Towards Building Block Propagation in XCS: A Negative Result and Its Implications

Kurian K. Tharakunnel, Martin V. Butz, and David E. Goldberg

Illinois Genetic Algorithms Laboratory (IlliGAL)
University of Illinois at Urbana-Champaign
104 S. Mathews, 61801 Urbana, IL, USA
{kurian,butz,deg}@illigal.ge.uiuc.edu

Abstract. The accuracy-based classifier system XCS is currently the most successful learning classifier system. Several recent studies showed that XCS can produce machine-learning competitive results. Nonetheless, until now the evolutionary mechanisms in XCS remained somewhat ill-understood. This study investigates the selectorecombinative capabilities of the current XCS system. We reveal the accuracy dependence of XCS's evolutionary algorithm and identify a fundamental limitation of the accuracy-based fitness approach in certain problems. Implications and future research directions conclude the paper.

1 Introduction

After Holland's introduction of *learning classifier systems* (LCSs), originally referring to a *cognitive system* [7], one of the most important steps in LCS research was the development of the accuracy-based classifier system XCS by Wilson [12]. Most of the recent work on LCSs focuses on XCS. Among the many changes from the original LCS, the accuracy-based fitness in XCS is considered as the distinctive feature and the most important reason for success.

Although crossover has always been applied in XCS, our experimental investigations showed that in most investigated classification problems to date mutation alone seems to be sufficient to evolve an appropriate solution. However, the larger a problem the more difficult it becomes to generate an accurate solution by mutation. This is the case since mutation in general results in a diversification pressure and in LCSs effectively in a specialization pressure since classifiers usually specify only a small fraction of the available features [2]. The larger the problem, however, the more general classifiers need to be to undergo sufficient evaluations and reproductive events. Thus, the larger a problem the smaller mutation needs to be. Consequently, only successful recombinations of accurate sub-structures, or *building blocks* [6], can do the trick. Additionally, as argued in [5], suitable search and ultimately innovative events are determined by effective recombination of appropriate sub-parts. As this insight led to the development of *competent GAs*—GAs that solve boundedly difficult problems quickly, accurately, and reliably—the same approach seems necessary in learning classifier system research to develop competent LCSs, that is, LCSs that solve typical (decomposable) machine learning problems efficiently.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1906–1917, 2003.

Recently, several studies addressed how XCS works. Many of these try to explain the XCS mechanisms and help choosing appropriate parameter settings. An important aspect missing so far is an explanation for the apparent non-performance of XCS in some problems. This paper investigates a set of such hard problems. By doing that, we reveal an important interaction between the accuracy based fitness and the selectorecombinative GA [6,4] in XCS that prevents learning. Previous early hints on this interaction can be found in [1].

The role of the GA in XCS is to evolve accurate classifiers as well as to search for optimal generalizations within classification niches. The basic working of GA dictates that selection, recombination, and mutation progressively results in the discovery of better individuals. We show that the GA in XCS may not be able to achieve this in particular problems due to a lack of suitable accuracy guidance from the over-general side. We use concatenated multiplexer problems (multiplexer problems formed by concatenating the condition and action parts of more than one single multiplexer problem) to illustrate how XCS performance is affected by the combination of accuracy based fitness and selectorecombinative GAs.

After a brief review on previous analyses of XCS mechanisms, we formalize the derivation of the estimated prediction error in XCS. Since the accuracy is derived from the prediction error estimate, this estimate is crucial for XCS success. In Sect. 4 we identify a fundamental weakness in XCS's accuracy approach, illustrating successful and unsuccessful selectorecombinative events in XCS. Concluding remarks complete the paper.

