Strong, Stable, and Reliable Fitness Pressure in XCS due to Tournament Selection

Martin V. Butz, Kumara Sastry and David E. Goldberg
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. Recent analysis of the XCS classifier system have shown that successful genetic learning strongly depends on the amount of fitness pressure towards accurate classifiers. Since the traditionally used proportionate selection is dependent on fitness scaling and fitness distribution, the resulting evolutionary fitness pressure may be neither stable nor sufficiently strong. Thus, we apply tournament selection to XCS. In particular, we exhibit the weakness of proportionate selection and suggest tournament selection as a more reliable alternative. We show that tournament selection results in a learning classifier system that is more parameter independent, noise independent, and more efficient in exploiting fitness guidance in single-step problems as well as multistep problems. The evolving population is more focused on promising subregions of the problem space and thus finds the desired accurate, maximally general representation faster and more reliably.

Keywords: Learning Classifier Systems, XCS, Tournament Selection, Genetics Based Machine Learning

1. Introduction

Learning Classifier Systems (LCSs) [18, 3] are rule-based learning systems. A basic LCS consists of a set of rules, an apportionment of credit system, and a rule evolution mechanism, which is usually implemented by a genetic algorithm (GA) [17]. The classifier population codes the current knowledge of the LCS. The apportionment of credit system, which often applies adapted reinforcement learning [19, 29] techniques, estimates rule utility. Based on the estimated utilities, the evolutionary mechanism is designed to generate better offspring.

Learning classifier systems can be distinguished between online learning LCSs and offline learning LCSs. Moreover, they can be distinguished between LCSs that evolve a single solution, often referred to as Michigan-style LCSs, and LCSs that evolve a set of solutions, often referred to as Pittsburgh-style LCSs. Pittsburgh-style LCSs are usually applied in offline-learning scenarios only. While our study of selection mechanisms focuses on one particular online-learning Michigan-style LCS, that is the accuracy-based learning classifier system XCS, we would like to emphasize that the basic analysis and comparisons as well as the drawn conclusions should readily carry over to other types of LCSs neither restricted to online-learning LCSs nor to Michigan-style LCSs.

As most other LCSs, the accuracy-based XCS evolves a population of condition-action prediction classifiers by the means of a GA. The selection mechanism was traditionally implemented using proportionate selection. Although many indicators can be found in the GA literature that suggest that proportionate selection is strongly dependent on fitness scaling [14] and on the fitness distribution in the current population [15], the LCS community has largely ignored this insight. In XCS, fitness is an estimate of the scaled relative accuracy of a classifier. Since the scaling usually assures the successful application of proportionate selection, it did not seem necessary to investigate other selection methods.

In this paper, we identify problem types in which proportionate selection does not result in a sufficiently strong fitness pressure so that XCS is not able to learn the problem. We show that using tournament selection the problem can be overcome allowing XCS to solve a broader class of typical classification problems. Moreover, we show that the tournament selection mechanism is also successful in multistep problems usually outperforming proportionate selection. Tournament selection results in a more stable, more reliable, and more focused pressure towards higher accuracy.

The aim of this study is threefold. First, we analyze and improve selection mechanisms in LCSs. Second, we contribute to the understanding of the functioning of XCS and LCSs in general. Finally, we prepare XCS and LCSs to solve decomposable machine learning problems quickly, accurately, and reliably.

The next section gives a short overview over the XCS classifier system focusing on XCS's proportionate selection mechanism. Section 3 identifies problem types in which proportionate selection does not cause an appropriate fitness pressure resulting in a learning failure of XCS. Section 4 introduces tournament selection and shows its superiority in the investigated problems. Section 5 investigates the two selection mechanisms further studying different environments and crucial parameter settings. Section 5.5 compares learning behavior in multistep problems. Finally, summary and conclusions are provided.

2. XCS in a Nutshell

XCS evolves a set of rules, the so-called *population* of *classifiers*. Rules are evolved by the means of a GA. 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, or classification, to execute. In contrast to the original LCSs, the fitness in the XCS classifier system, introduced by Wilson [33], is based on the accuracy of reward predictions rather than on the reward predictions directly. Thus, XCS is meant to evolve not only 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 comparably well to several other typical classification algorithms in many standard machine learning problems [1, 11, 2].

This section provides a short introduction to the XCS classifier system. We emphasize the underlying ideas and assumptions that enable the system to evolve a complete, accurate, and maximally general payoff map reliably. For a more detailed introduction to XCS the interested reader is referred to the original paper [33] and the algorithmic description [9].

2.1. Problem Types

XCS can be applied to classification problems as well as multistep problems. While only immediate reinforcement is relevant in the former, the necessary reinforcement propagation results in an additional challenge in the latter.

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 $\rho \in \Re$ 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 $\mathcal{A} = \{0,1\}$.

Somewhat similarly we define a multistep problem as a Markov decision process (MDP). A problem consists of a set of possible sensory inputs $s \in \mathcal{S}$ (i.e. the states in the MDP), a set of possible actions

$a \in \mathcal{A}$, a transition function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and a reinforcement function $\nabla : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$. Note that we restrict the multistep problems to MDP problems. Applications of XCS to POMDP problems can be found in [27, 25], in which the system is enhanced with internal memory. XCS is designed to learn a complete, accurate, and maximally general representation of the Q-table of the underlying MDP problem [33, 26]. As in the classification problem case we restrict the sensory state space to binary inputs $\mathcal{S} \subseteq \{0,1\}^L$. More than two actions may be available, though.

2.2. Knowledge Representation

As mentioned, 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 input space in which the classifier is applicable, or *matches*. The "don't care" symbol # matches both 0 and 1. The action part $A \in \mathcal{A}$ specifies the advocated action, or classification. The payoff prediction p estimates the average payoff encountered after executing action A in the situations in which condition C matches. The prediction error ε estimates the average deviation, or error, of the payoff prediction p. The fitness F reflects the scaled average relative accuracy of the classifier with respect to other overlapping classifiers.

Learning usually starts with an empty population. Given current input s, the set of all classifiers in $[P]$ whose conditions match s is called the *match set* $[M]$. If some action is not represented in $[M]$, a covering mechanism is applied. Covering creates classifiers that match s (#-symbols are inserted with a probability of $P_\#$ at each position) and specify the not covered actions. Given a match set, XCS can estimate the payoff for each possible action forming a *prediction array* $P(\mathcal{A})$,

$$P(a) = \frac{\sum_{cl.a=a \wedge cl \in [M]} cl.p \cdot cl.F}{\sum_{cl.a=a \wedge cl \in [M]} cl.F}. \tag{1}$$

Classifier parameters are addressed using the dot notation. Essentially, $P(a)$ reflects 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. During learning, XCS chooses actions randomly. During testing, the action a_{max} with the highest value $P(a_{max})$ is chosen.

2.3. Rule Evaluation

XCS iteratively updates its population of classifiers with respect to the successive problem instances. After the classification is selected by the means of the prediction array and applied to the problem, the scalar feedback ρ is received. In a classification problem, classifier parameters are updated with respect to the immediate feedback in the current action set $[A]$, which comprises all classifiers in $[M]$ that advocate the chosen classification a. In a multistep problem, classifiers are updated in the previous action set $[A]_{-1}$ with respect to the previous reward ρ_{-1} and the maximum value in the prediction array $max_{a \in \mathcal{A}} P(a)$. After rule evaluation and possible GA invocation, the next iteration starts.

Parameter updates are usually done in the order: prediction error, prediction, fitness. Prediction error is updated by $\varepsilon \leftarrow \varepsilon + \beta(|R-p| - \varepsilon)$ where β ($\beta \in [0,1]$) denotes the *learning rate* and R denotes the immediate feedback ρ in a classification problem and the combination of previous immediate feedback and maximum predicted reward $\rho_{-1} + \gamma max_{a \in \mathcal{A}} P(a)$, where γ denotes the discount factor, in a multistep problem. Next, prediction is updated in $[A]$ by $p \leftarrow p + \beta(R - p)$. The fitness value of each classifier in $[A]$ is updated with respect to its current scaled relative accuracy κ':

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0 \\ \alpha \left(\frac{\varepsilon_0}{\varepsilon}\right)^{\nu} & \text{otherwise} \end{cases} \tag{2}$$

$$\kappa' = \frac{\kappa \cdot num}{\sum_{x \in [A]} \kappa_x \cdot num_x} \tag{3}$$

$$F \leftarrow F + \beta(\kappa' - F) \tag{4}$$

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 (left-hand side) 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 [31]. This technique sets parameter values directly to the average of the so far encountered cases as long as the experience of a classifier is less than $1/\beta$. Each time the parameters of a classifier are updated, the experience counter *exp* of the classifier is increased by one.

