Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier Systems

Andrea Bonarini
Politecnico di Milano AI and Robotics Project
Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza Leonardo da Vinci, 32 - 20133 Milano - Italy
E-mail: bonarini@elet.polimi.it - URL:http://www.elet.polimi.it/people/bonarini

Abstract

We have implemented a tool to compare different modules of *Reinforcement Learning* algorithms applied to *Learning Classifier Systems* (LCS). We focus on three main classes of modules: *credit assignment* modules, *exploration policies*, and *evolutionary strategies*. For each class we have implemented many of the proposals we can find in literature and also some new algorithms that we have designed. In this paper, we present the results of the application of our tool to both fuzzy and crisp LCSs that learn behaviors for simulated autonomous agents. Fuzzy LCSs can be considered a successful approach to cope with real-valued input and output in a real environment. A lot of investigations can be done with this tool in this experimental setting. This paper is focused on the comparison among different credit assignment algorithms and on their performance in learning both crisp and fuzzy models. Our experiments show that the more complex credit assignment algorithms (such as, for instance, the *TD(λ)* generally have better performances than the more basic (such as *Q-learning* or *Bucket Brigade*) also when applied to LCSs. Moreover, fuzzy LCSs seem to require a larger computational effort, but also show more robustness.

1 Introduction

We are involved in a large project aimed at founding a methodology to design and select Reinforcement Learning algorithms to face classes of problems. We have implemented a tool supporting this activity with special concern to crisp and fuzzy LCS. In section 2 we define the LCS structure we are considering in this paper, particularly suitable to real-world, mobile robot applications [Bonarini1996], [Bonarini1997], [Bonarini and Basso1997], [Bonarini1998]. We also point out the relevance of the fuzzy representation to approach learning relationships among real-valued variables. In section 3, we present some of the credit assignment algorithms we have implemented. We also discuss a general methodology to extend traditional reinforcement distribution formulae to learn fuzzy models. Section 4 is devoted to the presentation of some of the results we obtained when comparing different credit assignment algorithms applied both on crisp and fuzzy models.

2 Fuzzy and crisp LCSs

In this paper we consider a simplified type of LCSs [Bonarini1998]. Each classifier is a rule with antecedents and consequents. Each antecedent corresponds to a value for a *linguistic variable* associated to a real-valued variable. In *crisp LCS* we consider that each value is one of the n symbols representing intervals of values of the linguistic variable. In *fuzzy LCS* each value is a symbol representing a fuzzy subset [Klir *et al.*1997] of the set of the values the real-valued variable can take. A *don't care* symbol may replace any of the mentioned symbols, meaning that the value of the corresponding variable is not relevant, that is any real value matches the antecedent. The consequents are symbols corresponding to crisp values for each of the consequent variable. A classifier of this kind as a conceptual shape like this:

```
IF (FrontDistance IS Low)
   AND (LeftDistance is High)
   AND (RightDistance is Don_T_Care)
THEN (TurnDirection IS Left)
```

actually implemented in a more compact string such as "130:3".

We call the set of real values, one for each variable, in input to the robot a *real-valued state*. At each *control step*, one crisp classifier is selected among the ones matching the current real-valued state, and its consequents are sent to the actuators of the robot, making it acting in the environment.

We call the set of fuzzy sets, one for each variable, matching a real-valued state, a *fuzzy state*. When operating with fuzzy LCSs, the current state matches different fuzzy states, i.e., different fuzzy classifiers. We consider subpopulations of classifiers having the same antecedent (eventually including don't cares) and different consequents. At each control step, we select one classifier for each matching subpopulation (i.e., one for each fuzzy state), we combine the proposed outputs with a fuzzy aggregation operator [Klir *et al.*1997], and we send the real value thus obtained to the actuators.

2.1 Why Fuzzy LCSs?

Since their introduction in the late sixties fuzzy sets have been adopted to map real numbers to symbolic labels. Elements of an universe of discourse belong to a fuzzy set to a certain extent, according to the so-called *membership function* that defines the fuzzy set. Let us consider an universe of discourse X (say, an interval of real numbers), and a fuzzy set, associated to the label *close*, defined by a membership function $\mu_{close}(.)$ that ranges on elements of the universe of discourse and maps them to the real interval [0,1]. We can say that an element $\bar{x}$ of the universe of discourse (in our case a distance in mm) belongs to the fuzzy set *close* to the extent $\mu_{close}(\bar{x})$. For the fuzzy set *close* in figure 1, $\mu_{close}(690) = 0.7$. Giving a semantic value to the label associated to the fuzzy set, we can say that the value $\bar{x} = 690$ can be $classified$ as *close* with degree 0.7.