2 XCS Performance

Along with the introduction of XCS, Wilson [12] illustrated the success of XCS both in single-step problems and multi-step problems with results from experiments on Multiplexer and Woods2 environments respectively. Later Wilson showed the successful performance of XCS in a standard data mining task [13]. However, the results obtained in several other test environments were not very encouraging. An interesting comparison of the performance of XCS with other LCS models in various Maze environments can be found on the web page: http://www.ai.tsi.lv/ga/lcs_performance.html

Although considerably more tractable than Holland's original LCS model, XCS is still a complex system. There have been several attempts recently to understand XCS mechanisms and their interactions. The studies by Kovacs [8] established that XCS would be able to accurately build an optimal representation of the payoff function of a problem. Kovacs also illustrated how XCS overcomes the problem of strong over-generals encountered in strength based LCSs without fitness sharing [9]. A fundamental problem difficulty dimension was characterized by Kovacs' optimal rule set size measure [O] [10]. Recently, Butz et al. [1,2] have given a more analytical treatment of XCS mechanisms, in which further dimensions of problem complexity were identified.

3 Accuracy Revisited

The most important and distinctive feature of XCS is its fitness definition. In the traditional LCS approach payoff prediction of the classifier, sometimes combined with condition specificity, is directly used as the fitness measure. In XCS, the accuracy with which a classifier predicts payoff is used as a measure of its fitness.

The actual procedure of fitness calculation involves several steps [3]. Every time a classifier takes part in an action set, first its prediction parameter is estimated using the immediate payoff received according to the standard Widrow-Hoff delta rule. Next, an estimate of the absolute error in estimate of the prediction is done using the current error (absolute difference between the immediate payoff and the current estimate of prediction). This is again done by the Widrow-Hoff delta rule. This error estimate is then used to calculate the accuracy of a classifier by using a power function on error. This ensures that classifiers with error less than a threshold value will have accuracy one and the accuracy falls steep with increase in error. Next, the accuracy values (all between zero and one) of all the classifiers in the action set are used to calculate a relative accuracy value for each classifier in the action set. These relative accuracy values are then used to update the fitness of classifiers using the Widrow-Hoff delta rule once again.

The procedure described above shows that the estimated absolute error plays a crucial role in the determination of accuracy of a classifier. To understand the full significance of this we should know what the estimates of prediction and prediction error mean to a classifier. A classifier represents a set of environmental states along with a possible action. Thus, a classifier in fact represents a set of state-action pairs for the given problem. Each state-action pair results in a particular reward (state-action value). Hence, as a classifier takes part in many action sets and with the update procedure described above, the prediction parameter of the classifier is estimated towards the average of the state-action values of the state-action pairs covered by the classifier. Similarly, the update procedure on prediction error results in an estimate of the mean absolute deviation (MAD) in the covered state-action pairs. Thus, the error estimate of a classifier in XCS is in fact an estimate of MAD of state-action values associated with that classifier. We call this value simply the MAD of the classifier.

The above definition of error can be used to calculate directly the error of a classifier and hence determine its accuracy. For example, let p_s denote the probability of correct classification with a uniform reward of r and zero reward otherwise. Then the prediction P of a classifier may be written as

$$P = rp_s + 0 * (1 - p_s) = rp_s \tag{1}$$

Similarly, the error ϵ which is the MAD of the classifier may be written as

$$\epsilon = |r - P|p_s + |0 - P|(1 - p_s) = 2r(p_s - p_s^2) \tag{2}$$

(see also [1]). We use these expressions to determine reward prediction and reward prediction error of classifiers in the following sections. Figure 1 illustrates

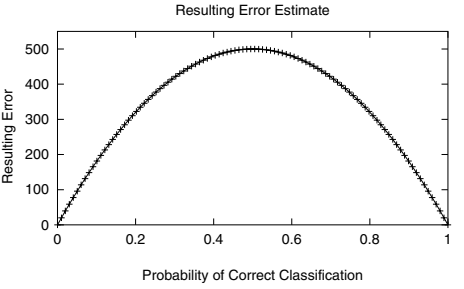


Fig. 1. Prediction-error dependence on correctness of classifier in a 1000/0 reward scheme

Equation 2. It can be seen that the more consistently a classifier classifies correctly (or incorrectly), the smaller its prediction error and thus the larger its accuracy will be.

