An Extension to the XCS Classifier System for Stochastic Environments

Pier Luca Lanzi and Marco Colombetti
Politecnico di Milano Artificial Intelligence and Robotics Project
Dipartimento di Elettronica e Informazione
Politecnico di Milano
{lanzi,colombet}@elet.polimi.it

Abstract

We analyze XCS learning capabilities in stochastic environments where the result of agent actions can be uncertain. We show that XCS can cope when the degree of uncertainty is limited. We propose an extension to XCS, called XCSμ, which can learn optimal solutions for higher degrees of uncertainty. We test XCSμ when the uncertainty affects the whole environment and when the uncertainty is limited to some areas. Finally, we show that XCSμ is a proper extension of XCS in that it coincides with it when it is applied to deterministic environments.

1 INTRODUCTION

The XCS classifier system [6] represents a major advance in learning classifier systems research because of its accurate generalization mechanism, and its learning mechanism which is based on Q-learning [5], the most known reinforcement learning technique. Wilson's XCS classifier system mainly differs from traditional Holland's classifier systems [2] in the definition of classifier fitness which in XCS is based on the *accuracy* of the classifier prediction instead of the prediction itself as in Holland's systems.

In XCS, *classifier accuracy* measures the degree of overgeneralization of classifiers. A classifier is considered accurate if its prediction closely estimates the payoff level in every situation in which it matches. Conversely, a classifier *becomes* inaccurate when, being overgeneral, it applies in different situations corresponding to different payoff levels. When XCS is applied to deterministic and completely observable environments, the accuracy parameter indeed identifies the only source of inaccuracy, i.e., overgeneralization. However, when considering more general environments there are two other sources of inaccuracy. One of these is experienced when the agent faces an environment which is partially observable with respect to the agent sensors. In such cases, the agent is not always able to determine the best action only looking at its sensors because two or more situations may be aliased. If these situations correspond to different payoffs levels, classifiers matching them tend to become inaccurate. The last, and probably most common, source of inaccuracy is found in *stochastic* environments where the result of agent actions may be uncertain. Classifiers matching these situations to become inaccurate because the actions they advocate may have different effects leading to different payoffs.

Note that in reinforcement learning algorithms, like Watkin's Q-learning [5], the effects of these three sources of inaccuracy (i.e., overgeneralization, perceptual aliasing, and uncertainty) are blended together. On the other hand, XCS is able to single out the inaccuracy due to overgeneralization [6] only when the environment is both completely observable and deterministic. Lanzi [3] showed that when the environment is partially observable memory can be added to XCS in order to disambiguate perceptually aliased situations. But no results have been presented in the literature for applying (Michigan-style) learning classifier systems in stochastic environments.[1] In this paper we study the behavior of XCS in stochastic environments, where the effect of agent actions is affected by uncertainty. Note that we do not consider the case of uncertainty affecting the agent sensors which indeed is an important topic that we plan to address in the future.

Initially, we analyze XCS performance in stochastic environments for different degrees of uncertainty. Our

[1] Schultz [1] presented experimental results for applying the Samuel Learning System, a Pittsburgh-style learning classifier system, to stochastic/noisy environments.

results show that XCS can learn an optimal policy when the uncertainty is limited, although significant. We analyze these results and suggest that this happens because XCS can be unable to separate the inaccuracy introduced by overgeneralization from the inaccuracy due to the uncertainty in the environment. We propose an extension to XCS, called XCSμ, in which a new parameter is added to classifiers to estimate the degree of uncertainty that the classifier experiences. XCSμ uses this parameter to try to separate the two sources of inaccuracy. We show that XCSμ can learn optimal solutions under a higher degree of uncertainty both when this affects the whole environment or only some areas. Finally, we show that XCSμ is a proper extension of XCS in that it coincides with it when the environment is deterministic.

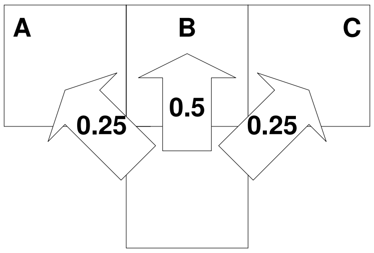

Figure 1: The transition function used in the experiments in this paper. The parameter ϵ is set to 0.5. When the agent performs action *go north* with probability 0.5 $(1-\epsilon)$ the action produces the expected result; with probability $(\epsilon/2)$ the agent slips to the left or right cell respectively.