Fuzzy sets are often used to classify real values in categories, thus making it possible (or improving) symbolic reasoning about the phenomena that underlie the numbers. When we say that a number $\bar{x}$ belongs to a fuzzy set to a given extent, we add some information to it that comes from the definition of the fuzzy set. For instance, the value $\mu_{close}(\bar{x})$ can be considered as an alternative representation of $\bar{x}$, although it is really so only for some fuzzy set definitions $\mu_l(\cdot)$. In general, the relationship between a real value $\bar{x}$ and its degree of membership to a fuzzy set $\bar{l}$, $\mu_{\bar{l}}(\bar{x})$, is not bijective. This potential problem can be solved by defining partially overlapping fuzzy sets, and considering as representative of $\bar{x}$ the set of all the values

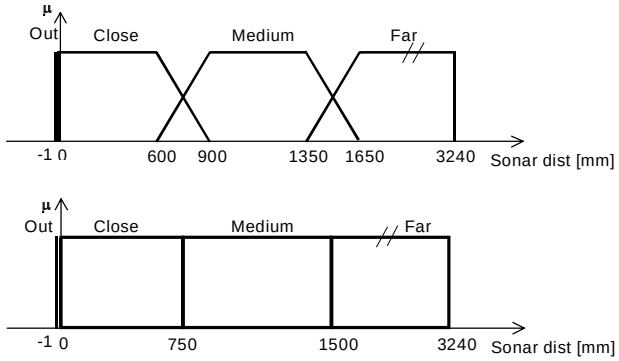

Figure 1: The fuzzy membership functions (above) and the intervals (below) to interpret the distance measured by a sonar sensor, used in the antecedents of, respectively, fuzzy and crisp classifiers presented in Section 4.

$\mu_l(\bar{x}), \forall\ l$.

Fuzzy rules represent relationships among fuzzy sets. Since the mapping between a number and its classification is crisp and well-defined, we can say that a fuzzy rule is a compact representation of a relationship between the real values that can be classified by the fuzzy sets in its antecedents and consequents. A set of fuzzy rules implement a mapping among real values that combines the mappings implemented by each rule. So, a fuzzy LCS is a way to consider different local models (fuzzy rules) to obtain a complex, in general non linear, mapping [Bonarini1996]. In particular, it is common to consider partially overlapping fuzzy sets, as the ones represented in figure 1. In this case, any set of real values for the input variables is matched to some extent by at least two rules.

To summarize: fuzzy sets can provide an alternative representation of real numbers, a fuzzy rule implements a model that maps real values to real values in given intervals, a set of fuzzy rules implement a complex mapping on the full range of the involved variables.

Interesting aspects of the fuzzy LCS mapping are: a compact, clear representation, global results coming from the combination of local models, smooth transition between close models. These aspects have some impact also on learning. A compact representation means that the model to be learnt is small, and so is the search space. The accent on local models implies the possibility to learn by focusing at each step on small parts of the search space only, thus reducing useless interaction among partial solutions.

The interaction among local models, due to the intersection of neighbor fuzzy sets guarantees that local learning reflects on global performance [Bonarini1996]. It is not so for crisp LCSs. Moreover, the smooth transition among different models implemented by fuzzy rules implies robustness with respect to data noise [Klir *et al.*1997]. These are the main reasons why fuzzy LCSs are an interesting approach to learn relationships among real values.

3 Credit assignment algorithms

In this paper we focus on some credit assignment algorithms that we have implemented as modules of our system to be applied to learn the rule-based models described in the last section. We firstly introduce the ones used with crisp models, then presenting a methodology to extend them to fuzzy models by considering their intrinsic nature. We associate to each classifier many parameters, among which we mention here only *strength*, and *accuracy*.

3.1 Algorithms for crisp classifiers

Here, we consider only the three algorithms, of the eleven we have implemented, which show the most different characteristics.