4 Accuracy Guidance for Successful Recombination

Two fundamental processes are associated with the working of every LCS - rule discovery and rule evaluation which are executed by the GA and the credit allocation components, respectively. In XCS, these two processes work towards the objective of building a complete, accurate, and maximally general payoff map. That is, a problem representation is formed that is able to predict the resulting payoff of an action, or classification, for any possible problem instance. Important changes in XCS are the accuracy-based fitness approach and GA selection which is applied in action set niches rather than panmictically in the whole population as in traditional LCSs. Deletion, however, is done panmictically in the whole population. This results in an inherent generalization pressure as hypothesized in [12] and formulated in [2].

The question is now how the GA evolves accurate classifiers in XCS. The GA used in XCS is basically selectorecombinative. This means that the driving force of evolution towards better individuals is the dual process of selection and recombination with a continuous influence of mutation. With this point of view, accurate classifiers are evolved from selection, mutation, and recombination of good partially accurate solutions. Especially successful recombinations of different partially accurate sub-parts, or *building blocks*, should lead to higher accuracy.

4.1 Accuracy Guidance Required

Let's define an accurate, maximally general classifier as a *completely successful classifier* and the corresponding generalization a *completely successful generalization*. A perfect classifier has maximum accuracy so that any further generaliza-

tion of the classifier condition will result in a reduction in accuracy. Such a classifier can be characterized by the specified positions in the condition part since all other positions are filled with don't care symbols. Let us say this classifier condition consists of three specified positions *a, b* and *c*. Based on the discussion above, this classifier can be generated through GA recombination or mutation. Hence we can reasonably assume that the specific pattern of *abc* can result only from recombinations or mutations of classifiers with the partial patterns *a, b* and *c* in them (since a direct generation of *abc* by mutation is highly unlikely). We term the classifiers that specify a subset of positions *abc partially successful classifiers* and the corresponding generalizations *partially successful generalizations*. Note that there might be more than one possible completely successful generalization and different completely successful classifiers might overlap. This issue, however, needs to be addressed separately from the concern in this paper.

The above argument leads to the hypothesis that successful generalization can be possible only if the following two conditions are satisfied. (1) Sustenance of partially successful classifiers in the population; (2) Partially successful classifiers whose conditions are more similar to completely successful classifiers have higher accuracy. The first condition says that the partially successful classifier should have some minimum accuracy (fitness) so that it will not get deleted from the population. To a great extent this is dependent on the parameter setting of XCS as well as on the problem properties. The second condition says that classifiers with conditions closer to a completely successful classifier should have higher fitness so that a proper recombination/mutation can lead to the generation of completely successful classifiers. As we investigate in the rest of this paper, this *accuracy guidance* is essential for successful evolutionary learning in XCS.

Note that we focus on when a successful recombination, i.e. a recombination that generates offspring that is more similar to a perfect classifier in its condition part than its parents, actually also results in higher accuracy. In this paper, we are not interested in the matter of a good crossover operator or even in the creation of a competent crossover operator (such as the probabilistic model building GAs field (PMBGAs) [11]), but in the satisfaction of the preconditions for such successful recombination.

4.2 Accuracy Guidance in the Six Multiplexer

To illustrate accuracy guidance further we take the Boolean multiplexer problem as an example.

Multiplexer problems are widely used test problems in LCS research. In a multiplexer problem, the agent tries to predict the output of a multiplexer function. A multiplexer function is a function defined on binary strings of length $k + 2^k$. The first k bits reference the output bit in the 2^k remaining bits. In a 3 multiplexer problem, for example, a zero in the first bit would determine the output being the second bit while a one would reference the third bit.

