How XCS Evolves Accurate Classifiers

Martin V. Butz
Department of Cognitive Psychology
University of Würzburg
Würzburg, 97070, Germany
butz@psychologie.uni-wuerzburg.de
+49 931 312176

Tim Kovacs
School of Computer Science
The University of Birmingham
Birmingham B15 2TT, United Kingdom
T.Kovacs@cs.bham.ac.uk
+44 121 414 4773

Pier Luca Lanzi
Dip. di Elettronica e Informazione
Politecnico di Milano
Milano 20133, Italy
pierluca.lanzi@polimi.it
+39 2 2399 3472

Stewart W. Wilson
University of Illinois at Urbana-Champaign
Prediction Dynamics
Concord, MA 01742, USA
wilson@prediction-dynamics.com
+1 978 369 9232

Abstract

Due to the accuracy based fitness approach, the ultimate goal for XCS is the evolution of a compact, complete, and accurate payoff mapping of an environment. This paper investigates what causes the XCS classifier system to evolve accurate classifiers. The investigation leads to two challenges for XCS, the *covering challenge* and the *schema challenge*. Both challenges are revealed theoretically and experimentally. Furthermore, the paper provides suggestions for overcoming the challenges as well as investigates environmental properties that can help XCS to overcome the challenges autonomously. Along those lines, a deeper insight into how to set the initial parameter values in XCS is provided.

1 INTRODUCTION

After more than twenty years have passed since the first learning classifier system (LCS) approaches (Holland & Reitman, 1978), recently, LCSs appear to reach competence. The XCS classifier system (Wilson, 1995) solved the former main shortcoming of LCSs, that is the problem of strong over-generals (Kovacs, 2001), by its accuracy based fitness approach. Previous LCSs evolved strong rules in terms of rules that encounter high rewards from environments. On the other hand, XCS evolves accurate rules, i.e., rules which accurately predict the payoff encountered after the execution of an action.

Despite this insight, til now it has not been clarified how the genetic algorithm (GA) in XCS can benefit from this approach. Essentially, it is not clear how and when the accuracy based fitness approach pushes the population of classifiers towards accurate classifiers. The aim of this paper is to investigate and clarify the evolutionary pressure in XCS towards accurate classifiers. Along those lines the paper exposes two problem boundaries or challenges that can more or less severely decrease the accuracy pressure. Alternatives how to circumvent the problem are provided. Moreover, environmental properties are investigated that help XCS to evolve accurate classifiers faster. Finally, the utility and best initial setting of several parameters is clarified.

The paper is structured as follows. First, we provide a short overview over the XCS classifier system mentioning all parameters and equations important for the remainder of the paper. Section 3 discusses the accuracy pressure in XCS theoretically, emphasizing two challenges and possible solutions. Next, section 4 validates the proposed pressures and solutions. Moreover, it provides further insight in the best initial parameter setting. Finally, we summarize and conclude the paper.

2 XCS OVERVIEW

The XCS classifier system was developed by Wilson (1995). Although we assume a basic familiarity with the system, this section provides a general overview of XCS displaying the accuracy related methods in detail. For further information the interested reader should refer to Wilson (1995), and the algorithmic description of XCS (Butz & Wilson, 2001).

As in all LCSs and reinforcement learning methods, XCS acts as a learning agent that perceives inputs describing the current environmental state, responds with actions, and receives reward (possibly from a separated reinforcement program) as an indication of the value of its action. The reward received is defined by the *reward function*, which maps state/action pairs to real numbers, and it is part of the problem definition (Sutton & Barto, 1998). For the investigation purposes in this paper we only use *single-step* tasks in which the agent's actions do not influence the successive states. The goal of the agent is to maximize the reward it receives.

When XCS receives an input it forms the *match set* [M] of rules whose conditions match the environmental input. XCS requires that at least θ_{mna} actions are present in a match set (Butz & Wilson, 2001). If this is not the case, covering classifiers will be created with a matching condition. Each attribute in the condition of such a covering classifier is a #-symbol (a so called don't care symbol that matches any input) with a probability of $P_{\#}$ and the corresponding perceived symbol otherwise. Next, XCS selects an action from among those advocated by the rules in [M]. The subset of [M] which advocates the selected action is called the *action set* [A].