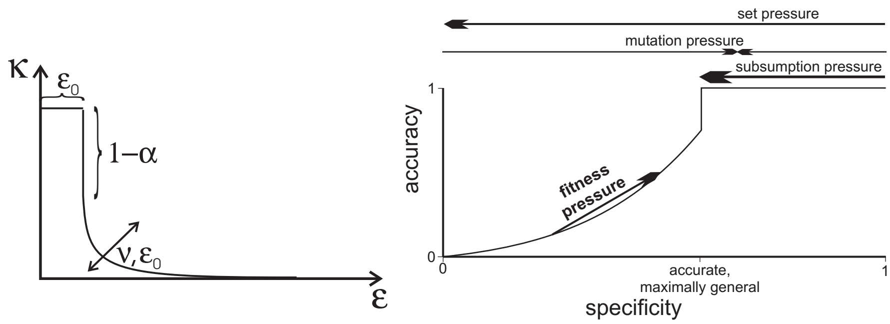

Figure 1. Left: The scaling of accuracy κ is crucial for successful proportionate selection. Parameters ε_0, α, and ν control tolerance, offset, and slope, respectively. *Right*: The sketched evolutionary pressures along the specificity axis give an idea how the evolutionary algorithm in XCS is designed to evolve accurate, maximally general classifiers. A strong fitness pressure is crucial to evolve accurate classifiers.

2.4. Rule Evolution

Besides the aforementioned covering mechanism that ensures that all actions in a particular problem instance are represented by at least one classifier, XCS applies a GA for rule evolution. A GA is invoked if the average time since the last GA application (stored in the additional classifier parameter ts) 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 crossover (uniform 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. This is in difference to the often applied *niche mutation* that assures that the mutated classifier still matches the current problem instance. Offspring parameters are initialized by setting prediction p to the actual value R, prediction error ε to the average error in the action set, fitness F to 10% of the parental fitness, experience counter exp to one, and numerosity num to one. This initialization is slightly different from the one used in the algorithmic description but showed slightly better performance. A comparison of the two initialization methods is provided in Figure 3. In the insertion process, *subsumption deletion* may be applied [34] to stress generalization. Due to the possible strong effects of *action-set subsumption* we apply *GA subsumption* only.

GA subsumption checks offspring classifiers to see whether their conditions are logically subsumed by the condition of another *accurate* and *sufficiently experienced* classifier in $[A]$. If an offspring is subsumed, it is not inserted in the population but the subsumer's numerosity is increased.

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 and updated similar to prediction p). 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.

2.5. Basic XCS Theory

XCS is designed to evolve a complete reward map of a given problem. Previous investigations in XCS theory include the work of Kovacs on problem difficulty. Most importantly, Kovacs investigated how problem difficulty increases with the size of the optimal set $[O]$ of a solution to the problem [22]. In the mean time, population sizing bounds were derived assuring supply of important building blocks (substructures in classifier conditions) and reproductive opportunities [6, 4].

The main concern of this paper is to ensure the propagation of better classifiers. Regardless if classification problem or multistep problem, the successful evolutionary learning process is dependent on several evolutionary pressures in XCS: (1) *set pressure*, (2) *mutation pressure*, (3) *fitness pressure*, (4) *deletion pressure*, and (5) *subsumption pressure* [6]. All pressures have an influence on the overall specificity of the classifiers in the population. A specificity of 1 denotes that all attributes in a condition part are either zero or one whereas a specificity of 0 denotes that all attributes are #-symbols. The set pressure denotes the tendency of XCS to reproduce more general classifiers since the average specificity in an action set is lower than the current average specificity in the population. Mutation pressure pushes towards an equal support of symbols in the attributes of a condition part ($\frac{1}{3}:\frac{1}{3}:\frac{1}{3}$ for the ternary alphabet $\{0, 1, \#\}$). Thus, if the population is more general than $\frac{2}{3}$ specificity, mutation pushes towards higher specificity. When starting with a completely general population, mutation introduces initial specificity. Fitness pressure pushes towards higher accuracy. Deletion pressure is rather weak but is meant to push towards an equal niche support. Finally, subsumption pressure only applies in deterministic problems since it requires complete accuracy. It pushes towards accurate, syntactically maximally general classifiers. The pressures are sketched in Figure 1 (right-hand side) along an imaginary specificity axis.

Most important for the remainder of this paper are set pressure, mutation pressure, and fitness pressure. While the set pressure continuously generalizes the population, mutation pressure specializes the population undirectedly (assuming the population is more general than $\frac{2}{3}$), and fitness pressure specializes the population towards higher accuracy. Only if fitness pressure is larger than the set pressure (possibly with the support of mutation pressure) XCS can successfully evolve accurate classifiers. The next section will show that proportionate selection is not necessarily sufficient to generate a strong-enough specialization pressure towards higher accuracy.

3. Trouble for Proportionate Selection

Proportionate selection depends on the fitness relative to action set $[A]$. The fitness F of a classifier is strongly influenced by the accuracy determination of Equation 2 as visualized in Figure 1 (left-hand side). 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 in its fitness value. To infer these dependencies, we apply XCS to the multiplexer problem with altered parameter settings or with additional noise in the problem.

The multiplexer problem is a widely studied problem in LCS research [10, 33, 34, 6]. It has been shown that LCSs are superior compared to standard machine learning algorithms, such as *C4.5*, in the multiplexer task [10]. 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 is that over-general classifiers occupy a big part of the population initially. Better offspring often loose against the over-general

[1] If not stated differently, 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: $N = 2000$, $\beta = 0.2$, $\alpha = 1$, $\varepsilon_0 = 10$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 0.8$, $\mu = 0.04$, $\gamma = 0.9$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, and $P_{\#} = 0.6$.

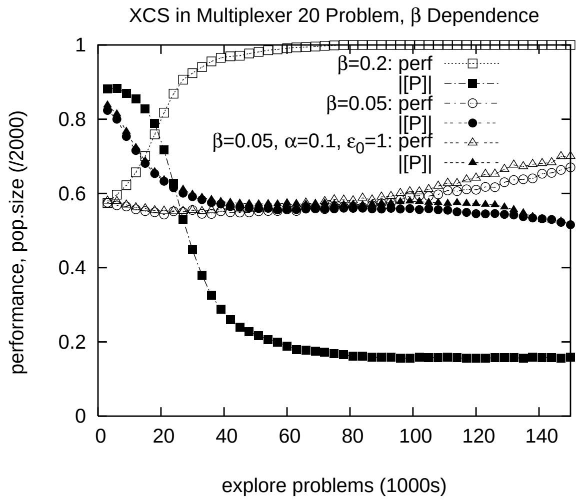

Figure 2. A lower learning rate β decreases XCS learning performance. Accuracy function parameters have no immediate positive influence.

parents since the fitness of the offspring only increases slowly (due to the low β value). 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.

Figure 3 reveals XCS's dependence on initial specificity. Increasing $P_\#$ (effectively decreasing initial specificity) impairs the learning speed of XCS facing the *schema challenge* [6]. Decreasing the mutation rate μ also has a detrimental effect strongly delaying learning progress. Apparently, the generalizing set pressure is often stronger than the specializing fitness pressure so that the chances of reaching maximal accuracy are significantly decreased. Additionally, we detected significant parameter initialization effects. If prediction p and error ε in offspring are set to the parental values, learning speed is decreased. When setting prediction to the actual value, accurate classifiers can be detected faster. Moreover, setting error to the average error in the action set seems to prevent error underestimation of inaccurate offspring.