In the 6 multiplexer problem the input strings are of length six and the first two bits determine the output bit position. According to our definition of a completely successful classifier, it can easily be observed that one of the completely

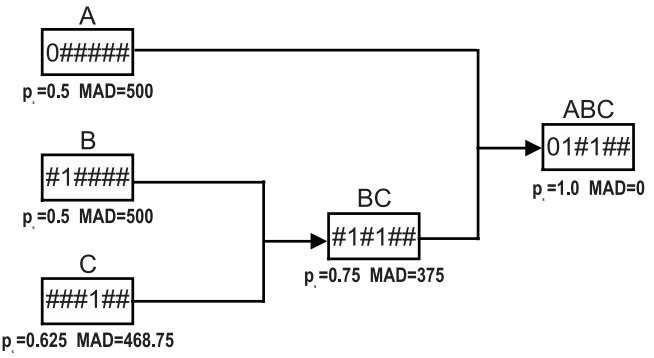


Fig. 2. Progress of successful generalization in 6 multiplexer problem

successful classifier in a 6 multiplexer problem is 01#1## → 1. Now consider that the condition part of this classifier consists of three consecutive patterns *a*, *b* and *c* involving the three specified bits respectively. Then a partially successful classifier with pattern *a* is 0##### → 1. Similarly partially successful classifiers with patterns *b* and *c* are #1#### → 1 and ###1## → 1 respectively. Here we have considered the most general version of partially successful classifiers though we can consider partially successful classifiers with more specific bits with the same effect. Now let us look at what happens to the accuracy when the partially successful classifiers combine.

Figure 2 illustrates the progressive recombination of partially successful classifiers A, B and C resulting in the completely successful classifier ABC. A 1000/0 payoff scheme is used in this example and the accuracy of classifiers is measured as MAD. It can be seen that the MAD diminishes when partially successful classifiers recombine. Eventually, the completely successful classifier with MAD 0 will be generated. Here the condition (2) of our hypothesis holds in that a successful recombination of partially successful classifiers also results in a decrease in MAD and thus in an increase in accuracy. This explains how XCS with the accuracy based fitness is able to successfully solve the 6 multiplexer problem. What happens when condition (2) is not satisfied? This is illustrated in the next section using what we call a *concatenated multiplexer problem*.

4.3 Concatenated Multiplexer Problems

We construct concatenated multiplexer problems by combining more than one multiplexer problem. In a concatenated multiplexer problem, the agent tries to predict the output of a concatenated multiplexer function. A concatenated multiplexer function is the function obtained by concatenating the input part and the output part of more than one individual multiplexer functions of the same type. The output will be the combination of outputs determined by the individual multiplexer functions. For example, Table 1 shows a sample of inputs and outputs of a concatenated multiplexer function made of three 3-multiplexer

Table 1. A sample input/output mapping of *3X3* multiplexer problem

Input	Output
000 000 000	000
010 100 111	101
110 101 001	010
111 110 001	100
111 111 111	111

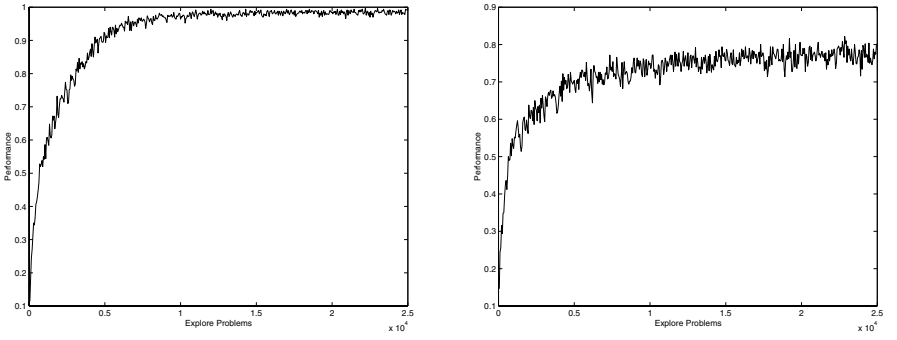
Table 2. In the *3X3* multiplexer problem the necessary classifiers for an accurate and correct classification need to be much more specific than those for an accurate but wrong classification (classifiers that always suggest a wrong classification)

Correct Classifiers		Wrong Classifiers	
Condition	Action	Condition	Action
00# 00# 00#	000	### ### 01#	000
00# 00# 1#0	000	### ### 1#1	000
00# 1#0 00#	000	### 01# ###	000
00# 1#0 1#0	000	### 1#1 ###	000
1#0 00# 00#	000	01# ### ###	000
1#0 00# 1#0	000	1#1 ### ###	000
1#0 1#0 00#	000	### ### 00#	001
1#0 1#0 1#0	000	...	...
00# 00# 01#	001	...	...
...	...	...	...

functions. We call this concatenated problem a *3X3* multiplexer problem. In general, a *mXn* multiplexer problem is a combination of m problems of type n.