In each cycle, XCS updates the rules in [A] based on the reward received. Rules not in [A] are not updated. Moreover, dependent on the threshold θ_{ga} and the average time in $[A]$ since the last GA application, a reproductive event is triggered, in which a GA is called upon to modify the population of rules. Since the GA in XCS only reproduces classifiers currently in [A] it realizes an implicit niching. The GA chooses two classifiers for reproduction proportionally to the fitnesses F of the classifiers in $[A]$. The selected classifiers are reproduced, crossed, mutated, and inserted in the population. The parents stay in the population competing with their offspring. Moreover, subsumption deletion acts in [A] deleting more specific classifiers if an accurate, experienced, and more general classifier exists.

If the number of classifiers in a population exceeds the threshold N, excess classifiers are deleted. Classifiers for deletion are selected in [P] proportionally to their action set size estimate as. If sufficiently experienced and with a significantly low fitness F, the probability of deletion is increased further.

The rule fitness calculation in XCS differs from traditional approaches. In traditional strength-based systems (e.g., Goldberg, 1989; Wilson, 1994), the fitness of a rule is called its *strength*. This value is used in both action selection and reproduction. In contrast, the accuracy-based XCS maintains separate estimates of rule utility for action selection and reproduction.

In single-step tasks an LCS typically uses an update like the following delta rule to update rule strength which is called the reward prediction p in XCS:

$$p \leftarrow p + \beta(R - p) \tag{1}$$

where $0 < \beta \leq 1$ is a constant controlling the learning rate and R is the reward from the environment. From the reward prediction p XCS updates a number of parameters for adjusting its fitness F:

$$\epsilon \leftarrow \epsilon + \beta(|R - p| - \epsilon) \tag{2}$$

$$\kappa = \begin{cases} 1 & \text{if } \epsilon < \epsilon_0 \\ \alpha(\epsilon/\epsilon_0)^{-\nu} & \text{otherwise} \end{cases} \tag{3}$$

$$\kappa' \leftarrow \frac{\kappa}{\sum\limits_{x \in [A]} \kappa_x} \tag{4}$$

$$F \leftarrow F + \beta(\kappa' - F) \tag{5}$$

The parameter ϵ_0 ($\epsilon_0 > 0$) controls the tolerance for prediction error ϵ; α ($0 < \alpha < 1$) and ν ($\nu > 0$) are constants controlling the rate of decline in accuracy κ when ϵ_0 is exceeded. The updates treat the strength of a rule as a prediction of the reward to be received, and maintain an estimate of the error ϵ of its reward prediction. An accuracy score κ is calculated based on the error as follows. If error is below some threshold ϵ_0 the rule is assumed to be accurate (has an accuracy of 1), otherwise its accuracy drops off quickly. The accuracy values in the action set [A] are then converted to relative accuracies κ', and finally each rule's fitness F is updated towards its relative accuracy. Figure 1 visualizes equation (3). Observable is the idea behind the accuracy calculation: ϵ_0 is the threshold to what extent errors are accepted; α causes a strong distinction between accurate and not quite accurate classifiers; the steepness of the successive slope is influenced by ν as well as ϵ_0.

To summarize, in XCS fitness behaves inversely to the reward prediction error, with errors below ϵ_0 being ignored entirely.

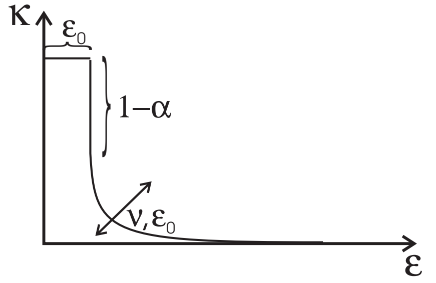

Figure 1: The calculation of the accuracy kappa is crucial for the fitness approach in XCS.

3 FINDING ACCURATE CLASSIFIERS

Although the last section gave an overview of the functioning of XCS, it is not clear how the GA method evolves accurate classifiers. The accuracy based fitness is only a prerequisite for a successful evolution. This section explains first how the pressure towards accurate, maximally general classifiers (i.e. classifiers which are accurate and in the mean time as general as possible) is realized in XCS. Next, it investigates problem boundaries that hinder parts of the pressures and consequently hinder XCS from evolving accurate classifiers. Finally, solutions to those boundaries are proposed.