3.1.1 Crisp Q-Learning

At time t the system is in the real-valued state s_t, after having executed action a_{t-1} from a state s_{t-1} matched by the interval-valued antecedent $\hat{s}_{t-1}$, and receives the reinforcement r_t. The Q-value given to the pair $\langle \hat{s}_{t-1}, a_{t-1} \rangle$ is updated by:

$$\begin{aligned} Q_t(\hat{s}_{t-1}, a_{t-1}) &= Q_{t-1}(\hat{s}_{t-1}, a_{t-1}) + \\ &\quad \alpha\,(r_t + \gamma\, max_a(Q_t(\hat{s}_t, a)) \\ &\quad - Q_t(\hat{s}_{t-1}, a_{t-1}) \end{aligned}$$

with the learning rate $0 < \alpha < 1$ and the discount factor $0 < \gamma \leq 1$. The strength of the classifier c_{t-1} selected by the exploration policy at time t is $Q(\hat{s}_{t-1}, a_{t-1})$.

The accuracy of the classifier c_{t-1} is updated by the formula:

$$\begin{aligned} E_t(c_{t-1}) &= E_{t-1}(c_{t-1}) + \\ &\quad \omega\,(|r_t + \gamma\, max_a(Q_t(\hat{s}_{t-1}, a)) \\ &\quad - Q_t(\hat{s}_{t-1}, a_{t-1})| - E_{t-1}(c_{t-1})) \end{aligned}$$

with the learning rate $0 < \omega \leq 1$.

3.1.2 Crisp $TD(\lambda)$

At time t the reinforcement distribution algorithm performs these steps.

1. Receive the eventual reinforcement r_{t-1} due to the action a_{t-1} selected by the policy in state s_{t-1}.

2. Compute the estimation error of the evaluation function $V(\cdot)$ in s_{t-1}, equivalent to the immediate prediction error ΔE of the corresponding classifier c_{t-1} by:

$$\Delta E(c_{t-1}) = r_t + \gamma V_t(s_t) - V_{t-1}(s_{t-1})$$

with the discount factor $0 \leq \gamma \leq 1$.

3. For any state $\hat{s}$:

3.1. Update the *eligibility* $e(\hat{s})$ by the formula:

$$e_t(\hat{s}) = \begin{cases} 1 & if\, \hat{s} = \hat{s}_{t-1} \\ \gamma\lambda e_{t-1}(\hat{s}) & otherwise \end{cases}$$

with $0 \leq \lambda \leq 1$.

3.2. If the eligibility of the state $\hat{s}$ is greater than a threshold value ϵ $(0 \leq \epsilon \leq 1)$, update the value function $V(\cdot)$ for any $\hat{s}$ according to:

$$V_t(\hat{s}) = V_{t-1}(\hat{s}) \alpha \Delta E(c_{t-1}) e_t(\hat{s})$$

with $0 < \alpha < 1$.

4. Update the value of the policy for the pair $\langle \hat{s}_{t-1}, a_{t-1} \rangle$, that is the strength w of the classifier c_{t-1}, by the formula:

$$w_t(c_{t-1}) = w_{t-1}(c_{t-1}) + \beta\,(\Delta E(c_{t-1}))$$

with $\beta > 0$.

5. Update the accuracy of the classifier c_{t-1} by:

$$\begin{aligned} E_t(c_{t-1}) &= E_{t-1}(c_{t-1}) + \\ &\quad \omega\,(|\Delta E(c_{t-1})| - E_{t-1}(c_{t-1})) \end{aligned}$$

with $0 < \omega \leq 1$.

3.1.3 Crisp Bucket Brigade

At time t the reinforcement distribution algorithm follows these steps.

1. Receive the eventual reinforcement r_{t-1} due to the action a_{t-1} selected by the policy in state s_{t-1} and the new action a_t selected by the policy to be done in the current state s_t.

2. Update the strength of the classifier c_{t-1} belonging to the action set of the previous step A_{t-1}, by the formula:

$$w_t(c_{t-1}) = w_{t-1}(c_{t-1}) + r_t + \frac{1}{n} \sum_{c_{t-1} \in A_{t-1}} \beta w_t(c_t)$$

where the bid coefficient $0 < \beta < 1$, and n is the number of classifiers in A_{t-1}.

3. Update the accuracy of each classifier c_{t-1} in A_{t-1} by:

$$\begin{aligned} E_t(c_{t-1}) \quad = \quad & E_{t-1}(c_{t-1}) + \\ & \omega \left(|w_t(c_{t-1}) - w_{t-1}(c_{t-1})| - \right. \\ & \left. E_{t-1}(c_{t-1}) \right) \end{aligned}$$

with $0 < \omega \leq 1$.

