Tournament Selection: Stable Fitness Pressure in XCS

Martin V. Butz[1], Kumara Sastry[1], and David E. Goldberg[1]

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

Abstract. Although it is known from GA literature that proportionate selection is subject to many pitfalls, the LCS community somewhat adhered to proportionate selection. Also in the accuracy-based learning classifier system XCS, introduced by Wilson in 1995, proportionate selection is used. This paper identifies problem properties in which performance of proportionate selection is impaired. Consequently, tournament selection is introduced which makes XCS more parameter independent, noise independent, and more efficient in exploiting fitness guidance.

1 Introduction

Learning Classifier Systems (LCSs) [1, 2] are rule learning systems in which rules are generated by the means of a genetic algorithm (GA) [3]. The GA evolves a population of rules, the so called *classifiers.* A classifier usually consists of a condition and an action part. The condition part specifies when the classifier is applicable and the action part specifies which action to execute. In difference to the original LCSs, the fitness in the XCS classifier system, introduced by Wilson [4], is based on the accuracy of reward predictions rather than on the reward predictions directly. Thus, XCS is meant to not only evolve a representation of an optimal behavioral strategy, or classification, but rather to evolve a representation of a complete *payoff map* of the problem. That is, XCS is designed to evolve a representation of the expected payoff in each possible situation-action combination. Recently, several studies were reported that show that XCS performs comparable to several other typical classification algorithms in different standard machine learning problems [5–7].

Although many indicators can be found in the GA literature that pointout that proportionate selection is strongly fitness dependent [8] and moreover is strongly dependent on the degree of convergence [9], the LCS community somewhat ignored this insight. In XCS, fitness is a scaled estimate of relative accuracy of a classifier. Due to the scaling, it does not seem necessary to apply a more fitness-independent selection method. Nonetheless, in this paper we identify problem types that impair the efficiency of proportionate selection. We show that tournament selection makes XCS selection more reliable outperforming proportionate selection in all investigated problem types.

We restrict our analysis to Boolean classification, or one-step, problems in which each presented situation is independent of the history of situations and classifications. Particularly, we apply XCS to several typical Boolean function problems. In these problems, feedback is available immediately so that no reinforcement learning techniques are necessary that propagate reward [10].

The aim of this study is threefold. First, to make the XCS classifier system more robust and more problem independent. Second, to contribute to the understanding of the functioning of XCS in general. Third, to prepare the XCS classifier system to solve decomposable machine learning problems quickly, accurately, and reliably.

The next section gives a short overview of the major mechanisms in XCS. Next, the Boolean multiplexer problem is introduced and noise is added which causes XCS to fail. Next, we introduce tournament selection to XCS. Section 5 provides further experimental comparisons of the two mechanisms in several Boolean function problems with and without additional noise. Finally, we provide summary and conclusions.

2 XCS in a Nutshell

Although XCS was also successfully applied in multi-step problems [4, 11–13], we restrict this study to classification problems to avoid the additional problem of reward propagation. However, the insights of this study should readily carry over to multi-step problems. This section consequently introduces XCS as a pure classification system providing the necessary details to comprehend the remainder of this work. For a more complete introduction to XCS the interested reader is referred to the original paper [4] and the algorithmic description [14].

We define a classification problem as a problem that consists of problem instances $s \in \mathcal{S}$ that need to be classified by XCS with one of the possible classifications $a \in \mathcal{A}$. The problem then provides scalar payoff $R \in \Re$ with respect to the made classification. The goal for XCS is to choose the classification that results in the highest payoff. To do that, XCS is designed to learn a complete mapping from any possible $s \times a$ combination to an accurate payoff value. To keep things simple, we investigate problems with Boolean input and classification, i.e. $\mathcal{S} \subseteq \{0,1\}^L$ where L denotes the fixed length of the input string and $\mathcal{A} = \{0,1\}$.