3.1 THE ACCURACY PRESSURE

The accuracy pressure in XCS is realized by several methods based on the principle of the survival of the fittest and die out of the weak. However, minor modifications and interpretation of these principles give XCS the real power. In the following, we first describe the methods separately and next discuss their interaction.

The reproduction method in the GA realizes the survival of the fittest principle. Reproduced are classifiers with a high fitness which implies a higher accuracy than other classifiers in the same environmental niche. Thus, accurate rather than over-general classifiers are reproduced. (We use the term *environmental niche* referring to one necessary solution of a problem described by a *schema* of lowest possible order (see e.g. Goldberg, 1989) together with its incidental action. The order o of a schema denotes the number of specific attributes in the schema.)

The deletion method, separated from the reproduction process, emphasizes the deletion of classifiers in large niches. Moreover, applying the deletion method in Kovacs (1999), deletion further emphasizes the deletion of inaccurate classifiers.

Together, reproduction and deletion realize a pressure towards the evolution of accurate classifiers in each environmental niche. Moreover, due to the application of the reproduction in the action set and the deletion in the population, the combination also realizes an intrinsic pressure towards more general classifiers as proposed by Wilson's Generality Hypothesis (Wilson, 1995), which was further investigated and enhanced to an Optimality Hypothesis by Kovacs (1997).

This combination of accuracy and generality pressure, however, requires several conditions in order to be applicable. The prerequisites are discovered in the remainder of this section.

3.2 COVERING CHALLENGE

The first rather straightforward challenge is to set the parameters in such a way that the GA actually takes place in XCS. One case in which this prerequisite is not met is expressed by the *covering challenge*. Normally, covering only occurs briefly at the beginning of a run. However, if covering continues indefinitely because inputs continue not to be covered by classifiers, the GA cannot take place in XCS and the accuracy pressure does not apply.

As described in the XCS overview above, dependent on the parameter θ_{mna}, one or more covering classifiers will be created if an input is not sufficiently covered. However, if the population is already filled up with classifiers, other classifiers are deleted to make space for the covering classifiers. In the beginning of a run, with a population of classifiers that have a very low experience, the fitness F as well as the action set size estimate as of these classifiers is basically meaningless. Consequently, the deletion method chooses classifiers for deletion at random. Dependent on the specificity of classifiers generated by covering (determined by the parameter $P_{\#}$) the population has on average a certain specificity in the beginning. If this specificity is too high, it can happen that the population is eventually filled up with over-specific classifiers and the "covering-random deletion" cycle continues forever.

More formally, the probability that an input is covered by at least one classifier in a randomly generated population is:

$$P(\text{cover}) = 1 - (1 - (\frac{1 + P_{\#}}{2})^l)^N \qquad (6)$$

where $P_{\#}$ is the probability of generating a don't care symbol when creating the condition of a classifier, l is the length of the input string, and N is the size of the population. The formula is correct for binary coding where the population was initially filled up with randomly generated classifiers. Moreover, the formula also applies in the case where $P_{\#}$ is too small, so that covering will continue until the population is filled up with different classifiers, and all expressible inputs are equally probable encountered without any dependence on previous states or actions. Note that, in multi-step problems $P(\text{cover})$ varies further since not all possible codings are perceivable and states closer to a goal state are usually encountered more often.

The covering challenge by itself can easily be circumvented by setting the parameter $P_{\#}$ high enough. Considering formula (6) it is possible to calculate a curve dependent on $P_{\#}$. As long as $P(\text{cover})$ is sufficiently larger than zero, the GA eventually takes place and the covering challenge is solved. Moreover, it should be possible to enhance XCS to detect the challenge itself and consequently adapt the $P_{\#}$ parameter autonomously. However, we did not experiment with such an enhancement so far.

3.3 SCHEMA CHALLENGE

Once the covering challenge is solved and the GA takes place, essential for a successful evolutionary process is that the accuracy pressure applies. The *schema challenge* addresses the problem that the accuracy of classifiers possibly does not lead to the accurate, maximally general point. Considering again the specificity of a classifier, this point can be reached from two sides, the over-specific side and the over-general side.