4. Reduce the strength of any classifier c_t in A_t, by the formula:

$$w_t(c_t) = 1 - \beta w_t(c_t)$$

3.2 Algorithms for fuzzy classifiers

We have extended the above presented algorithms to learn fuzzy models, following the considerations discussed here below.

A *real-valued state* is a set of real values for the input variables. We say that a real-valued state s matches a fuzzy state $\tilde{s}^i$ to some extent $\mu_{\tilde{s}^i}(s)$, when each real value s^k belonging to s belongs to the corresponding fuzzy set belonging to $\tilde{s}^i$, and $\mu_{\tilde{s}^i}(s)$ comes from the conjunction of the values $\mu_{\tilde{s}^i}(s^k)$, computed by the selected conjunction operator. Such operators belong to the class known as *T-norms* [Klir *et al.*1997], among which the most common are *min* and *product*.

A crisp state usually matches more than one fuzzy state, while in standard LCS, it matches the set of intervals that define only one *interval state*. Let us consider subpopulations of classifiers for each fuzzy (or interval) state. In crisp LCS, classifiers belonging to the only subpopulation that matches the current state put actions in the action set, whereas in fuzzy LCS, we have many subpopulations matching the same crisp state, each proposing an action with some strength, proportional to the degree of matching of the classifier selected within the subpopulation. The strength values of the proposed actions are composed by an *aggregation* operator, usually implemented by a *T-conorm* (such as *max*), and then defuzzyfied by one of the many methods available [Klir *et al.*1997]. Whatever *T-norm*, *T-conorm*, and defuzzyfication method is selected, the resulting action proposed by the set of selected fuzzy classifiers $\bar{a}$ is a function of the actions a^i proposed by the matching classifiers and of their degrees of matching $\mu_{\tilde{s}^i}$.

Since all the matching classifiers contribute to the performance of a Fuzzy LCS, the reinforcement should be distributed to all of them, proportionally to their contribution. Therefore, the first step to extend the reinforcement distribution algorithms consists of introducing a factor (let us call it ξ_{c^i}) that weights each classifier contribution. For each step, this factor is equal to the current contribution of the classifier (its degree of matching), weighted by the sum of the contribution given till now, saturated to a given value T (in these experiments $T = 2$). This enhances the learning rate of classifiers that do not have participated too much to the performance, yet.

$$\xi_{c^i} = \frac{\mu_{\tilde{s}^i}(s_t)}{min\left(T, \sum_{k=1,t} \mu_{\tilde{s}^i}(s_k)\right)}$$

The other important point concerns the evaluation of the best value in a given state, used in the $Q-learning$ and $TD(\lambda)$ formulas. In this case, we cannot just take the value corresponding to the best value in the given state, since this state matches many fuzzy states, each contributing with different actions. We present our proposal for Q-learning, leaving to the reader the analogous passages for $TD(\lambda)$. Let us consider $Q^*(c_t^{*i})$, the highest values of the classifiers c_t^{ji} belonging to each subpopulation corresponding to the matching fuzzy states i at time t.

$$Q^*(c_t^{*i}) = max_j(Q(c_t^{ji}))$$

Let us call $m_t(s)$ the sum of the $Q^*(c_t^{*i})$, each weighted by $\mu^*_{\tilde{s}^i}(s)$, the degree of matching of the corresponding classifier the current, real-valued state s. This means that we consider as the reference best value in the current state, the combination of the best classifiers that can trigger in this state, one for each matching subpopulation. The corresponding formula is:

$$m_t(s) = \sum_{k=1,K} \mu^*_{\tilde{s}^k}(s)\, Q^*(c_t^{*k})$$

3.2.1 Fuzzy Q-Learning

At time t the system is in the state s_t, after having executed action a_{t-1} from a state s_{t-1} matched by the fuzzy antecedent $\tilde{s}_{t-1}$, and receives the reinforcement r_t. The Q-value given to the classifiers c^i_{t-1} selected

by the policy for each pair $\langle \tilde{s}^i_{t-1}, a^i_{t-1} \rangle$ is updated by the formula:

$$\begin{aligned} Q_t(c^i_{t-1}) &= Q_{t-1}(c^i_{t-1}) + \\ & \alpha\, \xi_{c^i_{t-1}} \left(r_t + \gamma\, m_{t-1}(s_{t-1}) - Q_t(c^i_{t-1}) \right) \end{aligned}$$