2 THE XCS CLASSIFIER SYSTEM

Classifiers in XCS have three main parameters: (i) the prediction p, which estimates the payoff that the system expects if the classifier is used; (ii) the prediction error ε, which estimates the error of the prediction p; finally (iii) the fitness F, which estimates the accuracy of the payoff prediction given by p and thus is a function of the prediction error ε.

At each time step, the system input is used to build the *match set* [M] containing the classifiers in the population whose condition part matches the sensory configuration. If the match set is empty a new classifier which matches the input is created through *covering*. For each possible action a_i in the match set the *system prediction* $P(a_i)$ is computed as the fitness weighted average of the classifier predictions that advocate the action a_i in the match set [M]. The value $P(a_i)$ gives an evaluation of the expected payoff if action a_i is performed. Action selection can be *deterministic*, i.e. the action with the highest system prediction is chosen, or *probabilistic*, i.e. the action is chosen with a certain probability among the actions with a non-null prediction. The classifiers in [M] which propose the selected action form the current *action set* [A]. The selected action is then performed in the environment and a scalar reward r is returned to the system together with a new input configuration.

Classifiers parameters are updated as follows. First, the Q-learning-like payoff P is computed as the sum of the reward received at the previous time step and the maximum system prediction, discounted by a factor γ $(0 \leq \gamma < 1)$. P is used to update the prediction p by the delta rule with learning rate β $(0 \leq \beta \leq 1)$: $p_j \leftarrow p_j + \beta(P - p_j)$. Likewise, the prediction error ε is adjusted with the formula: $\varepsilon \leftarrow \varepsilon_j + \beta(|P-p| - \varepsilon)$. The fitness update is slightly more complex. Initially, the prediction error is used to evaluate the classification accuracy κ of each classifier as $\kappa = \exp(\ln \alpha(\varepsilon - \varepsilon_0)/\varepsilon_0)$ if $\varepsilon > \varepsilon_0$ or $\kappa = 1$ otherwise. Subsequently the relative accuracy κ' of the classifier is computed from κ as $\kappa' = \kappa / \sum_{[A]_{-1}} \kappa$. Finally the fitness parameter is adjusted by the rule $F \leftarrow F + \beta(\kappa' - F)$. The genetic algorithm in XCS is applied in the environmental niches represented by the action set $[A]_{-1}$ corresponding to the previous time step.

3 DESIGN OF EXPERIMENTS

The experiments we present in this paper have been conducted in the *woods* series of environments. These are grid worlds in which each cell can contain an obstacle (a "T" symbol), a goal (an "F"), otherwise it can be empty. An agent placed in the environment must learn to reach goal positions. The agent perceives the environment by eight sensors, one for each adjacent cell, and can move into any of the adjacent cells. If the destination cell contains an obstacle the move does not take place; if the destination cell is blank then the move takes place; finally, if the cell contains a goal the agent moves receiving a constant reward, and the problem ends.

Each experiment consists of a number of problems that the agent must solve. For each problem the agent is randomly placed in a blank cell of the environment; then it moves under the control of the system until it reaches a goal position receiving a constant reward, and the problem ends. The agent can solve a problem by *exploring* the environment trying to learn a better solution; otherwise, the agent can solve a problem *exploiting* the knowledge it has acquired. In the former case we say that the agent solves a *learning problem* or,

T	T	F		
T	T	T		
T	T	T		

Figure 2: The `Woods1` environment.

equivalently, that the agent solves the problem *in exploration* [6]; in the latter we say that the agent solves a *testing problem* or, equivalently, that the agent solves the problem in *exploitation*.

At the beginning of a new problem the agent decides with probability 0.5 whether it will solve the problem in exploration (i.e., if it will solve a *learning problem*) or in exploitation (i.e., if it will solve a *testing problem*). When solving a learning problem the system selects the actions to be performed randomly and the genetic algorithm is in operation. When solving a testing problem the system selects the action that predicts the highest payoff and the genetic algorithm is *not* in operation. The performance of XCS is computed as the average number of steps to a goal position in the last 50 testing problems. Every statistic presented in this paper is averaged over ten experiments.

