Studying XCS/BOA Learning in Boolean Functions: Structure Encoding and Random Boolean Functions

Martin V. Butz
Department of Cognitive Psychology
University of Würzburg
97070 Würzburg, Germany
mbutz@psychologie.uni-wuerzburg.de

Martin Pelikan
Dept. of Math and Computer Science, 320 CCB
University of Missouri at St. Louis
St. Louis, MO 63121, USA
pelikan@cs.umsl.edu

ABSTRACT

Recently, studies with the XCS classifier system on Boolean functions have shown that in certain types of functions simple crossover operators can lead to disruption and, consequently, a more effective recombination mechanism is required. Simple crossover operators were replaced by recombination based on estimation of distribution algorithms (EDAs). The combination showed that XCS with such a statistics-based crossover operator can solve challenging hierarchical functions more efficiently. This study elaborates the gained competence further investigating the coding scheme for the EDA component (BOA in our case) of XCS as well as performance in randomly generated Boolean function problems. Results in hierarchical Boolean functions show that the originally used 2-bit coding scheme induces a certain learning bias that stresses additional diversity in the evolving XCS population. A 1-bit coding scheme as well as a restricted 2-bit coding scheme confirm the suspected bias. The alternative encodings decrease the unnecessary bias towards specificity and increase performance robustness. The paper concludes with a discussion on the challenges ahead for XCS in Boolean function problems as well as on the implications of the obtained results for real-valued and multiple-valued classification problems, multi-step problems, and function approximation problems.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Learning Classifier Systems, XCS, Reinforcement Learning, Bayesian Networks

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

1. INTRODUCTION

The XCS classifier system, introduced by Wilson [28], may be the currently most well-understood learning classifiers system (LCS). Recent applications have shown that XCS is able to solve real-world classification problems effectively [1, 2]. Analyses in Boolean function problems suggest that XCS can learn a slightly restricted class of k-DNF problems efficiently and reliably [10, 8], confirming previous conjectures derived from experimental analysis [29].

In hierarchically structured Boolean function problems, however, it was shown that simple crossover operators may disrupt the learning process in XCS. The enhancement of XCS with techniques borrowed from estimation of distribution algorithms (EDAs) [22, 20, 26] showed to prevent disruption and support an efficient propagation of lower level building block structures [11]. Hereby, a Bayesian network is built, which reflects the statistical classifier structure distribution in XCS's population. The model is used to generate or optimize classifier offspring in local problem niches (that is, match sets or action sets).

Despite this first successful and efficient replacement of simple crossover operators with statistics-based recombination operators, it remained unclear, which structural dependencies of the problem are actually modeled by the Bayesian network. Additionally, the structural model was formed using a 2-bit encoding of the ternary classifier condition alphabet (specifying 0, 1, or "don't care"). This two-bit encoding, however, can induce additional diversification and specialization bias in the XCS learning process.

This paper investigates the influence of this encoding further and explores alternatives. First, we experimentally analyze the coding bias showing that it is mainly disruptive, except for in some specific problem types. Thus, alternative encodings are proposed and evaluated on a diverse collection of Boolean functions. The results show that the alternative encodings alleviate the original bias still yielding efficient recombination. Additionally, the performance results suggest that the Bayesian network mainly models the low-order problem dependencies.

Besides the coding issue, it is still under investigation how large the subset of problems is in which efficient recombination is necessary to learn effectively. Thus, we then evaluate XCS on a collection of randomly generated k-DNF problems comparing performance without and with competent crossover application.

2. XCS-BOA IN A NUTSHELL

The XCS classifier system was created by Stewart W. Wilson [28, 29]. XCS is a learning classifier system (LCS) [16, 18] that evolves a set of rules (i.e. a population of classifiers) that represents the problem solution. The solution specifies the expected payoff for each possible action (or classification) given a problem instance. In essence, XCS is designed to evolve a complete, maximally accurate, and maximally general representation of the optimal problem solution. As all Michigan-style LCSs, XCS learns iteratively interacting with an outside environment (such as the classification problem) receiving problem instances, proposing suitable actions (or classifications), and receiving corresponding feedback in the form of real-valued reward.

While XCS was mainly applied to single-step classification problems including real-world datamining problems [1, 2, 6, 9] as well as multi-step problems [7] recent system enhancements have shown that XCS can be modified to efficiently solve function approximation problems [30, 5] as well as approximation problems involving real-valued actions [31]. Thus, XCS is a very flexible learning mechanism for which application success depends on efficient classifier representation, the genetic operators, and the approximation techniques employed.

This section introduces XCS as well as the enhanced XCS version with a recombination mechanism based on the Bayesian optimization algorithm (BOA) [25, 24]. Due to space restrictions, the introduction is brief. The interested reader is referred to the algorithmic description on XCS [12] as well as a detailed treatise on the BOA integration [11].

2.1 XCS Introduction

Since the analysis herein focuses on Boolean classification problems, XCS is introduced as a pure classification system in which no reward propagation is necessary.