Additional to the dependence on parameters β, $P_\#$, μ, and initialization, 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 with a standard deviation σ 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. Figures 4 and 5 show that XCS's performance is strongly degraded when adding only

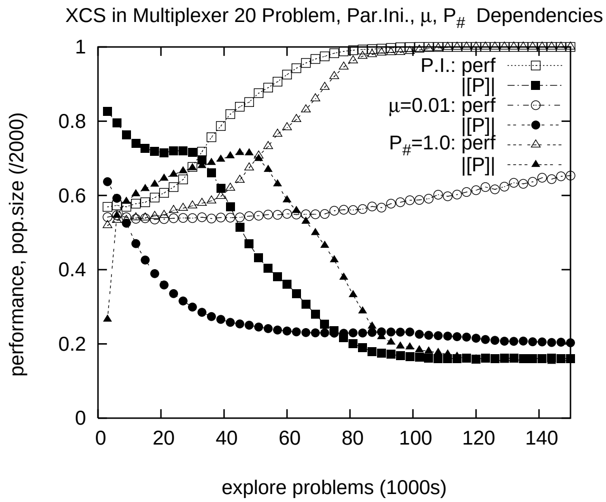

Figure 3. Proportionate selection is strongly dependent on offspring initialization, mutation rate, and initial specificity.

a small amount of either noise. Similar observations were made in [21], who added Gaussian noise to the reward in some of the output classes. In general, the more noise is added, the smaller the fitness difference between accurate and inaccurate classifiers. Thus, selection pressure decreases due to proportionate selection and the population starts to drift at random. Lanzi [24] proposed an extension to XCS that detects noise in environments and adjusts the error estimates accordingly. This approach, however, does not solve the parameter dependencies.

4. Stable Selection Pressure with Tournament Selection

In contrast to proportionate selection, tournament selection is independent of fitness scaling [14]. In tournament selection parental classifiers are not selected proportional to their fitness, but tournaments are held in which the classifier with 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. Fixed tournament sizes are generally used in GAs.

Compared to standard GAs, the GA in XCS is a steady-state, niched GA. Only two classifiers are selected in each GA application and selection is restricted to the classifiers in 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

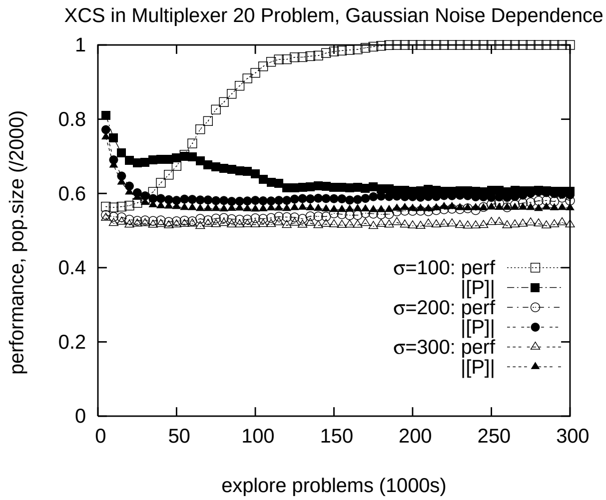

Figure 4. Adding Gaussian noise to the payoff function of the multiplexer problem significantly deteriorates performance of XCS.

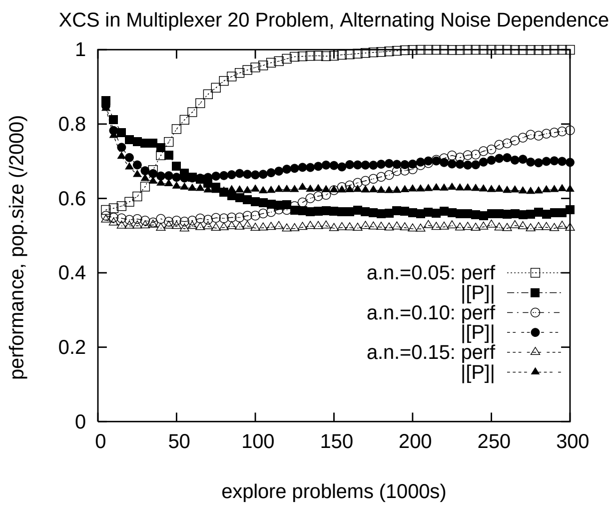

Figure 5. Also alternating noise, in which a percentage of examples is assigned to the incorrect class, significantly degrades performance of XCS.

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 $|[A]|$ choosing a subset of size $\tau|[A]|$

($\tau \in (0,1]$) of the classifiers in $[A]^2$. 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, shown below, which did not result in a stable selection pressure.

4.1. XCSTS in the Previous 20 Multiplexer Settings

Figures 6 and 7 show 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, a low parameter value μ, or a high parameter value $P_{\#}$. XCSTS is also more independent from initial parameter settings. The much stronger and stable fitness pressure overcomes the generalizing set pressure even without the help of mutation pressure. A reliable and stable performance increase is observable.

Figures 8 and 9 show that XCSTS is much more robust in noisy problems as well. XCSTS solves the same Gaussian noise 20 multiplexer problem with nearly no performance degradation (Figure 8). Despite the noisy parameter estimation values, tournament selection detects the more accurate classifiers generating a sufficiently strong fitness pressure. The decrease in the differences of accuracy due to the additional noise hardly affects XCSTS. Also in the alternating noise case, XCSTS reaches a higher performance level (Figure 9).

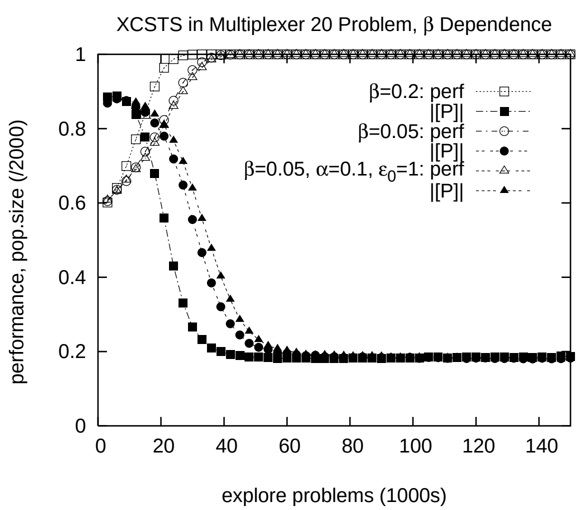

Figure 6. XCSTS is hardly influenced by a decrease in learning rate β. As in the proportionate selection case, accuracy parameters do not influence learning.

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

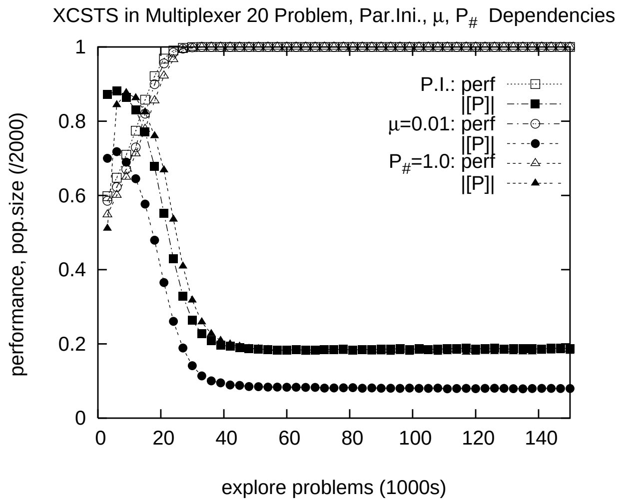

Figure 7. Other parameter variations such as a lower initial specificity ($P_{\#} = 1.0$), a low mutation rate ($\mu = 0.01$), or a different offspring initialization (*P.I.*) does hardly influence XCSTS's learning behavior.