3.3.1 THE OVER-SPECIFIC SIDE

Evolving an accurate, maximally general classifier from the over-specific side is a process that is based on Wilson's *generalization hypothesis* as mentioned in section 3.1. Once an accurate classifier was generated (which can happen by chance in covering or by evolution in the GA) this classifier will eventually have a higher fitness than the majority of the inaccurate classifiers and consequently reproduce more often. To what extent the reproduction rate of accurate classifiers differs from inaccurate ones depends on the parameters α and ν in the determination of the accuracy κ as visualized in figure 1. Setting α low enough and ν high enough ensures that the probability of loosing all accurate classifiers in a specific environmental niche is small. Thus, the intrinsic pressure expressed in the generalization hypothesis ensures a continuous pressure towards generality while the accuracy based fitness prevents from over-generalization. The subsumption deletion method increases the pressure towards accurate, maximally general classifiers from the over-specific side further.

3.3.2 THE OVER-GENERAL SIDE

While the pressure towards generality was already intensively investigated, the evolution from the over-general side appears to be much more awkward. How can an over-general classifier be fitter, when closer to being accurate? The distance is hereby the number of attributes that need to be changed in order to generate an accurate classifier out of the current classifier. The remainder of this section exposes necessary requirements in the environment. Moreover, if those requirements are not met, it exhibits how to circumvent the problem at least in small problem spaces.

Essential for a higher fitness of classifiers which are closer to accuracy is the applicability of the slope in the determination of κ as visualized in figure 1. Thus, it is necessary that the prediction error ϵ of classifiers which are closer to accuracy is lower than the ϵ of classifiers with a higher distance. Two environmental properties can provide such 'hints' from the over-general side. (1) The reinforcement program can provide intrinsic information in its reward function. (2) A bias in the consistency of making a correct/wrong classification when over-general can result in further benefit. Both possibilities are discussed in somewhat more detail below.

Layered Payoff. The first property assumes a bias in the reward function of the reinforcement program where a bias refers to the reward function returning different payoff in different states. The resulting layered payoff landscapes can provide 'hints' towards accuracy. Wilson (1995) used such a landscape for the multiplexer problem and Kovacs (2001) used them to develop a theory of strong over-generals calling a function that creates such landscapes a biased reward function. Moreover, layered payoff landscapes are always present in multi-step problems. In order to be helpful for a problem the difference in payoff between a correct and wrong classification must be on average smaller when closer to accuracy. If this property is present in any payoff landscape, XCS is able to exploit the property and consequently discovers the accurate classifiers faster as experimentally validated in section 4.

Biased Generality. While the layered payoff benefit requires an explicit bias in the reward function of a problem, the *biased generality benefit* can also be intrinsically present in a problem even when the reward function is not biased. The idea behind biased generality is that accuracy is greater when a rule is more consistently correct or more consistently wrong. Once a classifier is only correct or wrong it is most accurate. The property can be approached mathematically when assuming a simple $R/0$ (i.e. a two level) payoff landscape, R is provided if a prediction was correct and zero if it was wrong. Let's denote $P_c(cl)$ as the probability that classifier cl predicts the correct outcome. Due to the assumed payoff landscape and the assumption of a uniformly randomly encountering of both cases, the reward prediction $cl.p$ of classifier cl eventually oscillates around $P_c(cl) * R$ where the amount of oscillation can be influenced by β. Neglecting the oscillation and consequently setting $cl.p$ equal to $P_c cl * R$ the following derivation is possible:

$$cl.\epsilon = (R - cl.p) * P_c(cl) + cl.p * (1 - P_c(cl)) = \\ = 2R(P_c(cl) - P_c(cl)^2) \quad (7)$$

The formula sums the two cases of executing a correct or wrong action with the respective probabilities. The result is a parabolic function for the error ϵ that reaches its maximum of 0.5 when $P_c(cl)$ equals 0.5 and is 0 for $P_c(cl) = 0$ and $P_c(cl) = 1$. This shows that if the consistency of a correct/wrong prediction increases, the accuracy and consequently the fitness of a classifier increases.