4 STOCHASTIC ENVIRONMENTS

To study XCS performance is stochastic environments we employ the woods environments in which the usual deterministic transition function is modified adding uncertainty to the results of agent actions. The stochastic version of the woods environments works as follows. When the agent tries to move in a certain direction it has probability $1-\epsilon$ ($0 \leq \epsilon < 1$) of reaching the correct destination; it has probability ϵ of slipping, reaching one of the two positions adjacent to its original destination. The value ϵ represents the degree of uncertainty affecting agent actions. An example when ϵ is to 0.5 is illustrated in Figure 1.

We analyze XCS performance in a series of stochastic environments to test how much the generalization mechanism of XCS is influenced by the uncertainty of the actions. For this purpose we use the stochastic version of `Woods1` (Figure 2), `Woods1`ϵ for short, and the stochastic version of `Maze4` (Figure 3), `Maze4`ϵ for short. We apply two versions of XCS in these environments. The former is XCS as originally defined by Wilson [6], that is, when generalization is operating; the

T	T	T	T	T	T	T	T
T			T			F	T
T	T			T			T
T	T		T			T	T
T							T
T	T		T				T
T					T		T
T	T	T	T	T	T	T	T

Figure 3: The `Maze4` environment.

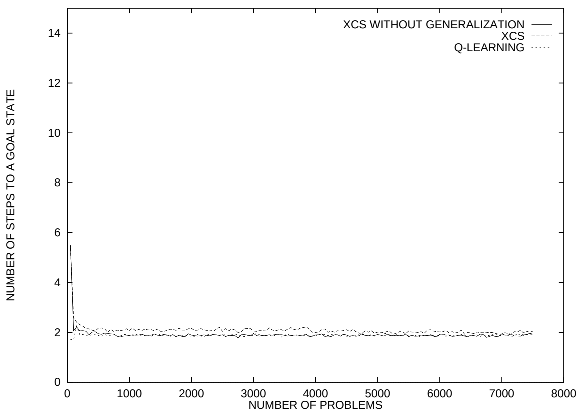

Figure 4: Performance of XCS when generalization is operating, dashed line, and when the generalization is turned off, solid line, in `Woods1`ϵ. Parameter ϵ is set to 0.12. Performance of tabular Q-learning indicates optimal performance.

latter is XCS when generalization is *not* operating.[2] In the first experiment, we apply these two versions of XCS to `Woods1`ϵ when ϵ is 0.1 and 0.50. Population size is set to 400 classifiers, while general parameters are as follows: β=0.2, γ=0.71, θ= 25, ε_0=.01, χ=0.8, μ=0.01, ϕ=0.5.[3] The results of these experiments are depicted respectively in Figure 4 and Figure 5. In each plot we use the performance of tabular Q-learning to indicate optimal performance. As can be noticed, when the uncertainty on the agent actions is limited (i.e. ϵ is 0.1), XCS converges very near to optimal performance: the plots of XCS, XCS with no generalization, and Q-learning (which represents optimal performance) are almost indistinguishable (Figure 4). The

[2] Don't care symbols (#) are not inserted in the newly created classifiers, nor by the mutation operator.

[3] Some of these parameters have not been presented in the overview in Section 2 but are reported here for the sake of completeness. We refer the interested reader to Wilson's original paper [6] for a complete discussion of XCS parameters.

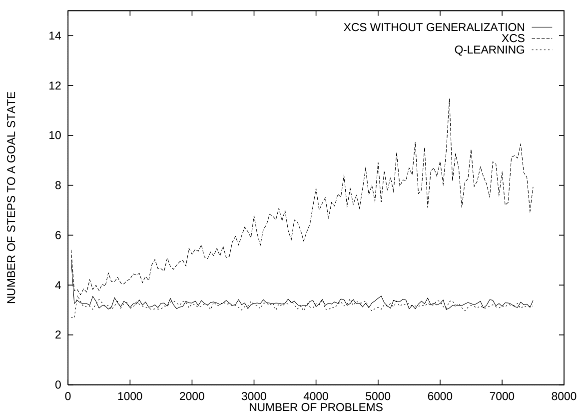

Figure 5: Performance of XCS when generalization is operating, dashed line, and when the generalization is turned off, solid line, in `Woods1`ϵ. Parameter ϵ is set to 0.50. Performance of tabular Q-learning indicates optimal performance.