XCS evolves a population $[P]$ of rules, or *classifiers*. Each classifier in XCS consists of five main components. The condition $C \in \{0,1,\#\}^L$ specifies the subspace of the problem instances in which the classifier is applicable, or *matches*. The "don't care" symbol # matches in all input cases. The action part $A \in \mathcal{A}$ specifies the advocated action, or classification. The payoff prediction p approaches the average payoff encountered after executing action A in situations in which condition C matches. The prediction error ε estimates the average deviation, or error, of the payoff prediction p. The fitness reflects the average relative accuracy of the classifier with respect to other overlapping classifiers.

XCS iteratively updates its knowledge base with respect to each problem instance. Given current input s, XCS forms a *match set* $[M]$ consisting of all classifiers in $[P]$ whose conditions match s. If an action is not represented in $[M]$, a covering classifier is created that matches s (#-symbols are inserted with a probability of $P_{\#}$ at each position). For each classification, XCS forms a *payoff prediction* $P(a)$, i.e. the fitness-weighted average of all reward prediction estimates of the classifiers in $[M]$ that advocate classification a. The payoff predictions determine the appropriate classification. After the classification is selected and sent to the problem, payoff R is provided according to which XCS updates all classifiers in the current action set $[A]$ which comprises all classifiers in $[M]$ that advocate the chosen classification a. After update and possible GA invocation, the next iteration starts.

Prediction and prediction error parameters are update in $[A]$ by $p \leftarrow p + \beta(R - p)$ and $\varepsilon \leftarrow \varepsilon + \beta(|R - p| - \varepsilon)$ where β ($\beta \in [0, 1]$) denotes the *learning rate*. The fitness value of each classifier in $[A]$ is updated according to its current scaled relative accuracy κ':

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0 \\ \alpha \left(\frac{\varepsilon_0}{\varepsilon}\right)^{\nu} & \text{otherwise} \end{cases} \qquad \kappa' = \frac{\kappa}{\sum\limits_{x \in [A]} \kappa_x} \tag{1}$$

$$F \leftarrow F + \beta(\kappa' - F) \tag{2}$$

The parameter ε_0 ($\varepsilon_0 > 0$) controls the tolerance for prediction error ε; parameters α ($\alpha \in (0, 1)$) and ν ($\nu > 0$) are constants controlling the rate of decline in accuracy κ when ε_0 is exceeded. The accuracy values κ in the action set $[A]$ are then converted to set-relative accuracies κ'. Finally, classifier fitness F is updated towards the classifier's current set-relative accuracy. Figure 1 shows how κ is influenced by ε_0 and α. The determination of κ, then, also causes the scaling of the fitness function and thus strongly influences proportionate selection. All parameters except fitness F are updated using the *moyenne adaptive modifiée* technique [15]. This technique sets parameter values directly to the average of the so far encountered cases as long as the experience of a classifier is still less than $1/\beta$. Each time the parameters of a classifier are updated, the experience counter *exp* of the classifier is increased by one.

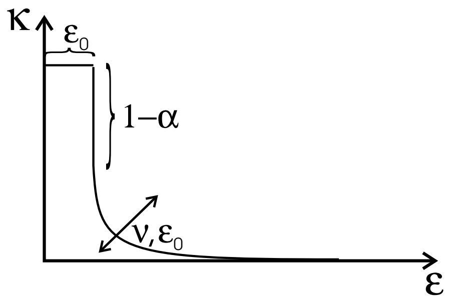

Fig. 1. The scaling of accuracy κ is crucial for successful proportionate selection. Parameters ε_0, α, and ν control tolerance, offset, and slope, respectively.

A GA is invoked in XCS if the average time since the last GA application upon the classifiers in $[A]$ exceeds threshold θ_{ga}. The GA selects two parental classifiers using proportionate selection (the probability of selecting classifier cl ($P_s(cl)$) is determined by its relative fitness in $[A]$, i.e. $P_s(cl) = F(cl)/\sum_{c\in[A]} F(c)$). Two offspring are generated reproducing the parents and applying (two-point) crossover and mutation. Parents stay in the population competing with their offspring. We apply free mutation in which each attribute of the offspring condition is mutated to the other two possibilities with equal probability. Parameters of the offspring are inherited from the parents, except for the experience counter *exp* which is set to one, the numerosity *num* which is set to one, and the fitness F which is multiplied by 0.1. In the insertion process, *subsumption deletion* may be applied [16] to stress generalization. Due to the possible strong effects of *action-set subsumption* we apply *GA subsumption* only.