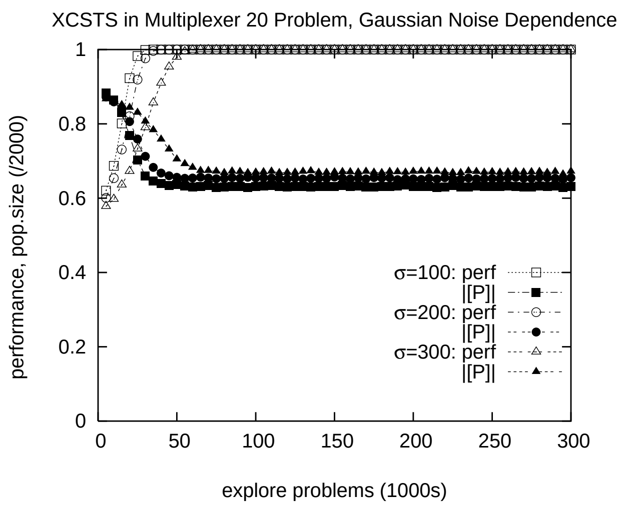

Figure 8. XCSTS performs much better than XCS in noisy problems. Performance is hardly influenced when adding Gaussian noise of up to $\sigma = 300$

4.2. Tournament Selection with Fixed Size

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 on average smaller than the

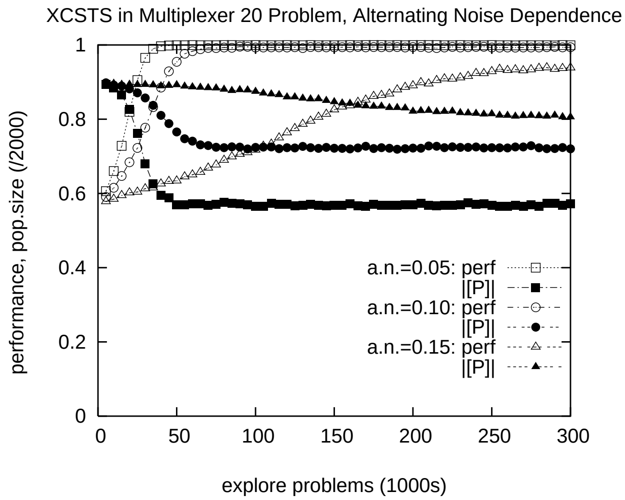

Figure 9. Also in the alternating noise case, XCSTS's learning behavior is faster and more reliable.

specificity in the whole population [7]. 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 quite dependent on the particular problem and probably not flexible enough. Kovacs used fixed tournament sizes to increase fitness pressure in his comparison to a strength-based XCS version [20, 21].

In Figure 10 we show that XCSTS with fixed tournament size τ_f only solves the multiplexer problem with the large tournament size of $\tau_f = 12$. 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 small tournament size mainly causes competition only among over-general classifiers. Thus, not enough fitness pressure is generated. Adding noise, an even larger tournament size is necessary (Figure 11). A tournament size of $\tau_f = 32$, 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.

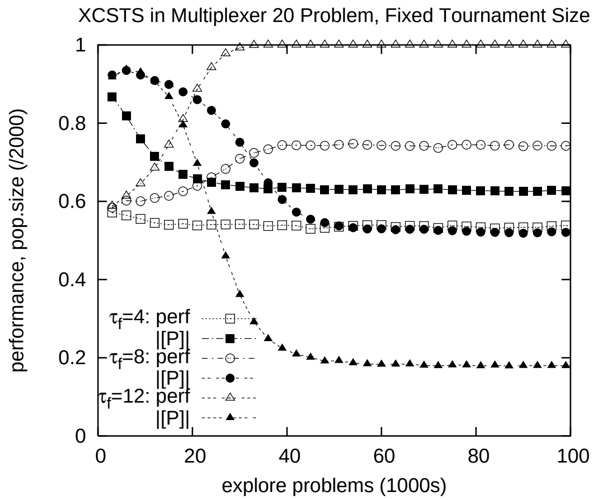

Figure 10. Due to fluctuations in action set sizes and distributions as well as the higher proportion of more general classifiers in action sets, fixed tournament sizes are inappropriate for selection in XCSTS.

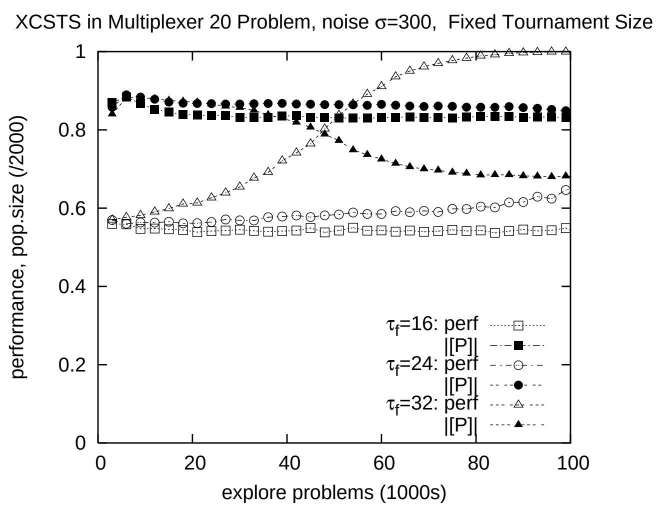

Figure 11. In the noisy problem case, tournament selection with a fixed tournament size can hardly solve the problem.

4.3. Different Tournament Sizes

A change in the relative tournament size effectively changes the strength of the selection pressure applied. While a tournament size $\tau = 0$ corresponds to random selection, $\tau = 1$ corresponds to a deterministic selection of the classifier with the currently highest fitness

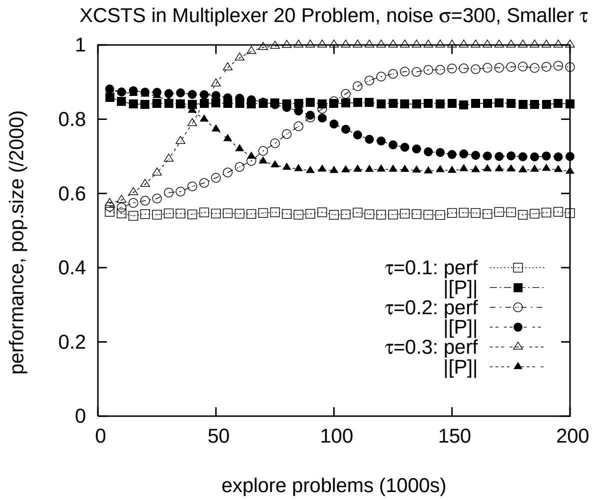

Figure 12. The strength of the selection pressure can be manipulated by the tournament size proportion. Very small proportions result in a not strong-enough pressure.

in the action set and thus the strongest selection pressure. As can be seen in Figure 12 and Figure 13, XCSTS is able to generate a complete and accurate problem representation for a large range of τ. However, if selection pressure is too weak, learning may not take place at all or may be delayed. On the other hand, if selection pressure is very strong, crossover never has any effect since identical classifiers are crossed. The lack of effective recombinations does hardly influence XCS performance in the multiplexer problem as shown in Figure 13. However, in other problems ineffective crossover may strongly impair XCS's learning capabilities. Thus, τ may not be set to 1. In practice, a value of 0.4 proved to be robust.

5. Experimental Study

This section investigates XCSTS's behavior further and provides further comparisons to XCS with proportionate selection. We reveal the stronger and more reliable fitness guidance showing the much faster increase in specificity as well as the independence of initial specificity. We also investigate the maximum amount of noise XCSTS is able to handle. Moreover, we compare performance in the large 70 multiplexer problem and the layered count ones problem revealing the necessity of low mutation rates and low learning rates. Finally, we provide an

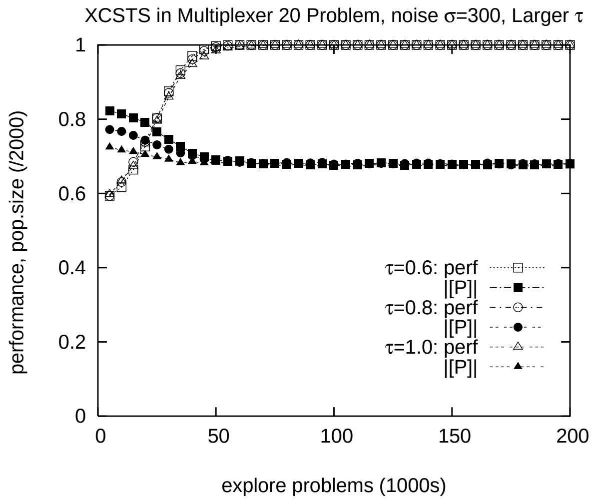

Figure 13. Very large tournament size proportions may prevent effective recombination. In the multiplexer problem, this restriction has hardly any performance influence.

experimental investigation in the two multistep mazes Woods2 and Maze6.