The accuracy of the classifier c^i_{t-1} is updated by the formula:

$$\begin{aligned} E_t(c^i_{t-1}) &= E_{t-1}(c^i_{t-1}) + \\ & \omega\, \xi_{c^i_{t-1}} \left(|r_t + \gamma\, m_{t-1}(s_{t-1}) - Q_t(c^i_{t-1})| - \right. \\ & \left. E_{t-1}(c^i_{t-1}) \right) \end{aligned}$$

3.2.2 Fuzzy $TD(\lambda)$

The extension of the $TD(\lambda)$ algorithm presented for crisp LCS is straightforward, given the considerations done at the beginning of this section. We do not present again all the steps, but only the key formulas.

The eligibility trace of the fuzzy state $\tilde{s}$ is updated by the formula:

$$e_t(\tilde{s}^i) = \begin{cases} \mu_{\tilde{s}^i}(s) & if\ \tilde{s}^i_t = \tilde{s}^i_{t-1} \\ \gamma\lambda \mu_{\tilde{s}^i}(s) e_{t-1}(\tilde{s}^i) & otherwise \end{cases}$$

The strength of the classifier c_{t-1}, is updated by the formula:

$$\begin{aligned} w_t(c^i_{t-1}) &= w_{t-1}(c^i_{t-1}) + \beta\, \xi_{c^i_{t-1}} (r_t + \\ & \gamma \sum_{\forall i} \mu_{\tilde{s}^i}(s_t) V(\tilde{s}^i) - V_{t-1}(\tilde{s}_{t-1})) \end{aligned}$$

The accuracy of the classifier c^i_{t-1} is updated by:

$$\begin{aligned} E_t(c^i_{t-1}) &= E_{t-1}(c^i_{t-1}) + \\ & \omega\, \xi_{c^i_{t-1}} (| \sum_{\forall i} \mu_{\tilde{s}^i}(s_t) V(\tilde{s}^i) - V_{t-1}(\tilde{s}_{t-1})| - \\ & E_{t-1}(c^i_{t-1})) \end{aligned}$$

3.2.3 Fuzzy Bucket Brigade

At time t the reinforcement distribution algorithm execute these steps.

1. Receive the eventual reinforcement r_{t-1} due to the action a_{t-1} selected by the policy in state s_{t-1} and the new action a_t selected by the policy to be done in the current state s_t.

2. Compute the bid basing on the action sets of each subpopulation:

$$Bid_t = \sum_{\forall i, \forall c^i_t \in A^i_t} \beta \mu_{\tilde{s}^i}(s) w_t(c^i_t)$$

3. $\forall i, \forall c^i_t \in A^i_t$ reduce the strength of c^i_t, by a quantity equal to its contribution to the bid:

$$w_t(c^i_t) = w_{t-1}(c^i_t) + (1 - \beta w_t \mu_{\tilde{s}^i}(s))$$

4. Distribute the bid to the classifiers belonging to the action set of the previous step $\forall i, \forall c^i_{t-1} \in A^i_{t-1}$

$$w_t(c^i_{t-1}) = w_{t-1}(c^i_{t-1}) + \xi_{c^i_{t-1}} (Bid_t + r_t)$$

5. Update the accuracy of any classifier c_{t-1} in A_{t-1} by:

$$\begin{aligned} E_t(c^i_{t-1}) &= E_{t-1}(c^i_{t-1}) + \\ & \omega\, \xi_{c^i_{t-1}} (|w_t(c^i_{t-1}) - w_{t-1}(c^i_{t-1})| - \\ & E_{t-1}(c^i_{t-1})) \end{aligned}$$

4 Experimental results

In this section, we first present the experimental settings, then discussing some of the results we obtained. Since the aim of these experiments is the comparison among different reinforcement distribution algorithms applied to fuzzy and crisp LCS, we kept fixed all the other learning parameters and procedures. Namely, the exploration policy is a Boltzman selection, with temperature equal to 5. The classifiers are generated by cover detection, and the worst matching the current state is deleted whenever the population size reaches a given threshold. Classifiers are stochastically selected for genetics by strength. The mutation probability is 0.3, the allele mutation probability is 0.1, and the probability of introducing a "don't care symbol" is also 0.3. Each trial lasts 20,000 steps and sets of trials to be compared with each other have the same stochastic seed.