The population of classifiers $[P]$ is of fixed size N. Excess classifiers are deleted from $[P]$ with probability proportional to an estimate of the size of the action sets that the classifiers occur in (stored in the additional parameter as). If the classifier is sufficiently experienced and its fitness F is significantly lower than the average fitness of classifiers in $[P]$, its deletion probability is further increased.

3 Trouble for Proportionate Selection

Proportionate selection depends on fitness relative to action set $[A]$. The fitness F of a classifier is strongly influenced by the accuracy determination of Equation 1 visualized in Figure 1. F is also influenced by the number of classifiers in the action sets the classifier participates in. Moreover, the initial fitness of offspring is decreased by 0.1 so that it may take a while until the superiority of offspring is reflected by its fitness value. To infer these dependencies, we apply XCS to the Multiplexer problem with altered parameter settings or with additional noise in the payoff function of the problem.

The multiplexer problem is a widely studied problem in LCS research [17, 4, 16, 18]. It has been shown that LCSs are superior compared to standard machine learning algorithms, such as *C4.5*, in the multiplexer task [17]. The multiplexer problem is a Boolean function. The function is defined for binary strings of length $k + 2^k$. The output of the multiplexer function is determined by the bit situated at the position referred to by the k position bits. For example, in the six multiplexer $f(100010) = 1$, $f(000111) = 0$, or $f(110101) = 1$. A correct classification results in a payoff of 1000 while an incorrect classification results in a payoff of 0.

Figure 2 reveals the strong dependence on parameter β.[1] Decreasing the learning rate hinders XCS from evolving an accurate payoff map. The problem

[1] All results herein are averaged over 50 experimental runs. Performance is assessed by test trials in which no learning takes place and the better classification is chosen as the classification. During learning, classifications are chosen at random. If not stated differently, parameters were set as follows: $N = 2000$, $\beta = 0.2$, $\alpha = 1$, $\varepsilon_0 = 1$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, and $P_{\#} = 0.6$.

is that over-general classifiers occupy a big part of the population initially. Better offspring often looses against the over-general parents since the fitness of the offspring only increases slowly (due to the low β value) and small differences in the fitness F only have small effects when using proportionate selection. Altering the slope of the accuracy curve by changing parameters α and ε_0 does not have any positive learning effect, either. Later, we show that small β values are necessary in some problems so that increasing β does not solve this problem in general. Figure 2 (right-hand side) also reveals XCS's dependence on initial specificity. Increasing $P_\#$ (effectively decreasing initial specificity) impairs learning speed of XCS facing the *schema challenge* [18].

Additional to the dependence on β we can show that XCS is often not able to solve noisy problems. We added two kinds of noise to the multiplexer problem: (1) Gaussian noise is added to the payoff provided by the environment. (2) The payoff is alternated with a certain probability, termed *alternating noise* in the remainder of this work. Figure 3 shows that when adding only a small amount of either noise, XCS's performance is strongly affected. The more noise is added, the smaller the fitness difference between accurate and inaccurate classifiers. Thus, selection pressure decreases due to proportionate selection so that the population starts to drift at random. Lanzi [12] proposed an extension to XCS that detects noise in environments and adjusts the error estimates accordingly. This approach, however, does not solve the β problem.