We use concatenated multiplexer problems to illustrate the hypothesis of Sect. 4.1 because we are sure of the completely successful generalizations existing in these problems. For example, in the *3X3* problem, we know the completely successful generalizations for the individual three multiplexer problems and so the completely successful generalization for the concatenated problem can be easily constructed. A part of the optimal classifier population $[O]$ for the *3X3* problem is shown in Table 2. Note that for the wrong classifiers the wrong output of one component is sufficient for the entire problem to have a wrong output and so a greater level of generalization is possible. The size of the optimal population $|[O]|$ in an *mXn* problem where the n-multiplexer has n_p position bits can be calculated by $|[O]| = 2^m(2^{n_p^m}) + 2^m m 2_p^n = 2^{2m+n_p m} + m2^{m+n_p}$ adding the number of correct and wrong classifiers together. In the *3X3* problem $|[O]| = 64 + 48 = 112$.

It is surprising that although an optimal population exists for the *3X3* problem, XCS fails to evolve it. We show that this is because condition (2) of our hypothesis in Sect. 4.1 is not satisfied in this problem.



(a) 1000/0 Reward (b) Layered Reward

Fig. 3. XCS performance in the *3X3* multiplexer problem

Before we proceed further with concatenated multiplexer problems we need to decide about the reward function to be used in these problems. We can use the usual 1000/0 reward structure which means that a correct output from all the component multiplexer problems will result in a reward of 1000 but if one of them is wrong then the output is 0. Another possibility is to use a layered reward structure, i.e. a progressive reward of 1000 with each component multiplexer giving the correct output. This will result in a reward of $1000c$ for the concatenated problem where c is the number of components correctly classified by the specified action. We use the 1000/0 reward function here. Later we will show that XCS performance will be worse with layered reward as the first condition (sustenance of partially successful classifiers) is violated as well in that case.

Next, we shall see why XCS is not able to perform well in the *3X3* problem. Remember that the three multiplexer is a rather trivial problem for XCS. There are only two actions and XCS will quickly find the accurate and optimal classifier population. Now, in the *3X3* problem, there are eight actions, two from each component problem. The XCS performance on the *3X3* with 1000/0 reward structure is shown in Fig. 3(a). (All the performance results shown in this paper are averages of 10 runs with parameters set as follows: $N = 2000,\ \beta = 0.2,\ \alpha = 0.1,\ \epsilon_0 = 0.01,\ \nu = 5,\ \theta_{GA} = 25,\ \chi = 0.8,\ \mu = 0.04,\ \theta_{del} = 20,\ \delta = 0.1, \theta_{sub} = 20,\ P_{\#} = 0.33,\ p_{explr} = 1,\ p_I = -10,\ \epsilon_I = 0,\ and\ F_I = 0.01$ [3]). It can be seen that XCS fails to reach 100% performance even after 25,000 steps.

To see why XCS is not able to perform well in this problem, let us look at the completely successful classifiers for this problem. For example, 01# 01# 01# $\rightarrow$ 111 is an accurate and optimal generalization (completely successful classifier) for the problem (the spaces between condition part of component multiplexers are for clarity). One possible way for this completely successful classifier to form in the population is by the recombination of the partially successful classifiers :