4.1 The application

The application we have selected is navigation of a simulated mobile robot in a simplified environment. The robot we have considered is CAT [Bonarini1996], a mobile base with car-like kinematics, 700 mm long, 600 mm wide, running at a maximum speed of 300 mm/sec both forward and backward. The maximum steering degree is 30° on each side. In the configuration we adopted for these experiments, CAT had 8 bumpers (on/off contact sensors) and six sonar sensors, placed as shown in figure 2. Each sonar produces an ultrasonic wave and the time between emission and detection of a reflected wave (*Time of Flight - ToF*) is measured. This is proportional to the distance from the closer surface orthogonal to one of the rays of a

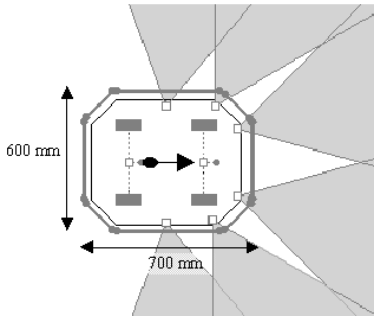

Figure 2: The upper view of CAT. The sonar sensibility cones are represented in gray, and the bumpers as solid lines around the robot body.

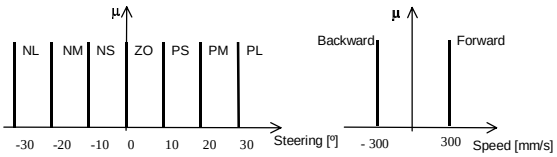

Figure 3: The output values of steering and speed.

60° cone originating from the sonar. The range is between 200 to 3,200 mm. The distance obtained by each sonar is affected by a noise, uniformly distributed in the interval ±5% of the maximum range. This model is rough, but realistic, and may be implemented by coupling two sensors available on the market, each having a detection cone of about 30°.

Data from bumpers are interpreted as eight different discrete values of the same input variable. Data from sonars are interpreted either in terms of fuzzy sets, or in terms of intervals, as described in figure 1. Notice the special singleton value used to represent a characteristic feature of sonars: when no echo returns within the maximum ToF, the sensor gives a specific *out of range* value. This happens both when no object is within the range, and when the object deflects the ultrasonic wave, due to its relative direction. A state is thus represented by six sonar variables and one for the bumpers. The action variables are speed and steering, represented by singletons, as in figure 3. Notice that, while crisp LCS can only produce one of the 7 values for steering (and one of the two for speed), fuzzy LCS, by composition of the actions proposed by each classifier belonging to the different subpopulations that cover the state, produce a real value for each of the output variables.

The reinforcement function r is the same in all the experiments, and it is designed to evolve a controller that makes the agent go forward, as far as possible from obstacles and walls. We are not concerned in this paper to optimize the reinforcement function for the task. In the formula below, V is the speed of CAT, D is the distance from the closer object, and K is a normalizing constant to keep this part of the reinforcement in the interval [0, 1].

$$r_t = \begin{cases} KVD & \text{if CAT goes forward, and does not bump} \\ 0 & \text{if CAT does not go forward, and does not bump} \\ -1 & \text{if CAT bumps} \end{cases}$$

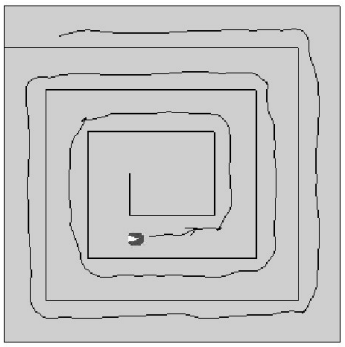

Figure 4: The environment for the experiments with Q-learning and Bucket Brigade, and one of the learnt fuzzy behaviors: CAT (black and white dot) starts from the center and goes along the corridors.

From the final set of learnt classifiers we consider only the best classifier for each subpopulation that have a strength higher than a threshold value (in this experiments, 0), and we evaluate their performance.

4.2 Results

The first results we are presenting concern the comparison between Q-learning and Bucket Brigade with crisp and fuzzy classifiers. The environment in which CAT operates is shown in figure 4. It is a shell labyrinth; each corridor is two meters wide. A detailed analysis of the characteristics of this environment is beyond the scope of this paper. We would like to notice only that there is some perceptual aliasing, and that there are qualitatively different sections of the environment to be faced (the most evident: straight and L-shaped corridors).