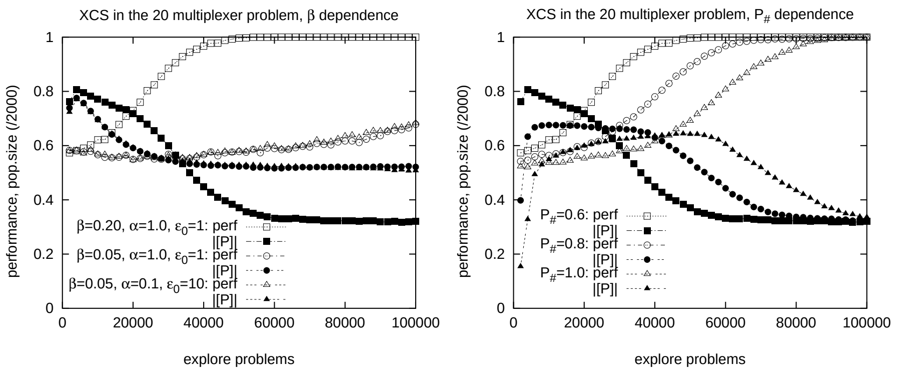

Fig. 2. Decreasing learning rate β decreases XCS learning performance (left-hand side). Accuracy function parameters have no immediate influence. Also a decrease in initial specificity strongly decreases XCS's performance (right-hand side).

4 Stable Selection Pressure with Tournament Selection

In difference to proportionate selection, tournament selection is independent of fitness scaling [8]. In tournament selection parental classifiers are not selected proportional to their fitness, but tournaments are held in which the classifier with

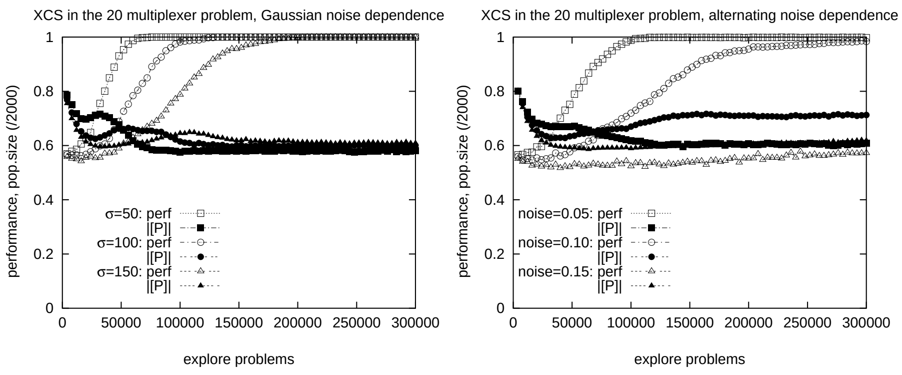

Fig. 3. Adding noise to the payoff function of the multiplexer significantly deteriorates performance of XCS. Gaussian noise (left-hand side) or alternating noise (right-hand side) is added.

the highest fitness wins (stochastic tournaments are not considered herein). Participants for the tournament are usually chosen at random from the population in which selection is applied. The size of the tournament controls the selection pressure. Usually, fixed tournament sizes are used.

In difference to standard GAs, the GA in XCS is a steady-state, niche GA. Only two classifiers are selected in each GA application. Moreover, selection is restricted to the current action set. Thus, some classifiers might not get any reproductive opportunity at all before being deleted from the population. Additionally, action set sizes can vary significantly. Initially, action sets are often over-populated with over-general classifiers. Thus, a relatively strong selection pressure appears to be necessary which adapts to the current action set size.

Our tournament selection process holds tournaments of sizes dependent on the current action set size $|[A]|$ by choosing a fraction τ of classifiers ($\tau \in (0, 1]$) in the current action set. [2] Instead of proportionate selection, two independent tournaments are held in which the classifier with the highest fitness is selected. We also experimented with fixed tournament sizes, as shown below, which did not result in a stable selection pressure.

Figure 4 shows that XCS with tournament selection, referred to as *XCSTS* in the remainder of this work, can solve the 20-Multiplexer problem even with a low parameter value β. The curves show that XCSTS is much more independent of parameter β. Even lower values of β, of course, ultimately also impair XCSTS's performance. In the case of a more general initial population ($P_{\#} = 1.0$) XCSTS detects accurate classifiers faster and thus hardly suffers at all from the initial over-general population.

Figure 5 shows that XCSTS is much more robust in noisy problems as well. XCSTS solves Gaussian noise with a standard deviation of $\sigma = 250$ better than XCS the $\sigma = 100$ setting. In the case of alternating noise, XCSTS solves the $P_x =$

[2] If not stated differently, τ is set to 0.4 in the subsequent experimental runs.