5.1. Specificity Guidance Exhibited

As theorized above, the two selection methods differ in their dependence on the fitness estimate. The fitness pressure, resulting from proportionate selection, depends on fitness scaling and in particular on the relative amount of fitness difference. Tournament selection on the other hand only depends on the fitness difference itself and not on the amount of the difference. That is, as long as the more accurate classifiers have a higher fitness estimate (regardless of how much higher the estimate is), tournament selection causes a sufficient fitness pressure towards higher accuracy.

To exhibit this pressure, we monitored the average specificity in the population as well as the average standard deviation of the specificity. Figures 14 and 15 show the change in specificity when starting with a completely general population ($P_{\#} = 1.0$) in the 11 multiplexer problem. Figure 15 shows that XCSTS immediately identifies higher accurate classifies causing the specificity to rise. XCS, on the other hand, stalls at the over-general level apparently relying on a lucky guess for successful learning (Figure 14). Another indicator can be found in the performance increase showing also the standard deviation over the experiments. Since XCSTS detects the fitness guidance in the problem

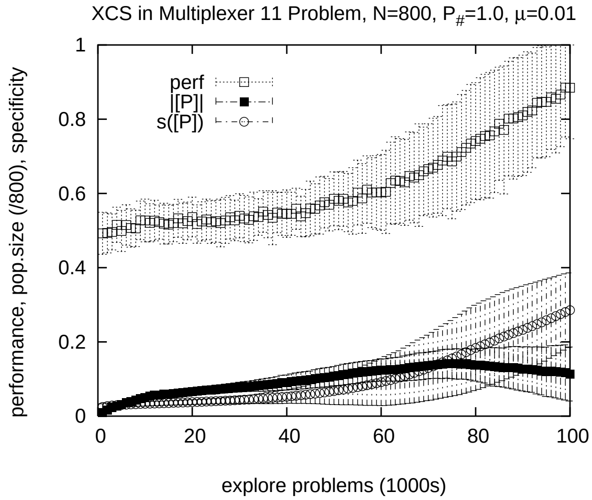

Figure 14. Starting with an over-general population and low mutation rate, XCS is not able to pick up the accuracy signal reliably.

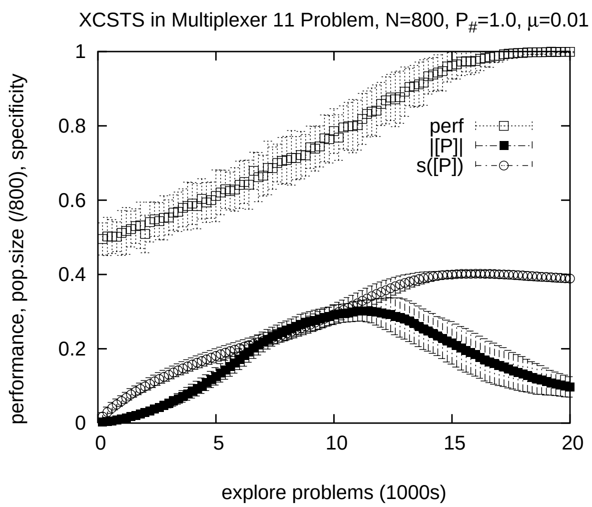

Figure 15. XCSTS immediately pushes the population towards higher specificity and thus higher accuracy.

immediately, the standard deviation between the runs remains small and the performance is hardly affected by a higher initial generality. Performance of XCS with roulette wheel selection on the other hand is much less reliable and strongly depends on initial generality. If there is no accurate classifier generated initially (thus facing the schema challenge), roulette wheel selection has problems to generate a sufficiently strong fitness guidance.

5.2. Very Noisy Problems

Adding Gaussian noise with a large standard deviation σ, also XCSTS has a hard time to evolve an accurate population. Figure 16 shows that the speed of learning is strongly decreased when a noise of $\sigma = 500$ is added. Perfect performance can only be reached if the learning rate β is lowered, effectively decreasing the noise in the parameter estimations of the classifiers. Even higher noise values prevent XCSTS from reaching 100% knowledge reliably (Figure 17). When setting $\sigma = 700$, hardly any learning is observable.

It is possible to determine the actual expected standard deviation of classifiers in problems with additional Gaussian noise. Given two normal distributions $N_1(\mu_1, \sigma_1)$ and $N_2(\mu_2, \sigma_2)$ assuming that a classifier approximates the mean average deviation (MAD), a classifier that is only applicable in situation-classification combinations that yield payoff distributed by N_1 will have approximately a prediction error estimate of $0.8\sigma_1$ (since $MAD(N(\mu, \sigma)) = \sqrt{2/\pi}\sigma \approx 0.8\sigma$). The prediction error estimate of a classifier that encounters the first payoff distribution with probability P and the second payoff distribution with probability $(1 - P)$ can then be approximated as follows:

$$
\begin{aligned}
\sigma^2 &= E(X^2) - E(X)^2 = \\
&= PE(X_1^2) + (1-P)E(X_2^2) - (P\mu_1 + (1-P)\mu_2)^2 \\
&= PE(X_1^2) + (1-P)E(X_2^2) - (P\mu_1)^2 - ((1-P)\mu_2)^2 - 2P(1-P)\mu_1\mu_2 \\
&= P(E(X_1^2) - \mu_1^2) + P\mu_1^2 - (P\mu_1)^2 + (1-P)(E(X_2^2) - \mu_2^2) + (1-P)\mu_2^2 - \\
&\qquad ((1-P)\mu_2)^2 - 2P(1-P)\mu_1\mu_2 \\
&= P\sigma_1^2 + (1-P)\sigma_2^2 + P(1-P)\mu_1^2 + P(1-P)\mu_2^2 - 2P(1-P)\mu_1\mu_2 \\
&= P\sigma_1^2 + (1-P)\sigma_2^2 + P(1-P)(\mu_1 - \mu_2)^2
\end{aligned}
\tag{5}
$$

Since this is not a normal distribution, we cannot use the approximation above. Nonetheless, we have an approximation of the prediction error that enables us to estimate the difficulty of the problem and to predict if a problem is still solvable at all. In the case of $\sigma = 600$, the completely general classifier, for example, would experience a mean of 500 and a standard deviation of $(0.5 \cdot 600^2 + 0.5 \cdot 600^2 + 0.25 \cdot 1000^2)^{0.5} = 781$ so that the problem appears to be still solvable theoretically (since the maximally accurate classifier experiences a standard deviation of $\sigma = 600$). It also shows how much more difficult the problem is than the problem without any noise. Without any noise, accurate classifiers experience a standard deviation of $\sigma = 0$ while the completely general classifiers experience a standard deviation of $\sigma = 500$. The smaller the differences in the experienced standard deviations, the more the evolutionary process will be misled since the deviations are only approximated by the means of temporal difference learning techniques. Even

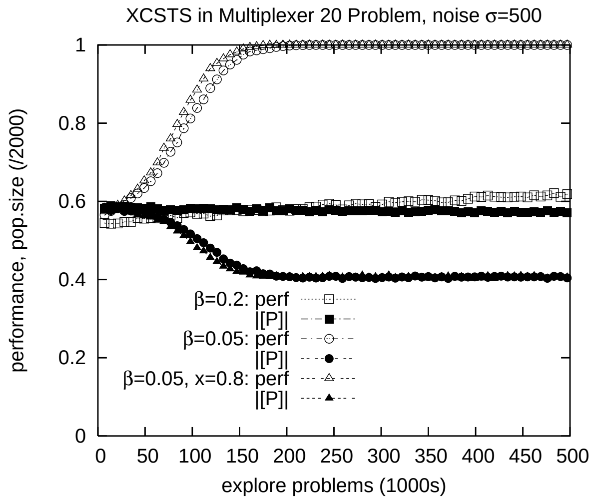

Figure 16. A lower learning rate β helps to derive accurate-enough fitness estimates in highly noisy problems. Crossover appears slightly disruptive.