3.3.3 CIRCUMVENT THE CHALLENGE

However, when neither of the two above properties are given in a problem (i.e. the problem does not provide any hints towards accuracy from the over-general side), it can be very hard for XCS to evolve accurate classifiers out of an over-general population of classifiers. One possibility to circumvent this problem is to set $P_\#$ low enough, so that accurate classifiers for most environmental niches are present due to covering.

Hereby, it is helpful to calculate the probability that a certain environmental niche is represented by at least one classifier. We can think of an environmental niche as a schema (see e.g. Goldberg, 1989) of order o combined with an action. An environmental niche is represented by a classifier if at least all attributes that are specified in the schema are equal in the classifier and the classifier has the same action. The probability of the existence of a representative of a specific schema is determined by the formula

$$P(\text{representative}) = 1 - (1 - \frac{1}{n}(\frac{1 - P_\#}{2})^o)^N \quad (8)$$

where n denotes the number of possible actions. As in equation (6) the equation again assumes a specificity in the population that is identical to $P_\#$. Thus, the schema challenge can be initially circumvented by setting $P_\#$ low enough so that $P(\text{representative})$ is significantly larger than zero.

While the covering challenge requires to set $P_\#$ high enough, the schema challenge—as long as no hints are given from the environment—actually requires $P_\#$ to be low enough. The next section validates all above discussed challenges and properties. Moreover, it shows that the two challenges can interfere consequently hindering XCS from evolving accurate classifiers.

4 EXPERIMENTAL VALIDATION

After many claims have been made in the last section, this section is dedicated to validate all the claims experimentally. In order to show the problem boundaries induced by the two challenges, we first investigate performance in the 20 multiplexer varying $P_\#$. Next, we show that despite the increased boundaries, XCS is able to solve the 37 multiplexer. Moreover, we show the benefit of a biased reward function in the 37 multiplexer and reveal the use of the parameters α and ϵ_0 in the κ function. With the biased reward function we are now even able to solve the 70 multiplexer. Finally, we modify the multiplexer function in order to show the benefit of biased generality.

If not stated differently, all experiments herein are averaged over twenty runs. Performance is measure by altering one pure exploration step with one pure exploitation step in which the percentage of correct classifications is recorded. The different methods in XCS go along with the recently published algorithmic description (Butz & Wilson, 2001). Essentially, we use the niche mutation type, the deletion method as investigated by Kovacs (1999), and GA- as well as action set subsumption. The parameters were set as follows: $N = 2000$, $\beta = 0.2$, $\alpha = 0.1$, $\epsilon_0 = 10$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, $p_I = 10$, $\epsilon_I = 0$, $F_I = 0.01$, $p_{explr} = 1$, and $\theta_{mna} = 2$. Note that γ is irrelevant since we only investigate single-step problems. Parameter $P_\#$ is set in each experiment separately.

4.1 THE TWO CHALLENGES

Before we present the challenges in the problem, we need to explain the multiplexer problem itself. The multiplexer function is defined for binary strings of length $k + 2^k$. The output of the function is determined by the bit referred to by the first k bits. In the six multiplexer for example, $f_{6-MP}(100010) = 1$ or $f_{6-MP}(000111) = 0$. Any multiplexer function can also be expressed in disjunctive normal form. The six multiplexer can be expressed as:

$$f_{6-MP}(x_1, ..., x_6) = \\ x_1 x_2 x_6 \vee x_1 \neg x_2 x_5 \vee \neg x_1 x_2 x_4 \vee \neg x_1 \neg x_2 x_3 \quad (9)$$

The symmetry of the multiplexer problem results in most over-general cases in no generality bias at all (i.e. the probability for an over-general classifier of being

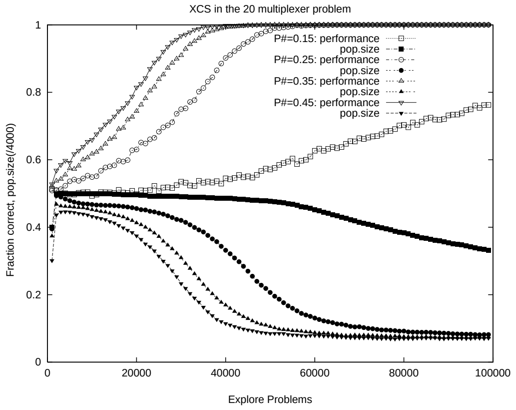

Figure 2: In the 20 multiplexer problem the covering challenge is easily observable.