0.1 noise case faster than XCS the $P_x = 0.05$ case. As expected, the population sizes do not converge to the sizes achieved without noise since subsumption does not apply. Nonetheless, in both noise cases the population sizes decrease indicating the detection of accurate classifiers.

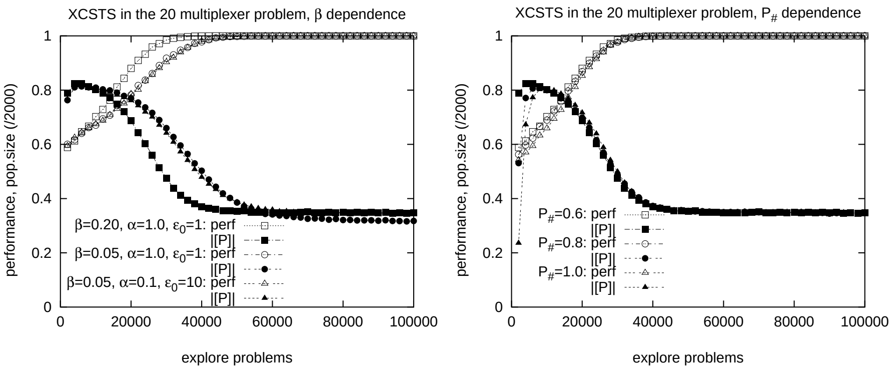

Fig. 4. XCSTS is barely influenced by a decrease of learning rate β. Accuracy parameters do not influence learning behavior. An initial over-general population ($P_{\#} = 1.0$) does hardly influence learning, either (right-hand side).

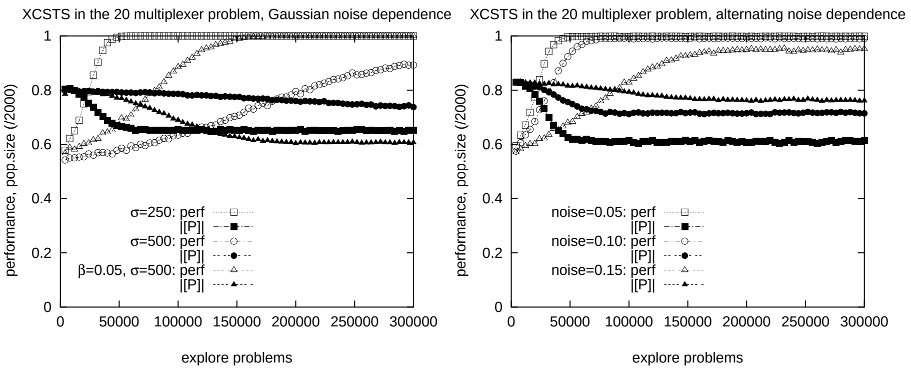

Fig. 5. XCSTS performs much better than XCS in noisy problems. For example, performance of XCSTS with Gaussian noise $\sigma = 250$ is better than XCS's performance with $\sigma = 50$ noise. Lowering β allows XCSTS to solve problems with even $\sigma = 500$.

5 Experimental Study

This section investigates XCSTS's behavior further. We show the effect of a fixed tournament size and different proportional tournament sizes τ. We also apply XCS to the multiplexer problem with layered reward as well as to the *layered count ones problem.* In the latter problem, recombinatory events are highly beneficial.

5.1 Fixed Tournament Sizes

XCS's action sets vary in size and in distribution. Dependent on the initial specificity in the population (controlled by parameter $P_{\#}$), the average action set size is either large or small initially. It was shown that the average specificity in an action set is always smaller than the specificity in the whole population [19]. Replication in action sets and deletion from the whole population results in an implicit generalization pressure that can only be overcome by a sufficiently large specialization pressure. Additionally, the distribution of the specificities depends on initial specificity, problem properties, the resulting fitness pressure, and learning dynamics. Thus, an approach with fixed tournament size is very problem dependent and probably not flexible enough.