All the algorithms produce satisfying controllers. In general, the fuzzy controllers contain more classifiers

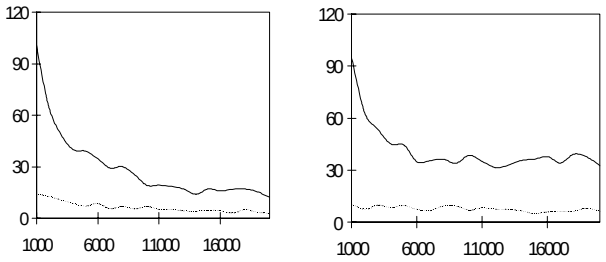

Figure 5: Generation frequence in Q-learning: crisp (left) and fuzzy (right)

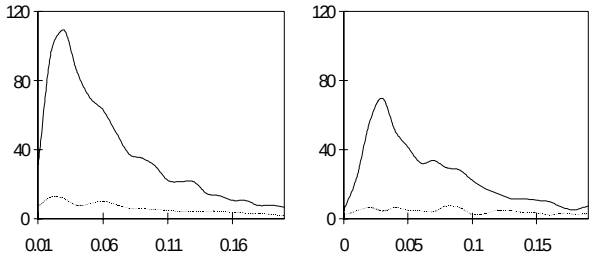

Figure 6: Accuracy in Q-learning: crisp (left) and fuzzy (right)

(about 800 vs. 600 crisp classifiers). This comes from the fact that more fuzzy states than interval states match the crisp states visited by CAT, and also that more fuzzy classifiers have the chance to obtain reinforcement, since more than one subpopulation matches a crisp state. In general, fuzzy controllers are a little bit more effective than crisp, and more robust with respect to increased noise or errors in measurements or actuation. A discussion about this is beyond the scope of this paper and can be found in [Bonarini1996].

It is interesting to compare some plots shown in figures through 5 to 9; all of them report averaged data from 12 trials of 20,000 sense-act (or control) cycles each.

Figure 5 shows the plots of the distribution of the frequence of classifier generation in crisp and fuzzy Q-learning (solid line) over time (cycles), and its mean square error (lower line). We may notice that the crisp LCS generates more classifiers at the beginning, then reaching a stable, quite low generation rate, while fuzzy LCS keeps a high generation rate also later on. This means that fuzzy LCS is still actively searching for an optimal configuration after 20,000 cycles. Therefore, we may expect that accuracy is higher for crisp LCS, since each crisp classifier is reinforced in average, more times than fuzzy. This is confirmed by

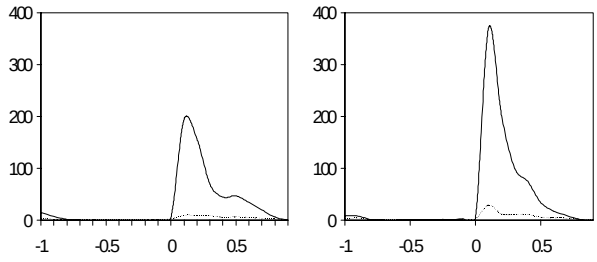

Figure 7: Distribution of classifier strength in Q-learning: crisp (left) and fuzzy (right)

figure 6. Comparing the distribution of the strength among classifiers (figure 7) between fuzzy and crisp LCS, we may notice that a relatively higher number of fuzzy classifiers concentrates in a smaller set of strength values around 0.2, whereas the strength of crisp classifiers is more distributed towards high values. This may be explained by noticing that fuzzy classifiers receive a small reinforcement also when the speed is not the maximum, since the speed they produce may take values in the full range [-300, 300]. Crisp classifiers are reinforced only when the speed is 300 mm/sec, therefore only the best classifiers survive. This is a characteristic of fuzzy LCS, and a full discussion about it will be presented in a forthcoming paper, where we will also explain why fuzzy LCS seem to work better with delayed reinforcement, and crisp LCS with continuous reinforcement.

We cannot show the plots concerning the Bucket Brigade algorithm in this apper. The generation frequence is similar for both the crisp and the fuzzy version, and similar to that of the crisp Q-learning: the algorithm seems to converge to a small rate of population updating. The accuracy of most of the fuzzy classifiers is low, whereas that of crisp classifiers distributes almost uniformely on the whole range. Bucket Brigade tends to reinforce classifiers proposing the same action. This is in contrast with the fuzzy philosophy, where the agent action comes from the combination of different actions proposed by different classifiers. The strength distribution confirms this, showing a high number of fuzzy classifiers with similar, relatively low strength, whereas crisp classifiers are more distributed, and more of them reach higher values. The conclusion may be that Bucket Brigade is inappropriate for fuzzy LCS, but we are identifying some interesting application niches, where fuzzy LCS with high level of generalization show interesting results. However, it is too early to make general considerations about this topic.