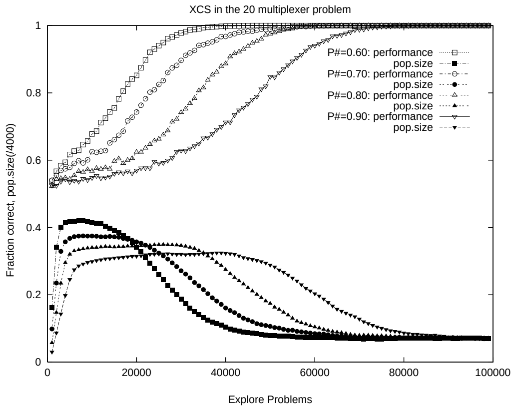

Figure 3: Despite the observable schema challenge XCS is always able to solve the problem.

correct is very close to 50%). Providing a 1000/0 payoff the payoff is not biased and no benefit can be drawn in this respect, either. Thus, the cover challenge from the over-specific side as well as the schema challenge from the over-general side should be observable.

Figure 2 exhibits the covering challenge. If the don't care probability $P_\#$ is set too low, the population is filled up with classifiers and has difficulties to start with the GA application. Increasing $P_\#$ results in a faster learning curve as well as a faster convergence in the population. Due to the generalization pressure, the population size decreases once the GA starts being applied and XCS is able to evolve a complete and accurate representation of the problem. The relation with formula (6) is observable in figure 4. At a don't care probability of 0.15 the curve is very close to zero consequently causing the covering challenge. With increasing $P_\#$ the curve increases and the covering challenge diminishes.

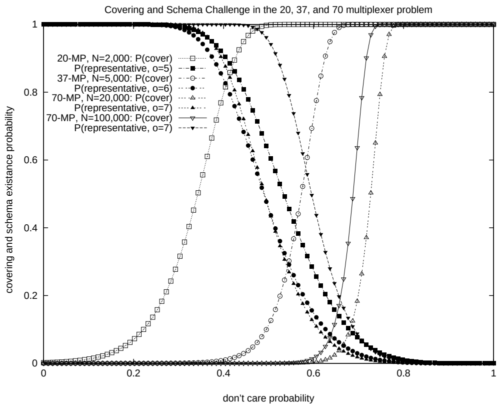

Figure 4: The two challenges visualized for the 20, 37, and 70 multiplexer problem.

Similarly, the schema challenge is observable in figure 3 although not quite as strong as the covering challenge. When $P_\#$ comes closer to one, the population is initially over-general and it is consequently more difficult to evolve accurate classifiers. Note that the schema challenge appears to make life harder for XCS until $P_\# = 0.6$ which shows that the drawback is not simply due to a further distance from the accurate, maximally general classifiers which have a generality of 0.75 in the 20-multiplexer. Figure 4 shows a close match between schema challenge curve and the results in the 20 multiplexer.

4.2 LAYERED PAYOFF BENEFIT

As visualized in figure 4 the window that allows a solution to the problem severely diminishes in the 37multiplexer. However, XCS is still able to find a solution as visualized in figure 5.

Moreover, figure 5 shows a strongly increased accuracy pressure once the reward function is changed resulting in a layered payoff landscape. The biased reward function uses the formula (value of the k position bits+return value) $* 100 +$ (correctness) $* 300$ as used in Wilson (1995). The consequence of the formula is that any specification of one of the relevant position bits, results in a decrease of the number of different rewards and consequently to a decrease in the prediction error. This decrease results in an increase of selecting classifiers which are closer to accuracy. The strong benefit of this increase is observable in figure 5. While XCS in the multiplexer with 1000/0 payoff reaches a 100% knowledge after about $500,000$ problems, XCS in the 37 multiplexer with the biased reward function actually reaches a 100% knowledge after about $100,000$ problems. Note also the differences when al-