2.1.1 Problem Definition

We define a classification problem as a set of problem instances $X = \{0,1\}^l$ with length l. Each problem instance $S \in X$ is thus characterized by l binary features. The target concept assigns each problem instance a corresponding class $A \in \{1, 2, ..., n\}$. Instances are generated at random from X according to some probability distribution D. The investigated problems herein are uniformly distributed over all 2^l possible problem instances. Problem instances are iteratively presented to XCS. In response to the resulting classification, the problem provides scalar reinforcement r reflecting the correctness of the classification. Reward $r = 0$ indicates incorrect classification while a non-null constant reward (here $r = 1000$) indicates correct classification.

2.1.2 Knowledge Representation

Each classifier in the population $[P]$ consists of five major attributes: (1) the condition part C specifies when the classifier matches; (2) the action part A specifies the action (or classification); (3) the reward prediction R estimates the average reward received given conditions C executing action A; (4) prediction error ε estimates the mean absolute deviation of the reward prediction; (5) fitness F estimates the average relative accuracy of the classifier. In the problem setting considered here, conditions are strings of l symbols in the ternary alphabet $\{0, 1, \#\}$ ($C \in \{0, 1, \#\}^l$) where the symbol # (called *don't care*) matches both zero and one.

2.1.3 Classifier Evaluation

Given the current problem instance S, XCS forms a *match set* $[M]$ consisting of all classifiers in $[P]$ whose conditions match S. The match set $[M]$ essentially represents the knowledge about the current problem instance. To decide on the classification, fitness-weighted reward predictions are formed for each possible classification with respect to the classifiers in $[M]$. After the execution of the chosen classification A, and the resulting reward R, an *action set* $[A]$ is formed consisting of all classifiers in $[M]$ that specify the chosen action A. Parameters R, ε, and F of all classifiers in $[A]$ are then updated with an iterative gradient approach using the Widrow-Hoff delta rule [27]. Hereby, the learning rate β controls the speed of parameter adaptation. The reward prediction R essentially estimates the mean reward received when executing specified action A in the problem subspace specified by condition C. The estimated error ε approximates the mean absolute deviation of the reward prediction R. Finally, fitness F estimates the scaled, mean, relative accuracy of the classifier derived from the current reward prediction error ε with respect to competing classifiers.

2.1.4 Rule Evolution

The initial population $[P]$ is usually empty. Given a problem instance and no classifier in $[P]$ matches, XCS applies a covering mechanism that generates a classifier for each possible classification. Covering classifiers match the problem instance and have an average predefined *specificity* $(1 - P_{\#})$.

The genetic algorithm (GA) is the main rule structuring component. Because classifier fitness estimates the accuracy of the reward prediction, the GA favors the evolution of classifiers that provide an accurate prediction of the expected payoffs. The genetic algorithm used is a steady-state niched genetic algorithm [13]. Given the GA is applied in a certain problem iteration (controlled by the GA threshold θ_{GA}), two classifiers are selected from the current action set $[A]$ maximizing fitness. The introduction of tournament selection with a tournament size proportionate to the current action set size strongly increased the noise-robustness of the system [9]. Offspring classifiers are crossed over with probability χ, mutated with probability μ, and inserted in the population. To keep the population size constant, two classifiers are deleted from the population. Additionally, a subsumption-deletion mechanism is applied that favors more general, accurate, reliable (low error ε) classifiers over more specialized offspring classifiers.

2.2 Recombination with Bayesian Networks

For recombination to be useful in XCS, crossover must effectively combine groups of features that represent important sub-structures of the desired maximally accurate, maximally general problem solution. While features may be processable independently in some problems, in other problems it is important that recombination does not destruct important sub-structures but rather detects such sub-structures in order to prevent their disruption and enforce their effective juxtaposition. As an example of such difficult problems, consider hierarchical classification problems in which sub-problems are evaluated on a lower level and their solutions are combined yielding the higher level problem input. Figure 1 shows such a two-level hierarchy with a parity (or XOR) problem at the lower level and the 6-multiplexer problem at the higher level.

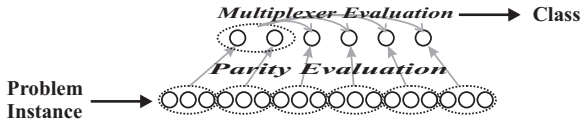

Figure 1: Two level hierarchy demanding efficient recombination in XCS.

Applications of XCS to difficult hierarchical problems showed that simple crossover operators can be disruptive preventing the learning of a successful problem solution [4]. Enhancements show that XCS with the recombination operator based on the extended compact GA [14] or BOA [25] can solve the problems efficiently [4]. Particularly, the enhanced system XCS-BOA builds a Bayesian network model of the currently most successful classifiers in the XCS population. The model is then used to generate or optimize condition and action of classifier offspring, effectively replacing the previously applied simple crossover operators. Since XCS generates classifiers in action sets, the model needs to be adjusted to the probability distribution in the current action set.