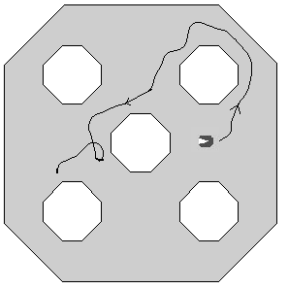

Figure 8: The environment for the experiments with TD(λ)

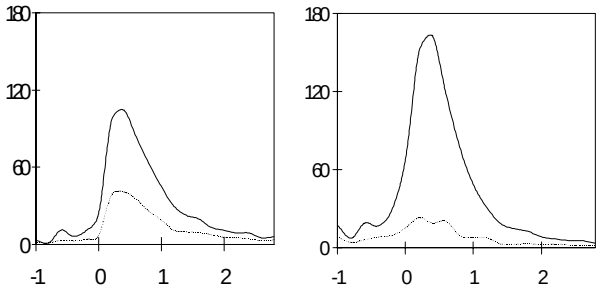

Figure 9: Distribution of classifier strength in TD(λ)

The second experiment we propose concerns $TD(\lambda)$. The environment is shown in figure 8. We cannot show all the plots also for this algorithm. We only notice that the frequency of classifier generation has a behavior similar to the others, reaching and equilibrium before them, thus showing that $TD(\lambda)$ is faster than other algorithms. The accuracy has the same qualitative shape of that of Q-Learning, but, here, fuzzy classifiers are less distributed. From figure 9 it is possible to notice that not only fuzzy rules are more accurate, but also that they are stronger, since, in average, they have received a larger amount of reinforcement each, given that the same reinforcement is distributed to a number of fuzzy classifiers, but to only one crisp classifier.

5 Conclusions

We have presented an extension of classical reinforcement learning algorithms to crisp and fuzzy LCS and a first comparison among some of them. This would like to be a first contribution towards the foundation of a methodology to select components of learning systems to face specific applications. We have also shown that these algorithms can be applied to fuzzy LCS to learn robust mapping from real-vauled input to real-valued output. Our model is very simple, and more investigation is needed to cope with more complex models [Cordon *et al.*1997], once the principal properties have been clarified. We are also working on the identification of characteristics of the environment and of the model (variables, values, granularity) in order to match them with characteristics of the different learning algorithms: reinforcement function, reinforcement distribution, exploration policy, genetics.

Acknowledgments

This research has been partially supported by the MURST Project CERTAMEN. I an indebted with Nicolino De Marco who implemented the tool supporting this research, and with Matteo Matteucci and Claudio Bonacina for many interesting discussions about the experiments.

References

[Bonarini and Basso1997] A. Bonarini and F. Basso. Learning to compose fuzzy behaviors for autonomous agents. *International Journal of Approximate Reasoning*, 17(4):409 – 432, 1997.

[Bonarini1996] A. Bonarini. Evolutionary learning of fuzzy rules: competition and cooperation. In W. Pedrycz, editor, *Fuzzy modeling: paradigms and practice*, pages 265–284, Norwell, MA, 1996. Kluwer Academic Press.

[Bonarini1997] A. Bonarini. Anytime learning and adaptation of hierarchical fuzzy logic behaviors. *Adaptive Behavior Journal*, 5(3 – 4):281 – 315, 1997.

[Bonarini1998] A. Bonarini. Reinforcement distribution to fuzzy classifiers: a methodology to extend crisp algorithms. In *IEEE International Conference on Evolutionary Computation – WCCI-ICEC'98*, volume 1, pages 51–56, Piscataway, NJ, 1998. IEEE Computer Press.

[Cordon *et al.*1997] O. Cordon, F. Herrera, and M. Lozano. On the combination of fuzzy logic and evolutionary computation: a short review and bibliography. In W. Pedrycz, editor, *Fuzzy Evolutionary Computation*, pages 33 – 44, Norwell, MA, 1997. Kluwer Academic Press.

[Klir *et al.*1997] G. J. Klir, B. Yuan, and U. St. Clair. *Fuzzy set theory: foundations and applicatons*, volume 49A. Prentice-Hall, Englewood Cliffs, MA, 1997.