int. Conf. Genetic Algorithms*, B. Forrest, Ed., 1989, pp. 244–255.

[18] R. Sutton, "Reinforcement Learning Architecture for Animats," in *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour*. Cambridge, MA: MIT Press, 1991, pp. 188–296.

[19] S. Wilson, "ZCS: A zeroth level classifier system," *Evol. Comput.*, vol. 2, no. 1, pp. 1–18, 1994.

[20] S. Smith, "A Learning System Based on Genetic Adaptive Algorithms," Ph.D. dissertation, University of Pittsburgh, Pittsburgh, PA, 1980.

[21] S. Wilson and D. Goldberg, "A critical review of classifier systems," in *Proc. Third Int. Conf. Genetic Algorithms*, D. Schaffer, Ed., 1989, pp. 244–255.

[22] A. Tomlinson and L. Bull, "Symbiogenesis in learning classifier systems," *Artif. Life*, vol. 7, no. 1, pp. 33–62, 2001.

[23] A. Tomlinson and L. Bull, "An accuracy-based corporate classifier system," *Soft Computing*, vol. 6, no. 3–4, pp. 200–215, 2002.

[24] J. Grefenstette, "Multilevel Credit Assignment in a Genetic Learning System," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Piscataway, NJ: Lawrence Erlbaum Associates, 1987, pp. 202–209.
[25] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann arbor, MI: The University of Michigan Press, 1975.
[26] L. Booker, "Triggered rule discovery in classifier systems," in *Proc. Third Int. Conf. Genetic Algorithms and Their Applications*, J. Schaffer, Ed., 1989, pp. 265–274, Morgan Kaufmann.
[27] L. Booker, "Improving the performance of genetic algorithms in classifier systems," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 80–92, Lawrence Erlbaum Associates.
[28] S. Wilson, "State of XCS classifier system research," in *Learning Classifier Systems: From Foundations to Applications*, P. Lanzi, W. Stolzmann, and S. Wilson, Eds. New York: Springer, 2000, pp. 63–81.
[29] L. Bull, Ed., *Applications of Learning Classifier Systems*. New York: Springer, 2004.
[30] L. Bull and T. Kovacs, Eds., *Foundations of Learning Classifier Systems*. New York: Springer, 2005.
[31] H. Ishibuchi, T. Yamamoto, and T. Murata, "Hybridization of fuzzy GBML approaches for pattern classification problems," *IEEE Trans. Syst., Man, and Cybern.—Part B: Cybern.*, vol. 35, no. 2, pp. 359–365, 2005.
[32] S. Wilson, "Classifiers that approximate functions," *Natural Computing*, vol. 1, no. 2–3, pp. 211–233, 2002.
[33] J. Castro, "Fuzzy logic controllers are universal approximators," *IEEE Trans. Syst., Man, and Cybern.*, vol. 25, no. 4, pp. 629–635, 1995.
[34] M. Mucientes and J. Casillas, "Quick design of fuzzy controllers with good interpretability in mobile robotics," *IEEE Trans. Fuzzy Syst.*, 2007, to be published.
[35] J. Urzelai, J. Uribe, and M. Ezkerra, "Fuzzy controller for wall-following with a non-holonomous mobile robot," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, 1997, pp. 1361–1368.

Jorge Casillas received the B.Sc., M.Sc., and Ph.D. graduate degrees in computer science from the University of Granada, Spain, in 1996, 1998, and 2001, respectively.

He is an Associate Professor with the Department of Computer Science and Artificial Intelligence, University of Granada, where he is a member of the Soft Computing and Intelligent Information Systems research group. He has worked in several research projects supported by the Spanish Government and the European Union. He has co-edited two books, co-edited two journal special issues, and organized four special sessions in international conferences on the topics "interpretability-accuracy tradeoff in fuzzy modeling," "genetic fuzzy systems," and "intelligent robotics." He serves on the Editorial Board of the Evolutionary Intelligence journal of Springer. He is coauthor of about 17 journal papers, eight book chapters, and 35 conference papers. His research interests include fuzzy modeling, intelligent robotics, knowledge discovery, and metaheuristics.

Brian Carse (M'86) was born in Durham, U.K., in 1960. He received the B.A. (Hons.) degree and the M.A. degree in natural sciences (physics with theoretical physics) from the University of Cambridge, U.K., in 1981 and 1984, respectively, and the Ph.D. degree in computer science from the University of the West of England in 1997.

He is Senior Lecturer in the Department of Computing, Engineering and Mathematical Sciences at the University of the West of England, and is a member of the recently formed Bristol Robotics Laboratory (formerly the Intelligent Autonomous Systems Laboratory). His main research interests include fuzzy systems, neural networks and evolutionary computing. He has been active in research on genetic fuzzy systems for over 13 years and has published over 80 international journal and conference papers, as well as several book chapters.

Dr. Carse is a member of the IET.

Larry Bull received the B.Sc. (Hons.) degree in computing for real-time systems in 1992 and the Ph.D. degree in 1995 from University of the West of England (UWE).

He is Professor of Artificial Intelligence within the Faculty of Computing, Engineering and Mathematical Sciences (CEMS) at UWE. His research interests include evolutionary computing, reinforcement learning, neural computing, and biology-inspired artificial systems in general.