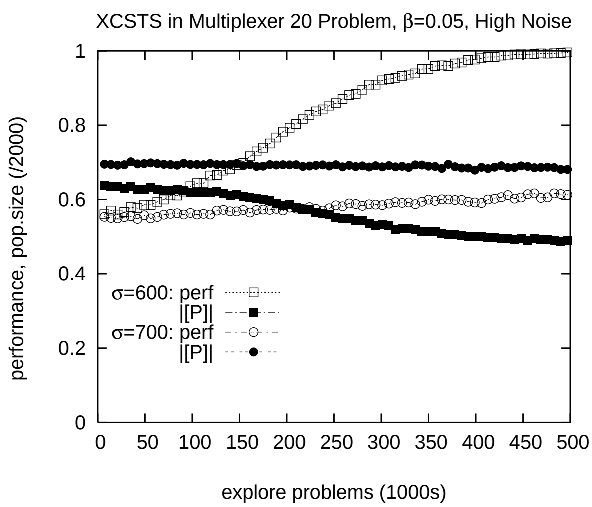

Figure 17. Up to a noise level of $\sigma = 600$, XCSTS is able to solve the Boolean multiplexer problem.

stronger approximation mistakes are expectable and even inevitable in "young" classifiers with only a few parameter updates so far.

5.3. The 70 Multiplexer Problem

Recently, we were able to show that XCS with proportionate selection is able to solve the 70 multiplexer problem [5]. XCS reached 100%

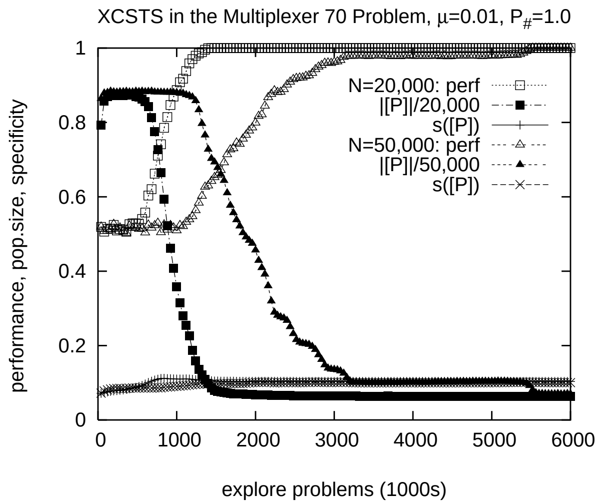

Figure 18. XCSTS reliably solves the very large 70 multiplexer problem. A smaller population size delays the learning progress.

performance in approximately $4,000,000$ learning iterations using a population size of $N = 50,000$, a mutation rate $\mu = 0.04$ (niche mutation), and a sufficiently high initial specificity in the population ($P_{\#} = 0.75$). Figure 18 shows the performance of XCSTS in the 70 multiplexer problem with a population size $N = 50,000$ and a population size of $N = 20,000$, a mutation rate of $\mu = 0.01$, and an initially completely general population ($P_{\#} = 1.0$). The curves are averaged over 25 experiments. The graphs show that XCSTS with a population size of $50,000$ reliably solves the problem within $1,400,000$ learning iterations. Decreasing the population size to $20,000$ results in a much harder problem and all runs except one converged after $3,400,000$ problems. The last run took more than $5,000,000$ problems to find the optimal solution. Due to the small size of the population it is hard to give more accurate classifiers reproductive opportunities often loosing the detected higher accurate classifiers [4]. No XCS run with roulette wheel selection was able to solve the problem within $6,000,000$ learning steps in the applied parameter settings—the specializing fitness pressure apparently was never able to overcome the generalizing set pressure when starting with a completely general population and applying a smaller mutation rate.

5.4. 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 conferences are filled with studies on effective crossover operators, linkage learning GAs, or the more recent probabilistic model building GAs (PMBGAs) [28], LCS research has largely 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-fitness 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 by returning a payoff dependent on the number of ones in 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.

Figures 19 and 20 show runs in the layered count ones problem with string length $L = 100$ and seven relevant bits. XCSTS clearly outperforms XCS exploiting fitness guidance much more effectively. To solve the problem at all, the initial specificity needs to be set very low ($P_{\#} = 1.0$) to avoid the *covering challenge* [6]. Mutation needs to be set low enough to avoid over-specialization via mutation [7]. 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. Due to the long string length, mutation overspecializes the population and learning stalls. Figure 20 shows that recombination is highly beneficial in this problem. Disallowing crossover by setting $\chi = 0$ strongly impairs learning speed. Similarly, applying two-point crossover instead of uniform crossover results in a much less effective recombination in this problem and learning speed is decreased as well. Thus, tournament selection enables successful recombinatory events but effective crossover operators are necessary.

5.5. Multistep Problems

Our argument for the usefulness of tournament selection so far was that for a successful evolutionary process a selection mechanism is necessary that results in a strong and stable fitness pressure. As we argued before

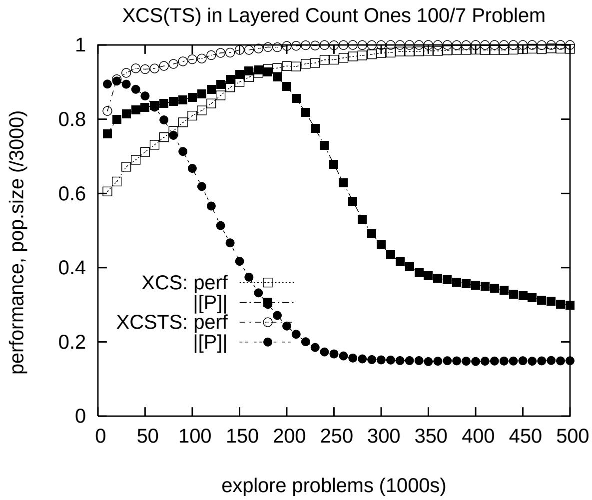

Figure 19. Also in the layered count ones problem tournament selection exhibits better performance.

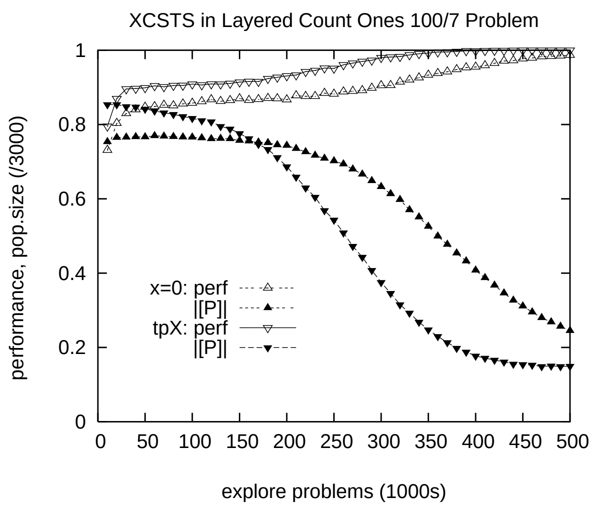

Figure 20. In the layered count ones problem, the benefit of crossover becomes clearly visible since no crossover or a less effective crossover operator decrease learning speed.

[8], the same mechanism is also necessary in a multistep environment in which reinforcement needs to be propagated. As XCS is designed to effectively approximate the Q-function [32], as suggested in [33] and analyzed in [26], tournament selection should detect the accuracy gradient towards accurate Q-values more effectively and more reliably.