In Figure 6 we show that XCSTS with fixed tournament size only solves the multiplexer problem with the large tournament size of 12. With lower tournament sizes not enough selection pressure is generated. Since the population is usually over-populated with over-general classifiers early in a run, action set sizes are large so that a too small tournament size usually results in a competition among over general classifiers. Thus, not enough fitness pressure is generated. Adding noise, an even larger tournament size is necessary to learn. A tournament size of 32, however, does not allow any useful recombinatory events anymore since the action set size itself is usually not much bigger than that. Thus, fixed tournament sizes are inappropriate for XCS's selection mechanism.

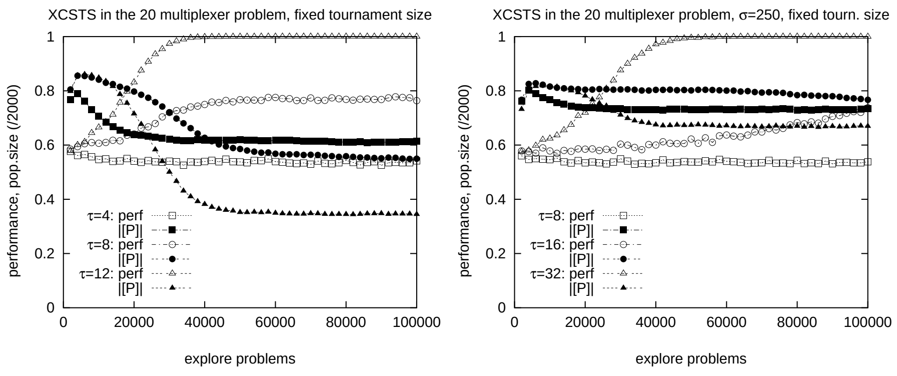

Fig. 6. Due to fluctuations in action set sizes and distributions as well as the higher proportion of more general classifiers in action sets, fixed tournament sizes are inappropriate for selection in XCSTS.

5.2 Layered Boolean Multiplexer

The layered multiplexer [4] provides useful *fitness guidance* for XCS [18]. The more position bits are specified (starting from the most significant one) the less different reward levels are available and the closer the values of the different reward levels are together so that classifiers that have more position bits specified have higher accuracy. Consequently, those classifiers get propagated. Thus, the reward scheme somewhat simplifies the problem.

The exact equation of the reward scheme is $Int(positionbits) * 200 + Int(referencedbit) * 100$. An additional reward of 300 is added if the classification was correct. For example, in the 6-multiplexer the (incorrect) classification 0 of instance (100010) would result in a payoff of 500 while the (correct) classification 1 would result in a payoff of 800.

We ran a series of experiments in the layered 20-Multiplexer increasing Gaussian noise. Figure 7 shows results for noise values $\sigma = 100$ and $\sigma = 300$. With a noise with standard deviation $\sigma = 100$, XCS as well as XCSTS have no problem to solve the task even with the lower learning rate $\beta = 0.05$ for XCS. XCSTS however already learns faster than XCS. With a noise of $\sigma = 300$ performance of XCS does hardly increase at all. XCSTS learns with either learning rate setting but reaches 100% knowledge only with a learning rate of $\beta = 0.05$. This shows that a lower learning rate is necessary in noisy problems since otherwise fluctuations in classifier parameter values can cause disruptions.

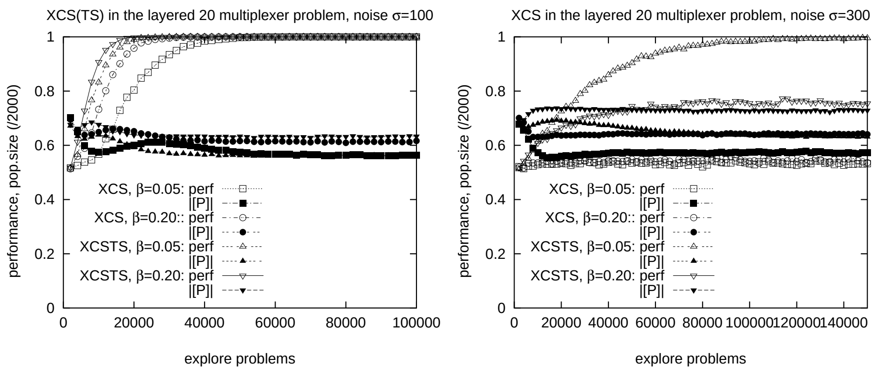

Fig. 7. Gaussian noise disrupts performance of XCS more than performance of XCSTS. While in a problem with little noise a higher value of parameter β allows faster learning, in settings with more noise high β values cause too high fluctuations and are thus disruptive. XCS fails to solve a noise level $\sigma = 300$ with either setting of parameter β while XCSTS still reaches 100% performance with parameter β set to 0.05.