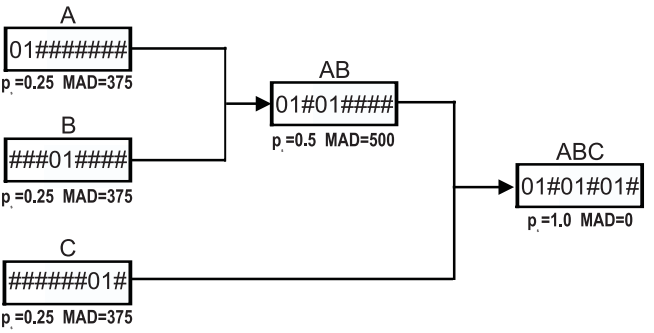


Fig. 4. Necessary learning progress in *3X3* multiplexer problem

01# ### ### → 111, ### 01# ### → 111 and ### ### 01# → 111. We have shown only the most general case of partially successful classifiers though in reality more specific forms are likely. However, this will not change the results we are going to illustrate. Also, remember that the GA occurs only in the action set specified by the action 111 and these recombinations happen progressively. In Fig. 4, a possible progressive formation of the completely successful classifier from partially successful classifiers is shown. The accuracy of classifiers are expressed as MAD. Note that the MAD of the completely successful classifier is 0.

In particular, let's look at the exact derivation of those numbers. Classifiers cl_1 = 01# ### ### → 111 and cl_2 = ### 01# ### → 111 classify a random input string correctly with a probability of 0.25 since there is a 50/50 chance that either of the other components is correct. Thus, the reward predictions of these classifiers will be approximately $0.25 \cdot 1000 = 250$ and consequently their reward prediction errors (MAD) will approximate 375 (see equations 1 and 2). The more specific classifier cl_3 = 01# 01# ### → 111 is much closer to the desired completely successful classifier cl_4 = 01# 01# 01# → 111. Thus, cl_3 should have higher fitness than cl_1 and cl_2 since a less complex recombinatory event or the mutation of the appropriate two attributes can actually lead to the generation of classifier cl_4. Classifier cl_3 has a probability of 0.5 to perform the correct classification. Thus, both its reward prediction and MAD will approximate 500. With a higher MAD, classifier cl_3 will actually have a smaller accuracy than classifiers cl_1 and cl_2. Thus, cl_3 will encounter less recombinatory events and is consequently highly likely to be deleted soon. Moreover, in order to get to classifiers cl_1 and cl_2 it is necessary that those classifiers have a lower MAD than, for example, the completely general classifier (classifier with only don't care symbols in the condition part). However, the probability of a completely general classifier to classify a random input correctly is 0.125 which results in a MAD of 218.75 so that the completely general classifier has an even higher accuracy than cl_1 or cl_2. Thus, besides the inherent generalization pressure in XCS [12,2], in this problem, fitness also pushes towards generality instead of towards

specificity so that it is highly unlikely that XCS will ever evolve an accurate representation of the problem.

Indeed, looking at the evolving population in more detail revealed that none of the completely successful classifiers are present in the population. It seems somewhat surprising, though, why XCS still reaches a performance of near 100% as displayed in Fig. 3(a). A more detailed analysis revealed that XCS is actually learning the classifiers that specify inaccurate classifications accurately (shown on the right column in table 2). Thus, given a current match set, XCS eventually 'knows' that seven of the eight possible classifications will be incorrect and thus chooses the correct classification most of the time.

In Fig. 3(b), we show performance of XCS in the *3X3* multiplexer problem when the reward is $1000c$ where c is the number of components correctly classified by the specified action. It can be observed that performance is worse with this layered reward structure. This is because, with layered reward, the difference between MAD of partially successful classifiers resulting from recombination/mutation and MAD of their parent classifiers is even larger. This results in a smaller chance for the newly generated partially successful classifiers to sustain in the population and cause subsequent progress of successful generalizations. This shows that layered reward does not necessarily need to be helpful.

The concatenated multiplexer problem is essentially a problem in which the base probability of classifying an input correctly is below 0.5. Looking back at Fig. 1 we see that the resulting problem is that XCS needs to cope with the problem that more correct classifiers (those closer to 1.0 and thus initially closer to 0.5) are actually less accurate due to the MAD-based accuracy. Moreover, the problem is not symmetrical in that an accurate, maximally general classifier that predicts payoff 0 does not have the same structure as an accurate, maximally general classifier that predicts payoff 1000. If there are other problem properties that can cause the same problem or if these properties are typical in interesting problems is a subject of future research.