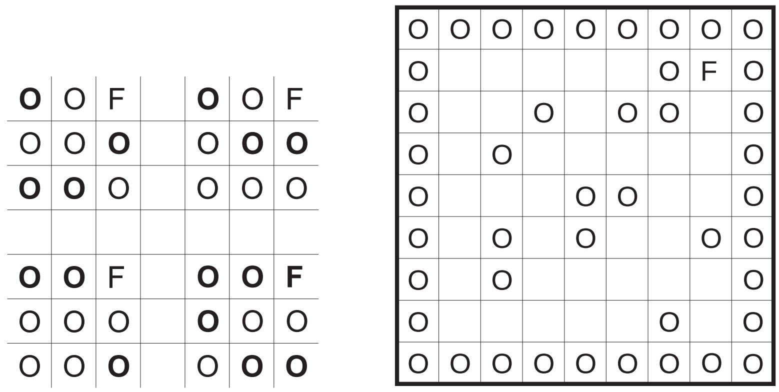

Figure 21. XCS(TS) is tested in two maze problems. Woods2 (left-hand side) extends in all directions indefinitely and provides lots of room for generalization. Maze6 (right-hand side) is hardly generalizable.

Thus, a more reliable convergence in multistep problems should be achieved using tournament selection.

To validate this proposition, we apply XCS to the Woods2 problem introduced in [33] and the Maze6 problem. Both environments are shown in Figure 21. Sensory input in these mazes is coded binary. Perceived are the eight surrounding cells of the current position of XCS starting north and coding clockwise. In Woods2, each position is coded by three bits. The first bit indicates if the food can be found at a coded position. The second bit indicates if the position is blocked by an obstacle. The third position is effectively a random bit that is zero or one dependent on the type of obstacle. The types of obstacles are distributed randomly. Maze6 is coded without the additional random bit. For example, the position just below the food in Maze6 would be coded by `10 01 01 01 00 00 01 01` (spaces are added for clarity) indicating the food to the north and the obstacles surrounding the position. However, note that XCS has information neither about the meaning of the attributes nor about any relations between them nor about any relations with the available actions.

Woods2 is a rather simple problem since the distance to the food is maximally two in any position of the maze. Lots of generalizations are possible. Figure 22 compares performance of XCS (left-hand side) and XCSTS (right-hand side)[3]. It can be seen that XCS and XCSTS perform nearly identically well. Moreover, the environment appears to

[3] Experiments are divided in learn and test trials monitoring performance in the test trials. As usually done in the literature on multistep problems, rule evaluation (but not rule evolution) also takes place in the test trials. Moreover, the number of steps in one trial is restricted to 100 steps to prevent excessive looping or hiding of

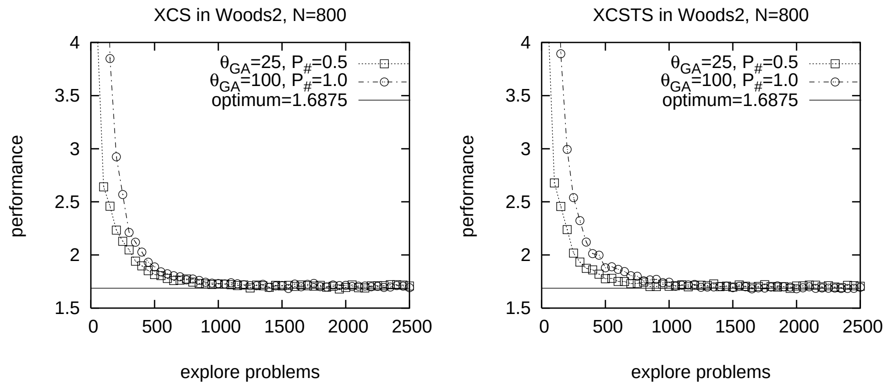

Figure 22. In the simple Woods2 problem, hardly any performance differences are detectable between XCS and XCSTS.

be so easy so that initial parameter settings hardly affect performance. Even with a completely general initial population, XCS as well as XCSTS quickly evolve a Q-function representation and thus find the optimal number of steps to the food.

The Maze6 environment is much more difficult since the maximum distance to the food is much larger (eight). Fewer generalizations are possible. Maze6 previously underwent several investigations and has been shown to be a hard problem for XCS. Lanzi [23] showed that only if an additional specification mechanism is applied in XCS and a sufficiently uniform exploration of the search space is assured (by resetting the problem every 50 steps), XCS can solve the problem with near optimal performance. Figure 23 compares XCS and XCSTS in the Maze6 problem. It can be seen that XCSTS can solve the problem optimally albeit correct parameter settings are necessary. In all settings, though, XCSTS outperforms XCS. Even starting with an initially completely general population, tournament selection is able to detect the accuracy gradient and gradually approaches optimum performance. To ensure fast convergence, the initial population needs to be very specific, population size needs to be set high, and the GA threshold needs to be set high to allow proper parameter estimation. While XCSTS's performance is certainly still not perfect in multistep problems, the comparison confirms that tournament selection is helpful also in multistep problems.

exploration steps. If 100 steps are reached and food is not yet found, 100 steps are recorded (if in an exploit trial) and the next trial begins.

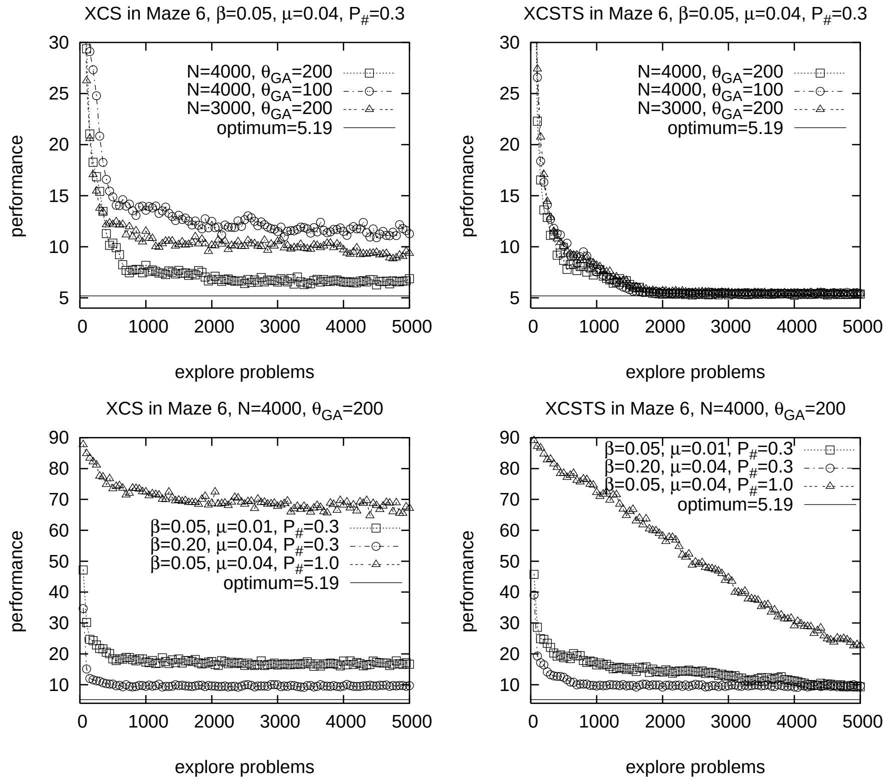

Figure 23. XCSTS can solve the Maze6 problem only with the proper parameter settings. Nonetheless, XCS without tournament selection never reaches optimal performance reliably.

6. Summary and Conclusions

This paper showed that proportionate selection can inhibit sufficient fitness pressure and thus successful learning in XCS. Applying tournament selection results in better parameter independence, stronger noise robustness, and more effective fitness exploitation. We also showed that a tournament size of a fraction $\tau = 0.4$ of the current action set size results in a strong enough selection pressure and leaves enough room for efficient recombinatory events. In essence, no problem was found in which XCS with tournament selection with an action-set-size proportionate tournament size performed worse than proportionate selection. Also in multistep problems, XCSTS outperforms XCS. Thus, it is clear that future research in XCS should start using tournament selection instead of proportionate selection.