5.3 Layered Count Ones Problem

The final Boolean function in this study is constructed to reveal the benefit of recombination in XCS(TS). While in GA research whole conferences are filled with studies on crossover operators, linkage learning GAs, or the more recent probabilistic model building GAs (PMBGAs) [20], LCS research somewhat neglected the investigation of the crossover operator. Herein, we take a first step towards understanding the possible usefulness of recombination in XCS by constructing a problem in which the recombination of low-fit classifiers can increasingly result in higher fitness.

We term the problem the *layered count ones* problem. In this problem a subset of a Boolean string determines the class of the problem. If the number of ones in this subset is greater than half the size of the subset, then the class is one and otherwise zero. Additionally, we layer the payoff somewhat similar to the layered multiplexer problem. The amount of payoff is dependent on the number of ones in the relevant bits and not the integer value of the relevant bits. For example, consider the layered count ones problem 5/3 (a binary string of length five with three relevant bits) and a maximal payoff of 1000. Assuming that the first three bits are relevant, the classification of 1 of string 10100 would be correct and would result in a payoff of 666 while the incorrect classification 0 would result in a payoff of 333. Similarly, a classification of 1 of string 00010 would result in a payoff of 0 while the correct classification 0 would give a payoff of 1000. Thus, always the correct classification results in the higher payoff.

Figure 8 shows runs in the layered count ones problem with string length $L = 70$ and five relevant bits. Without any additional Gaussian noise, XCSTS outperforms XCS. To solve the problem at all, the initial specificity needs to be set very low ($P_{\#} = 0.9$) to avoid the *covering challenge* [18]. Mutation needs to be set low enough to avoid over-specialization via mutation [19]. Runs with a mutation rate $\mu = 0.04$ (not shown) failed to solve the problem. Performance stalls at approximately 0.8 regardless of the selection type. Figure 8 (left-hand side) also shows that a higher tournament size proportion τ decreases the probability of recombining different classifiers. Thus, the evolutionary process cannot benefit from efficient recombination and learning speed decreases. Note however that proportionate selection is still worse than the selection type that always chooses the classifier with the highest fitness ($\tau = 1.0$). This indicates that recombination, albeit helpful, does not seem to be mandatory. The more important criterion for a reliable convergence is a proper selection pressure.

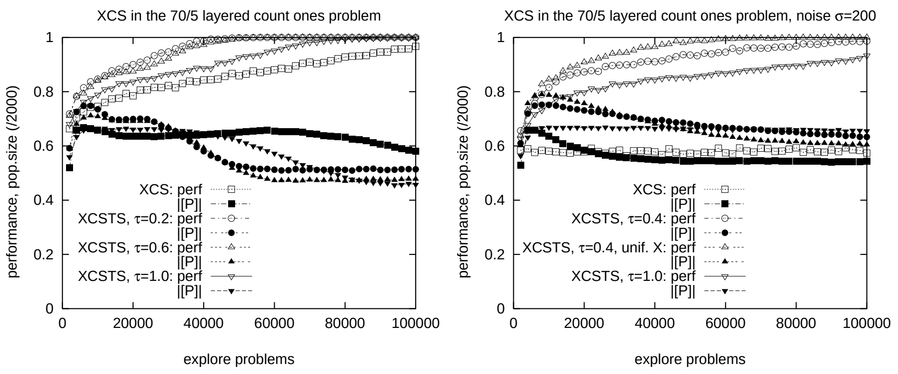

Fig. 8. Also in the layered count ones problem tournament selection has better performance. The benefit of crossover becomes clearly visible. Performance increases when uniform crossover is applied whereas it decreases when no mixing ($\tau = 1.0$) can occur.