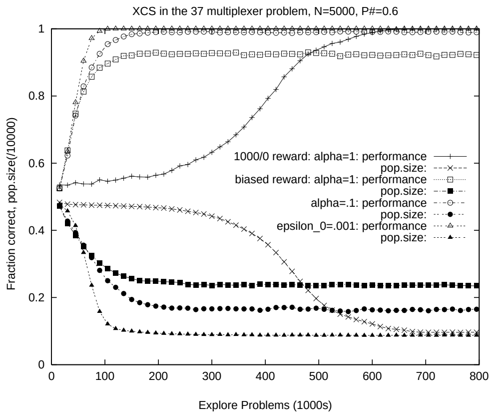

Figure 5: The 37 multiplexer problem is solvable for an appropriate parameter setting. When introducing a biased reward function, epsilon needs to be lowered.

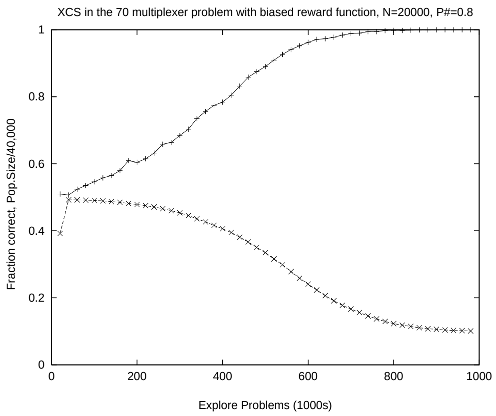

Figure 6: XCS is able to solve the 70 multiplexer problem if a biased reward function is provided.

tering the parameters α and ϵ_0 in the calculation of κ. The stronger difference when decreasing α and ϵ_0 enables XCS to distinguish the different reward levels stronger. Thus, it evolves an accurate performance faster and stays more reliably on a 100% performance level.

The layered payoff benefit even enables XCS to solve the 70multiplexer problem as shown in figure 6. So far, we didn't succeed to solve the 70multiplexer problem without layered payoff. Although the two challenges decrease the space for possible solutions as shown in figure 4 it seems theoretically possible for XCS to find a solution if a sufficiently large population is provided. However, since the first environmental niches can only be found with low chance, the solution of the 70 multiplexer without layered payoff appears to be hard.

4.3 BIASED GENERALITY BENEFIT

After we revealed the possible benefit due to a biased reward function, what remains is to show the benefit due to a change in the correct prediction when getting closer to accuracy. In order to uncover this benefit, we decided to modify the multiplexer function resulting in what we call a *xy-biased multiplexer function.* This function forms a hierarchy of depth two. The first x-position bits refer to one of the 2^x biased multiplexers located in the remaining $l - x$ bits. A biased multiplexer is defined for all $l = y + 2^y - 1$ since the biased multiplexer is always one if all y position bits are one or always zero if all its y position bits are zero dependent on if the value of the first x position bits is bigger or smaller than $(2^x - 1)/2$, respectively. The result is a biased consistency on several generality levels in over-general classifiers.

Figure 7 shows that XCS can benefit from such a bias. Shown are runs with $x = 1$ and $y = 3$, $x = 2$ and $y = 2$, and $x = 3$ and $y = 1$ which we refer to as $1,3$, $2,2$ and $3,1$ respectively. The don't care probability is set to 0.95 to assure that there are no accurate classifiers in the beginning of a run. Having a look at the previously proposed difficulty measure in Kovacs and Kerber (2001) (i.e. the size of a population that covers all environmental niches accurately, is non-overlapping, and minimal, denoted by $|[O]|$) we can observe that $|[O]|(3,1) = 48$, $|[O]|(2,2) = 56$ and $|[O]|(1,3) = 60$. The figure confirms that this is indeed a crucial measure. Moreover, we can observe that the actual problem length l seems not to have a broad impact on performance, since $l(3,1) = 19$, $l(2,2) = 22$, and $l(1,3) = 21$. However, why the performance in all three cases is much better than the performance in the 20 multiplexer where $|[O]| = 64$ and $l = 20$ is not explainable by either measure. Also the fact that the $2,2$ problem appears to be similarly difficult to the $3,1$ problem is not explainable. Finally, the plateau in the $3,1$ curve is not explainable, either. All three peculiarities however show that XCS benefits from the percentage bias. Despite the low specificity in the beginning of all runs, XCS is able to evolve the necessary specializations fast. Moreover, despite the higher $|[O]|$ measure in the $2,2$ run, XCS is able to further benefit from the percentage bias in this setting and consequently reaches a 100% knowledge as fast as in the $3,1$ run. Finally, the plateau in the $3,1$ run reveals that XCS first discovers the necessary specificity of the first bit which results, if specified in a 0.75% correctness of a classifier. However, due to the minor bias in the remaining four to be specified bits, it takes longer to proceed to a 100% knowledge. The $2,2$ case has a lower bias in the specification of the first bit, but