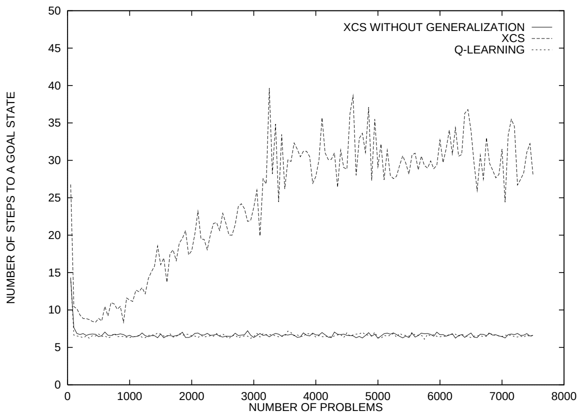

Figure 6: Performance of XCS when generalization is operating, dashed line, and when the generalization is turned off, solid line, in `Maze4`ϵ. Parameter ϵ is set to 0.50. Performance of tabular Q-learning indicates optimal performance. Curves are averaged over ten runs.

performance of XCS, XCS with no generalization, and Q-learning are still indistinguishable when ϵ is 0.25 (results not reported). However, this does not hold anymore when the uncertainty is higher, ϵ is 0.5.

The results plotted in Figure 5 show that, when ϵ is 0.5, XCS cannot converge to the optimum if generalization is operating. The analysis of the population dynamics in each single run, shows that during the first two hundred problems the agent has tried few stochastic transitions that affected only a subset of classifiers. Consequently only few classifiers become inaccurate because of the uncertainty in the environment; in fact, at the beginning of the plot in Figure 5 XCS seems to converge to the optimum. However, as the learning proceeds, the inaccuracy caused by uncertainty on agent actions diffuses to the rest of the population. [4] As a consequence, classifiers tend to become inaccurate, and the genetic algorithm in XCS may be unable to distinguish between those classifiers that are inaccurate because they are overgeneral, from the ones that are inaccurate because of the uncertainty in the environment. Since all the classifiers are almost *equally inaccurate*, the genetic algorithm selects the classifier for reproduction almost at random. This phenomenon results in a loss of the pressure toward *truly* more accurate classifiers because these are not reproduced preferentially anymore. The overall result is that performance decreases with wide oscillations.

[4] This phenomenon is analogous to the propagation of reward through the classifiers, because the accuracy parameters (prediction error ε, and fitness F) are updated together with the payoff prediction p.

We find similar results when we apply the two versions of XCS to the stochastic `Maze4`ϵ using a population of 1600 classifiers and the same parameter settings used in the previous experiments. Experimental results reported in [4] show that also in this environment XCS can converge very near to the optimum when the degree of uncertainty is limited, ϵ is 0.1 or 0.25 (not reported). Like in `Woods1`ϵ, XCS performance decrease when ϵ is 0.5. In fact XCS performance in `Maze4`ϵ when ϵ is 0.5 (Figure 6) is similar to XCS performance in `Woods1`ϵ for the same value of ϵ (Figure 6). In particular, during the initial five hundreds problems, XCS performance seems to converge to a near optimal solution; then the performance suddenly decreases and rapidly becomes completely unstable. The behavior we discussed so far becomes more evident when we apply XCS to `Maze4`ϵ when don't cares are *not* inserted during covering but only during mutation. Figure 7 compares the performance of this version of XCS and the optimal performance of tabular Q-learning. Specifically, the average performance over ten runs is plotted with a solid line, while two dashed lines represent two single runs. In this case, classifiers that are inserted in the population cannot be overgeneral because they have no don't cares. Accordingly, the system initially learns an optimal solution for the problems (in Figure 7, XCS performance is optimal for the first part of problems). As evolution proceeds don't cares are inserted in the population and XCS starts to become unable to distinguish between those classifiers that are inaccurate because they are overgeneral from those that are inaccurate because of environmental uncertainty. Accordingly, we have a sudden decrease of XCS per-

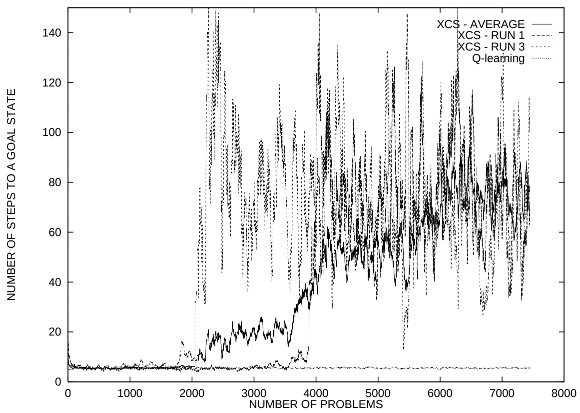

Figure 7: Performance of XCS when don't cares are *not* inserted in newly created classifiers (solid line) in `Maze4`ϵ when ϵ is 0.5. Two single runs are represented by dashed lines. Performance of tabular Q-learning (horizontal dashed line) indicates optimal performance. Curves are averaged over ten runs.