The need for a set-size dependent tournament selection mechanism sheds further light on the functioning of XCS. Starting from the over-general side, sets of increasingly specialized classifiers evolve that cover the various subproblems, or *niches*, in the given problem. Once maximum accuracy is reached, the set pressure boils the niches down to the maximal accurate, maximal general classifiers. Subsumption deletion and other techniques like *condensation* [33], which reduces mutation and crossover rates once accuracy is reached, can further increase generality and decrease population size. Due to this adaptive learning mechanism, action set sizes and specificity distributions vary. Thus, proportionate selection as well as tournament selection with a fixed tournament size may be misled. Set-size dependent tournament selection ensures strong selection pressure and still allows effective recombination. While this study focused on the accuracy-based classifier system XCS, the general result shows that, given selection is applied in dynamic subsets of a population, set-size proportionate tournament selection achieves a stronger, more reliable and more stable fitness pressure. The successful application of similar tournament selection mechanisms to other LCSs awaits future research.

Additionally, we showed in Section 5.4 that in the layered count-ones problem crossover can be beneficial in LCSs. Although mutation alone was still able to solve the problem, crossover strongly improved performance. While the effective recombination of *building blocks* has been shown to be crucial for successful learning in many problems in the GA optimization literature [17, 12, 30, 13], in the LCS realm, recombination has never been shown particularly effective. The results in the layered count-ones problem show that the effective recombination of building blocks can be very important also in classification problems. In the layered count ones problem, building blocks are of size one since each specified relevant position increases accuracy independent of the other relevant positions. In general, a building block for XCS may be defined by a subset, or *schema*, of specialized attributes that increases accuracy. It is under investigation, in which problems more advanced recombination operators, such as the diverse linkage-learning techniques in GA optimization [16, 28, 13], may be necessary.

In conclusion, the introduction of tournament selection to XCS is only a first step on the road towards *competent LCSs*. An LCS may be called *competent*, if it is able to solve boundedly difficult, decomposable machine learning problems quickly, accurately, and reliably evolving a maximally accurate, maximally general population of classifiers for a perfect representation of the problem solution. Tournament selection ensures appropriate selection pressure. Once we can also ensure effective building block processing in XCS, the goal to design competent LCSs may be within our grasp.

Acknowledgements

We are grateful to Pier-Luca Lanzi, Xavier Llorà, Martin Pelikan, Abhishek Sinha, and the whole IlliGAL lab for their help and the useful discussions.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-03-1-0129. 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. Additional support from the Computational Science and Engineering graduate option program (CSE) at the University of Illinois at Urbana-Champaign 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 or the U.S. Government.

References

1. Bernadó, E., X. Llorà, and J. M. Garrell: 2002, 'XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001*. Berlin Heidelberg: Springer-Verlag, pp. 115–132.
2. Bernadó-Mansilla, E. and J. M. Garrell-Guiu: 2003, 'Accuracy-Based Learning Classifier Systems: Models, Analysis, and Applications to Classification Tasks'. *Evolutionary Computation* **11**, 209–238.
3. Booker, L. B., D. E. Goldberg, and J. H. Holland: 1989, 'Classifier Systems and Genetic Algorithms'. *Artificial Intelligence* **40**, 235–282.
4. Butz, M. V. and D. E. Goldberg: 2003, 'Bounding the Population Size in XCS to Ensure Reproductive Opportunities'. *Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003)* pp. 1844–1856.
5. Butz, M. V., D. E. Goldberg, and K. Tharakunnel: 2003a, 'Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection, and Bilateral Accuracy'. *Evolutionary Computation* **11**, 239–277.
6. Butz, M. V., T. Kovacs, P. L. Lanzi, and S. W. Wilson: 2001, 'How XCS evolves accurate classifiers'. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)* pp. 927–934.
7. Butz, M. V. and M. Pelikan: 2001, 'Analyzing the evolutionary pressures in XCS'. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)* pp. 935–942.
8. Butz, M. V., K. Sastry, and D. E. Goldberg: 2003b, 'Tournament Selection in XCS'. *Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003)* pp. 1857–1869.

9. Butz, M. V. and S. W. Wilson: 2001, 'An Algorithmic Description of XCS'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Advances in learning classifier systems: Third international workshop, IWLCS 2000*. Berlin Heidelberg: Springer-Verlag, pp. 253–272.
10. De Jong, K. A. and W. M. Spears: 1991, 'Learning concept classification rules using genetic algorithms'. *IJCAI-91 Proceedings of the Twelfth International Conference on Artificial Intelligence* pp. 651–656.
11. Dixon, P. W., D. W. Corne, and M. J. Oates: 2002, 'A preliminary investigation of modified XCS as a generic data mining tool'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001*. Berlin Heidelberg: Springer-Verlag, pp. 133–150.
12. Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.
13. Goldberg, D. E.: 2002, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Boston, MA: Kluwer Academic Publishers.
14. Goldberg, D. E. and K. Deb: 1991, 'A comparative analysis of selection schemes used in genetic algorithms'. *Foundations of Genetic Algorithms* pp. 69–93.
15. Goldberg, D. E. and K. Sastry: 2001, 'A practical schema theorem for genetic algorithm design and tuning'. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)* pp. 328–335.
16. Harik, G.: 1997, 'Learning gene linkage to efficiently solve problems of bounded difficulty'. Ph.D. thesis, University of Michigan. also IlliGAL Report No. 97005, University of Illinois at Urbana-Champaign.
17. Holland, J. H.: 1975, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press. second edition 1992.
18. Holland, J. H.: 1976, 'Adaptation'. In: R. Rosen and F. Snell (eds.): *Progress in theoretical biology*, Vol. 4. New York: Academic Press, pp. 263–293.
19. Kaelbling, L. P., M. L. Littman, and A. W. Moore: 1996, 'Reinforcement Learning: A Survey'. *Journal of Artificial Intelligence Research* **4**, 237–285.
20. Kovacs, T.: 2002, 'XCS's strength based twin: Part I'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Fifth International Workshop on Learning Classifier Systems (IWLCS-2002), Workshop Working Notes*. Granada, Spain:, pp. 59–79.
21. Kovacs, T.: 2003, *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Berlin Heidelberg: Springer-Verlag.
22. Kovacs, T. and M. Kerber: 2001, 'What makes a problem hard for XCS?'. In: P. L. Lanzi, W. Stolzmann, and S. W. Wilson (eds.): *Advances in learning classifier systems: Third international workshop, IWLCS 2000*. Berlin Heidelberg: Springer-Verlag, pp. 80–99.
23. Lanzi, P. L.: 1999a, 'An Analysis of Generalization in the XCS Classifier System'. *Evolutionary Computation* **7**(2), 125–149.
24. Lanzi, P. L.: 1999b, 'An extension to the XCS classifier system for stochastic environments'. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)* pp. 353–360.
25. Lanzi, P. L.: 2000, 'Adaptive agents with reinforcement learning and internal memory'. *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior* pp. 333–342.
26. Lanzi, P. L.: 2002, 'Learning classifier systems from a reinforcement learning perspective'. *Soft Computing: A Fusion of Foundations, Methodologies and Applications* **6**, 162–170.

27. Lanzi, P. L. and S. W. Wilson: 2000, 'Toward optimal classifier system performance in non-Markov environments'. *Evolutionary Computation* **8**(4), 393–418.
28. Pelikan, M., D. E. Goldberg, and F. Lobo: 2002, 'A survey of optimization by building and using probabilistic models'. *Computational Optimization and Applications* **21**(1), 5–20.
29. Sutton, R. S. and A. G. Barto: 1998, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
30. Thierens, D. and D. E. Goldberg: 1993, 'Mixing in genetic algorithms'. *Proceedings of the Fifth International Conference on Genetic Algorithms* pp. 38–45.
31. Venturini, G.: 1994, '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* pp. 371–381.
32. Watkins, C. J. C. H.: 1989, 'Learning from Delayed Rewards'. Ph.D. thesis, King's College, Cambridge, UK.
33. Wilson, S. W.: 1995, 'Classifier Fitness Based on Accuracy'. *Evolutionary Computation* **3**(2), 149–175.
34. Wilson, S. W.: 1998, 'Generalization in the XCS classifier system'. *Genetic Programming 1998: Proceedings of the Third Annual Conference* pp. 665–674.