Note that our analysis made several simplifying assumptions. First, we assumed that reward prediction and reward prediction error of a classifier immediately take on the average values. However, the Widrow-Hoff delta rule allows only an approximation of these values with an accuracy dependent on the update rate β. Second, since classifier fitness in XCS is based on the set-relative accuracy of a classifier, overlapping classifiers (classifiers that match sometimes both and sometimes either one of them) can alter fitness determination which might cause additional problems. Third, we studied XCS in one-step, or classification, problems. In multi-step problems the additional problem of reinforcement propagation arises which can cause over-general classifiers to propagate inappropriate reinforcement values. Thus, in multi-step problems, the determination of current MAD is even harder.

5 Conclusion

This paper investigated the accuracy-based fitness approach in XCS and its implications for a successful selectorecombinative evolutionary process. Accuracy

of a classifier was derived from its mean absolute deviation (MAD) which is approximated by its reward prediction error estimate. Two necessary conditions were identified for a successful evolutionary process: (1) Sustenance of partially successful classifiers; (2) classifiers closer to completely successful ones need to be more accurate. Violating either of those conditions disables the evolutionary process from evolving an accurate problem representation since there are no classifiers to propagate and/or the propagation leads towards the wrong direction. The concatenated multiplexer problem was introduced as an example problem that violates the second condition. Consequently, XCS was not able to evolve the desired complete, accurate, and maximally general payoff map of the problem.

As the concatenated multiplexer problem showed, there are problems that are inherently accuracy-misleading for the current accuracy-based fitness approach in XCS. Thus, a modification of the accuracy definition appears to be necessary to enable successful learning. Since accuracy is derived from the reward prediction error which approximates the MAD, it appears that the MAD itself is not an appropriate basis for XCS's accuracy measure. Thus, it is necessary to redefine the reward prediction error in XCS. One approach would be to change the error to an upper and lower error or a two-sided error measure. Future research needs to investigate the various possibilities for modifying the reward prediction error in XCS.

Although a sufficiently large population size with a large mutation might be able to create and maintain completely successful classifiers in our investigated concatenated multiplexer problem, mutation cannot remedy the problem in general since it does not scale up. Thus, accuracy guidance needs to be exploited for successful selectorecombinative events. Once accuracy guidance is available in a problem, the next step is to investigate and develop competent crossover operators that can further speed-up and scale-up evolutionary search in XCS.

Acknowledgments. We are grateful to Xavier Llora for the useful discussions. This work was sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by the National Science Foundation under grant DMI-9908252. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.W.: How XCS evolves accurate classifiers. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 927–934

2. Butz, M.V., Pelikan, M.: Analyzing the evolutionary pressures in XCS. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 935–942
3. Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. In Lanzi, P.L., Stoltzman, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000, Berlin Heidelberg, Springer-Verlag (2001) 253–272
4. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, MA (1989)
5. Goldberg, D.E.: The design of innovation: Lessons from and for competent genetic algorithms. Kluwer Academic Publishers, Boston, MA (2002)
6. Holland, J.H.: Adaptation in natural and artificial systems. Universtiy of Michigan Press, Ann Arbor, MI (1975) second edition 1992.
7. Holland, J.H.: Adaptation. In Rosen, R., Snell, F., eds.: Progress in Theoretical Biology. Volume 4., New York, Academic Press (1976) 263–293
8. Kovacs, T.: 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. Springer-Verlag, London (1997) 59–68
9. Kovacs, T.: Towards a theory of strong overgeneral classifiers. Foundations of Genetic Algorithms 6 (2001)
10. Kovacs, T., Kerber, M.: What makes a problem hard for XCS? In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000. Springer-Verlag, Berlin Heidelberg (2001) 80–99
11. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey on optimization by building and using probabilistic models. Computational Optimization and Applications **21** (2002) 5–20
12. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
13. Wilson, S.W.: Mining oblique data with XCS. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000. Springer-Verlag, Berlin Heidelberg (2001) 158–174