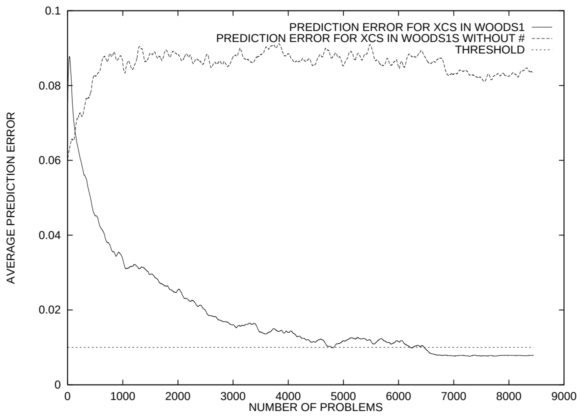

Figure 8: The average prediction error of the classifiers in the population for XCS in the deterministic `Woods1` (solid line) and in the stochastic `Woods1`$_\epsilon$ (dashed line).

formance when this phenomenon becomes significant. As the plots of the two single runs in Figure 7 show this phenomenon can start at different times depending on the exploration in the environment.

Before we proceed any further, we wish to note that our results show that XCS can deal with a significant degree of uncertainty. In fact the system can converge to a performance that is very near to the optimum for values of ϵ up to 0.35 (results not reported). When the degree of environmental uncertainty increases XCS performance can dramatically decrease.

5 DEALING WITH UNCERTAINTY

We showed that in stochastic environments XCS can converge to an optimal solution when the level of uncertainty is limited. For instance, in the stochastic versions of `Woods1` and `Maze4` (`Woods1`ϵ and `Maze4`ϵ), XCS can converge to optimal performance when ϵ is 0.10 or 0.25, while it cannot converge to optimal performance when ϵ is 0.5. This happens because the system is not able to separate the inaccuracy introduced by overgeneralization from the inaccuracy due to the uncertainty. In fact both these sources of inaccuracy are *blended together* in the parameter ε, which represents the prediction error of a classifier.

In these types of environments, the prediction error ε can be thought of as the combination of two distinct components: the prediction error ε_{gen} due to overgeneralization, and the prediction error ε_{env} caused by the uncertainty on the actions. Assume now that the degree of uncertainty in the environment does not change in time, i.e., the transition probability distribution is stationary.[5] An extension of XCS can be developed in which the component ε_{gen} caused by overgeneralization is separated from the component ε_{env} caused by the uncertainty in the environment. To follow the philosophy underlying evolution in Wilson's XCS this extension should separate the two sources of inaccuracy locally, by acting in environmental niches. In the following, starting from the results discussed in the previous section, we introduce an extension of XCS, called XCSμ, capable of separating the two components, ε_{gen} and ε_{env}, of the prediction error.

The analysis of XCS behavior in `Woods1`ϵ, when generalization is not operating, shows the prediction errors of classifiers tend to converge to a value that is an *internal representation* of the uncertainty observed by the agent. In particular, at the beginning classifiers have low prediction errors because their values have been just initialized. As the system learns prediction errors increase since the classifiers experience the uncertainty on their actions. Then, after 1000 problems, prediction errors slightly oscillate because the agent performs random exploration but, on the average, they never go beneath a certain threshold.

This phenomenon is illustrated in Figure 8, where the average prediction error of classifiers for XCS (without generalization) in the deterministic `Woods1` (solid line), and in the stochastic `Woods1` (dashed line) are compared. As can be noticed, in the deterministic case the classifiers become more accurate as the system learns, and the average prediction error decreases. This con-

[5] Note that this is also the assumption that is made in Q-learning to guarantee the convergence to the optimum.

tinues until all the classifiers are accurate since their prediction error is below the threshold specified by the XCS parameter ε_0. In the stochastic case (dashed line), the average prediction error never decreases under a certain threshold that depends on the degree of uncertainty in the environment. If XCS could evaluate such threshold it could also try to separate the two error components. Following this idea, we introduce an extension of XCS in which the system tries to separate ε_{gen} and ε_{env} by estimating that threshold.