Adding Gaussian noise of $\sigma = 250$ to the layered count ones problem again completely deteriorates XCS's performance (Figure 8, right-hand side). Noise hinders mutation even more from doing the trick. After $100,000$ learning steps only selection with proper recombination was able to allow the evolution of perfect performance (Figure 8). Selection of always the best classifier hinders crossover from recombining classifiers that are partially specified in the relevant bits so that the evolutionary process is significantly delayed.

6 Summary and Conclusions

This paper showed that proportionate selection can prevent proper fitness pressure and thus successful learning in XCS. Applying tournament selection selection results in better (1) parameter independence, (2) noise robustness, and (3) recombinatory efficiency. No problem was found in which XCS with tournament selection with an action-set proportionate tournament size performed worse than proportionate selection. Thus, it is clear that future research in XCS should start using tournament selection instead of proportionate selection.

Despite the recent first insights in problem difficulty it remains unclear which problems are really hard for XCS. Moreover, it remains to be shown in which machine learning problems crossover is actually useful or even mandatory for successful learning in XCS and LCSs in general. In XCS the recombination of important substructures should lead to successively higher accuracy. In strength-based LCSs, on the other hand, the recombination of substructures should lead to larger payoff. Due to the apparent importance of recombinatory events in nature as well as in GAs, these issues deserve further detailed investigations in the realm of LCSs.

Acknowledgment

We are grateful to Xavier Llora, Martin Pelikan, Abhishek Sinha, and the whole IlliGAL lab for their help with equation solving and the useful discussions. The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Additional funding from the German research foundation (DFG) under grant DFG HO1301/4-3 is acknowledged. 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. Holland, J.H.: Adaptation. In Rosen, R., Snell, F., eds.: Progress in theoretical biology. Volume 4. Academic Press, New York (1976) 263–293
2. Booker, L.B., Goldberg, D.E., Holland, J.H.: Classifier systems and genetic algorithms. Artificial Intelligence **40** (1989) 235–282
3. Holland, J.H.: Adaptation in Natural and Artificial Systems. Universtiy of Michigan Press, Ann Arbor, MI (1975) second edition 1992.

4. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
5. Wilson, S.W.: Get real! XCS with continuous-valued inputs. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Learning Classifier Systems: From Foundations to Applications. Springer-Verlag, Berlin Heidelberg (2000) 209–219
6. Bernadó, E., Llorà, X., Garrell, J.M.: XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001. Springer-Verlag, Berlin Heidelberg (2002) 115–132
7. Dixon, P.W., Corne, D.W., Oates, M.J.: A preliminary investigation of modified XCS as a generic data mining tool. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001. Springer-Verlag, Berlin Heidelberg (2002) 133–150
8. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms (1991) 69–93
9. Goldberg, D.E., Sastry, K.: A practical schema theorem for genetic algorithm design and tuning. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 328–335
10. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–258
11. Lanzi, P.L.: An analysis of generalization in the XCS classifier system. Evolutionary Computation **7** (1999) 125–149
12. Lanzi, P.L., Colombetti, M.: An extension to the XCS classifier system for stochastic environments. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) (1999) 353–360
13. Barry, A.: A hierarchical XCS for long path environments. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 913–920
14. Butz, M.V., Wilson, S.W.: An algorithmic description of 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) 253–272
15. Venturini, G.: Adaptation in dynamic environments through a minimal probability of exploration. From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (1994) 371–381
16. Wilson, S.W.: Generalization in the XCS classifier system. Genetic Programming 1998: Proceedings of the Third Annual Conference (1998) 665–674
17. De Jong, K.A., Spears, W.M.: Learning concept classification rules using genetic algorithms. IJCAI-91 Proceedings of the Twelfth International Conference on Artificial Intelligence (1991) 651–656
18. 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
19. Butz, M.V., Pelikan, M.: Analyzing the evolutionary pressures in XCS. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 935–942
20. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications **21** (2002) 5–20