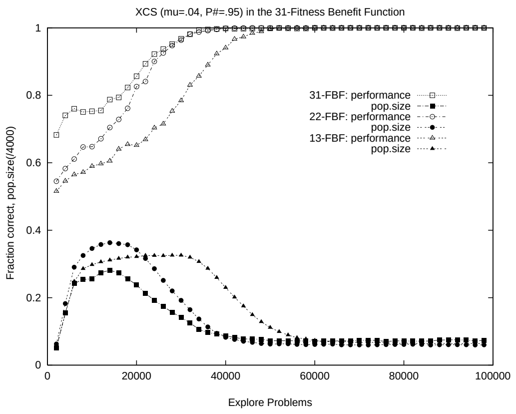

Figure 7: The increasing consistency in classification when getting closer to being accurate helps XCS in evolving a complete and accurate representation.

a stronger one, in the remaining specifications. Thus, in this case it takes longer to reliably evolve the first specifications but faster to specify the rest.

5 SUMMARY AND CONCLUSION

This paper investigated how XCS evolves accurate classifiers. We showed that the accuracy based fitness approach in XCS together with the generality pressure causes an evolutionary pressure towards accurate, maximally general classifiers. However, several problems were encountered that can prevent XCS from finding a complete and accurate solution to a given problem. The *covering challenge* needs to be met in order to prevent XCS from getting stuck in a continuous covering - deleting loop without any sort of evolution. The *schema challenge* faces XCS with the problem of evolving classifiers from the over-general side. Two environmental properties proved to help XCS in this endeavor. (1) *Layered payoff benefit* can be encountered if specialization of relevant bits leads to an on average smaller difference in payoff encountered. (2) *Biased generality benefit* is the result of a bias in the consistency of a correct/wrong classification in over-general classifiers. Moreover, it has been shown that the accuracy function is crucial in exploiting the benefits and evolving a complete and accurate solution to a problem.

Although most of the investigations herein appear to be rather theoretical, we hope that our approach leads to a broader understanding of XCS. Moreover, the two challenges together with the formulas provide rules of thumb how to set several initial parameters in XCS. Finally, the paper provides several insights how to include background knowledge in the system. By introducing a bias in the reward function, XCS can get hints that lead the system to a certain direction. Future research will show to what extent this characteristic is exploitable in the system.

References

Butz, M. V., & Wilson, S. W. (2001). An algorithmic description of XCS. *In Lanzi, P. L., Stolzmann, W. and Wilson, S. W. (Eds.), Advances in Learning Classifier Systems, LNAI 1996*. to appear.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, Massachusetts: Addison-Wesley.

Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A., & Hayes-Roth, F. (Eds.), *Pattern Directed Inference Systems* (pp. 313–329). New York: Academic Press.

Kovacs, T. (1997). XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In Roy, Chawdhry, & Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 59–68). Springer-Verlag, London.

Kovacs, T. (1999). Deletion schemes for classifier systems. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)* (pp. 329–336). San Francisco, CA: Morgan Kaufmann.

Kovacs, T. (2001). Towards a theory of strong overgeneral classifiers. *In Fogarty, T. C., Martin, W. and Spears, W. M. (Eds.), Proceedings of the Workshop on Foundations of Genetic Algorithms (FOGA2000)*. to appear.

Kovacs, T., & Kerber, M. (2001). What makes a problem hard for XCS? *In Lanzi, P. L., Stolzmann, W. and Wilson, S. W. (Eds.), Advances in Learning Classifier Systems, LNAI 1996*. to appear.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, *2*(1), 1–18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, *3*(2), 149–175.