When XCS is applied to `Woods1`ϵ the prediction error of classifiers oscillates for two reasons: the possible overgeneralization, and the uncertainty due to the environment. These oscillations are wider than the ones experienced when the system does not perform any generalization. However, if we analyze the evolution of the error of the classifier that has the smallest error in each action set we observe that this statistic is more stable than prediction error. This happens because the prediction error in a certain action set, on the average, does not decrease beneath the threshold defined by the environmental uncertainty that affects agent actions.

Therefore, the minimum prediction error experienced in each environmental niche can be used to evaluate the inaccuracy introduced by the environment. We define a new parameter, μ, estimating the minimum prediction error that the classifier has observed in the positions it matched. Each time the classifier parameters are updated, $\overline{\mu}$ is computed as the minimum prediction error of the classifiers that match the current environmental niche. The new parameter μ is then updated by the usual Q-learning like technique:

$$\mu \leftarrow \mu + \beta_\varepsilon(\overline{\mu} - \mu).$$

Parameter μ estimates the minimum prediction error of the niches experienced by the classifier. The learning rate β_ε is smaller than the β used to update the other classifiers parameters (a typical value is $\beta_\varepsilon = 0.05$). A small β_ε value guarantees that the estimate given by μ is less sensitive to the variance of the population.

The parameter μ is used to separate the prediction error due to uncertainty ε_{env} from the prediction error due to overgeneralization ε_{gen}. The original update formula for the prediction error ε is modified as:

$$\varepsilon \leftarrow \varepsilon + \beta(|P - p| - \mu - \varepsilon)$$

In the formula the minimum prediction error μ is considered an evaluation of the inaccuracy due to the uncertainty on the agent actions. When updating the prediction error ε, μ is subtracted from $|P - p|$ in order to separate the inaccuracy due to the environment (represented by μ) from the total inaccuracy experienced (represented by $|P - p|$). Note that, due to the uncertainty in the environment both μ and $|P-p|$ may oscillate so that the difference $|P - p| - \mu$ can become negative. In general we expect that $|P - p| - \mu \geq 0$ because μ is a lower bound to the prediction error. Accordingly when $|P - p| - \mu$ becomes less than zero because of oscillations the prediction error update is done using ε_0 in place of $|P - p| - \mu$.

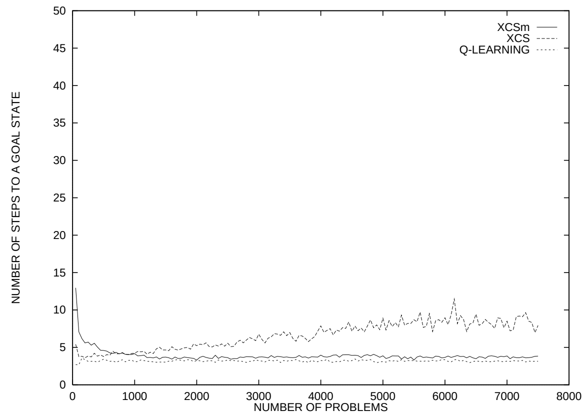

Figure 9: Performance of XCSμ, solid line, and XCS, dashed line, in `Woods1`ϵ. Parameter ϵ is set to 0.50. Performance of tabular Q-learning is used to represent the optimum. Curves are averaged over ten runs.

6 EXPERIMENTAL RESULTS

We now test the extension of XCS we introduced in the previous section by applying XCSμ to the stochastic versions of `Woods1` and `Maze4` when the uncertainty affects the entire environment, as in the previous experiments, and when uncertainty is limited to some areas. Finally, we show that XCSμ is a proper extension of XCS in that it coincides with it when the environment is not stochastic.

6.1 XCSμ WITH GLOBAL UNCERTAINTY

We now test the extension of XCS we introduced in the previous section by applying XCSμ to `Woods1`ϵ and to `Maze4`ϵ when ϵ is 0.5. Parameter β_ε is set to 0.05; general parameters are set as in the previous experiments. First, we compare XCS and XCSμ in `Woods1`ϵ. The experimental results, depicted in Figure 9, show that XCSμ converges very near to the optimal performance in this environment. Note that XCSμ cannot reach optimal performance because it tries to general-

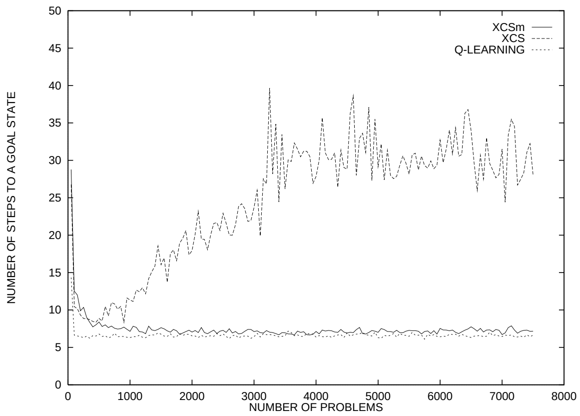

Figure 10: Performance of XCSμ, solid line, and XCS, dashed line, in `Maze4`ϵ. Parameter ϵ is set to 0.50. Performance of tabular Q-learning is used to represent the optimum. Curves are averaged over ten runs.

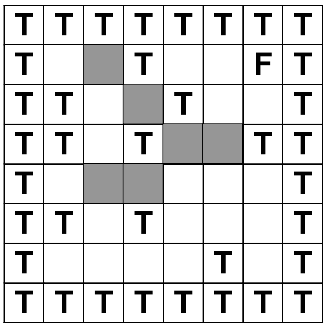

Figure 11: `Maze4` with local uncertainty: in grey cells the results of agent actions can be uncertain.

ize and therefore it has to deal with overgeneralization. We have similar results when XCS and XCSμ are compared in `Maze4`ϵ. Results reported in Figure 10 show that XCSμ can converge to a performance that is near optimal.

6.2 XCSμ WITH LOCAL UNCERTAINTY

Uncertainty rarely affects the whole environment. In most cases uncertainty is only limited to some areas. As an example, consider a wheeled agent moving in an office environment where all the rooms are covered with carpet, except for some rooms that are covered with tiles. The action *turn 90 degrees left* will have different results depending on which type of floor the agent is moving on.

Although an environment with limited uncertainty might appear simpler than one with global uncertainty, it is worth noting that learning with local uncertainty implies that the agent learns that same actions may

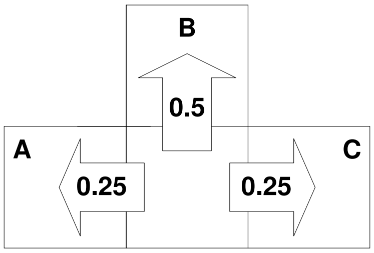

Figure 12: The transition function used in the experiments with the *local* version of `Maze4`.

behave differently in different situations. In particular, when using XCS to learn in presence of local uncertainty, the system must be able to evolve accurate rules that propose the best action in situations free of uncertainty, and rules which indeed propose the best action but are inaccurate because they apply in situation subject to uncertainty.

We apply XCSμ in `Maze4` when only few cells are subject to uncertainty. The new version of `Maze4` is depicted in Figure 11. Grey cells represent positions where the result of agent's action follow the transition model depicted in Figure 12. Note that cells subject to uncertainty separates `Maze4` into two main areas. The idea is to make it more difficult for the agent to reach the goal by having a strip of uncertainty to pass through. Figure 13 compares the performance of XCS, XCSμ, and the optimal performance of tabular Q-learning in `Maze4` with local uncertainty. As can be noticed, XCSμ reaches an optimal solution for this problem. On the contrary, XCS performance is worse than it was when the uncertainty affected all of the environment. This is mainly due to the fact that classifiers matching positions affected by uncertainty tend to reproduce less because they are inaccurate. Moreover, as in the global case, in these positions the system rapidly becomes unable to distinguish between classifiers that are inaccurate because they are overgeneral and classifiers that are inaccurate because of uncertainty. Due to the combination of these two factors, the system tends to learn a partial solution for the problem in which for those cells subject to uncertainty there are no classifiers that propose the best action.

6.3 DETERMINISTIC ENVIRONMENTS

XCSμ is an extension to XCS, in that it attempts to preserve all the main characteristics of the original system. Consequently, we should expect that in deterministic environments XCSμ performs as XCS. To show that this is the case, we apply XCSμ and XCS

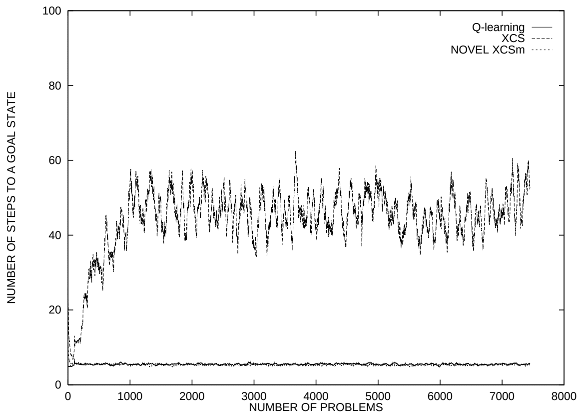

Figure 13: Comparison of XCS performance, solid line, and XCSμ performance, dashed line, in the the version of `Maze4` with *local* uncertainty. Curves are averaged over ten runs.

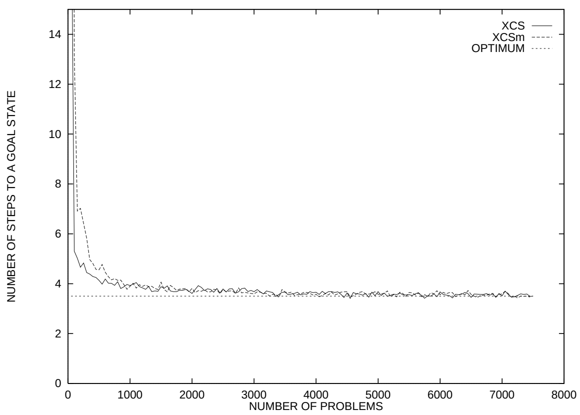

Figure 14: Comparison of XCS performance, solid line, and XCSμ performance, dashed line, in the *deterministic* `Maze4`. Curves are averaged over ten runs.

to the deterministic `Maze4` with the same parameter settings employed in the previous experiment. Figure 14 compares the performances of XCSμ and XCS in `Maze4`. As it can be noticed the performance of the proposed system is almost identical to that of the original. These results confirm that XCSμ is a proper extension of XCS.

7 SUMMARY

We have analyzed XCS behavior in stochastic environments where the results of the agent actions are affected by uncertainty. We have shown that XCS converges to an optimal solution if the degree of uncertainty is significant but limited. Otherwise, XCS cannot converge to an optimal solution because the system may be unable to distinguish between classifiers that are inaccurate because they overgeneral and classifiers that are inaccurate because of the uncertainty of the actions they advocate. We have introduced an extension of XCS, XCSμ, that tries to separate the inaccuracy due to overgeneralization from that due to uncertainty. We have shown that XCSμ converges to optimal performance for a higher degree of uncertainty both when uncertainty affects the whole environment, and when it affects only some areas of the environment. Finally, we have shown that XCSμ is a natural generalization of XCS, and in fact coincides with it when the environment is deterministic.

Acknowledgments

This project was partially supported by the Politecnico di Milano Research Grant "Development of autonomous agents through machine learning," and by the project "CERTAMEN" co-funded by the Italian Ministry of University and Scientific and Technological Research.

References

[1] John Grefenstette and Alan Schultz. Using a genetic algorithm to learn behaviors for autonomous vehicles. In *Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation and Control Conference (AIAA)*, number AIC-92-009, pages 739–749, August 1992.

[2] John H. Holland. *Machine learning, an artificial intelligence approach. Volume II*, chapter Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems, pages 593–623. Morgan Kaufmann, 1986.

[3] Pier Luca Lanzi. An Analysis of the Memory Mechanism of XCS. In J. Koza et al, editor, *Proceedings of the Third Annual Genetic Programming Conference*, pages 643–651, Madison (WI), 1998. Morgan Kaufmann San Francisco (CA).

[4] Pier Luca Lanzi. *Reinforcement Learning by Learning Classifier Systems.* PhD thesis, Politecnico di Milano – Dipartimento di Elettronica e Informazione, Milano, Italy, 1999. Submitted.

[5] C.J.C.H. Watkins. Learning from delayed reward. PhD Thesis, Cambridge University, Cambridge, England, 1989.

[6] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.