Several settings were investigated to generate classifier offspring: First, classifiers may be directly sampled from the model using probabilistic logic sampling (PLS) [15]. In PLS the variables are ordered topologically and values are generated following the topological order. As a result, once the value of a variable x_i is to be generated, its parents Π_i are assured to have been generated already. Thus, the probabilities of different values of x_i can be directly extracted from the conditional probability table for x_i using the known values of Π_i. To ensure that the offspring is biased to the current action set, the model is adjusted by adapting the probability values in the model to the current action set. This is accomplished by selecting (with replacement) a certain number of classifiers (here: 20) from the action set (as for normal offspring selection) and using the probability distribution of these classifiers to adjust the model parameters (conditional and marginal probabilities).

Alternatively, selected offspring may be optimized by using a Markov-chain Monte Carlo method (MCMC) [23] to update the structure of a selected classifier probabilistically based on the model. MCMC applies random bit-flips to the binary representation of the classifier (see below) and determines the likelihood of the classifier structure before and after the flip using the probabilistic model. A flip is maintained probabilistically by a probability of $l_f/(l_f+l_n)$, where l_f denotes the likelihood of the classifier structure after the flip and l_n the one before the flip. To avoid zero likelihoods, all conditional probabilities are linearly normalized to values ranging from 0.05 to 0.95.

Due to the binary encoding of the model, classifiers need to be translated into a binary representation. In [11], each attribute was encoded by two bits: One bit specifies if the attribute is specific (either 0 or 1) or general (#-symbol). In the former case, the second bit specifies the actual classifier value. For example, the classifier condition $C_1 = 0\#\#$ may be encoded by `001011` where the first two bits specify that the first condition attribute is 0 and the third and fifth bits specify that the second and third condition attributes are don't cares. The values of the fourth and sixth bits are essentially meaningless and are chosen uniformly randomly.

While this coding choice emphasizes specificity, another coding choice recently proposed in [21] rather emphasizes the values of an attribute: Two bits encode an attribute, each bit stands for one possible value of the attribute (that is, 0 or 1) and accepts the value if set. Thus, two ones represent a #-symbol whereas two zeros are meaningless.

3. CODING CONSIDERATIONS AND VARIATIONS

We now consider several alternative encodings and the resulting offspring generation biases.

3.1 Learning Disruption with 2-Bit Encoding

The chosen two-bit encoding induces a certain offspring bias in XCS classifiers. In fact, the encoding enables the generation of offspring that does not match the current action set. Assuming that the model probabilities are updated for the problem instance 011 using the two (matching) classifiers with conditions $C_1 = 0\#\#$, encoded by e.g. `001011` and $C_2 = 01\#$, encoded by e.g. `000111`, offspring may be generated encoded by `000011`, which translates into condition $C_o = 00\#$. The random encoding of the # symbol in C_1 causes the generation of a zero in the second attribute of C_o so that the offspring does not match the current problem instance. This cannot happen with any simple crossover operator. The consequent potentially overly diverse populations may result in learning disruption.

Such disruption becomes imminent in the 37-multiplexer problem. The multiplexer problem is a problem in which k address bits encode the solution position in the 2^k remaining value bits of the problem. In the 37-multiplexer problem, $k = 5$. Figure 2 compares XCS with simple uniform crossover with XCS/BOA with offspring sampling (setting 50/0 - selecting 50 classifiers to set the local probabilities in the Bayesian model) and offspring adaptation (setting 10/18 - selecting 10 classifiers to set local probabilities and trying 18 random bit flips). The results show that when using a two-bit encoding, XCS/BOA is outperformed by XCS with simple crossover. Note that the 10/18 setting is much less disruptive since hereby the induced diversification bias is much weaker.

3.2 Non-disruptive Encodings

We now investigate several restricted encodings, in which the recombination operator cannot generate offspring that does not match the current problem instance.

3.2.1 1-Bit Specific/General Encoding

Instead of giving XCS a choice of which value a specified condition attribute should take, we can tightly constrain that value to the current value in the problem instance. Thus, each condition attribute may be simply coded by one bit that indicates if the attribute is specific or general. If the offspring code generated by sampling from the Bayesian model, or applying MCMC to a selected offspring classifier, specifies a specific attribute, then the attribute of the classifier simply is set to the corresponding attribute in the current problem instance. For example, given problem instance 0110, and classifier offspring code `1100` (where 1 specifies don't care), the corresponding unique classifier condition is ##10.

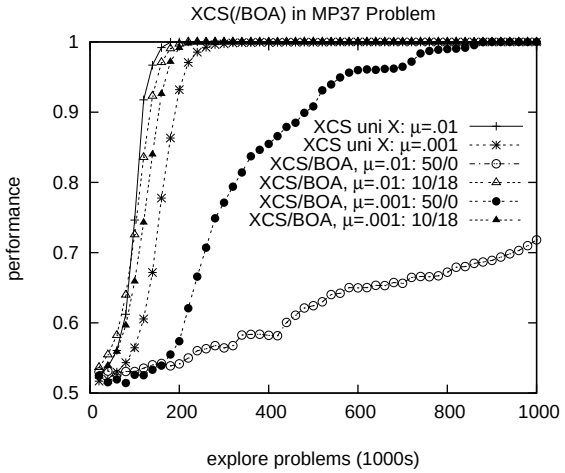

Figure 2: The 2-bit encoding of the classifier model may lead to learning disruption.

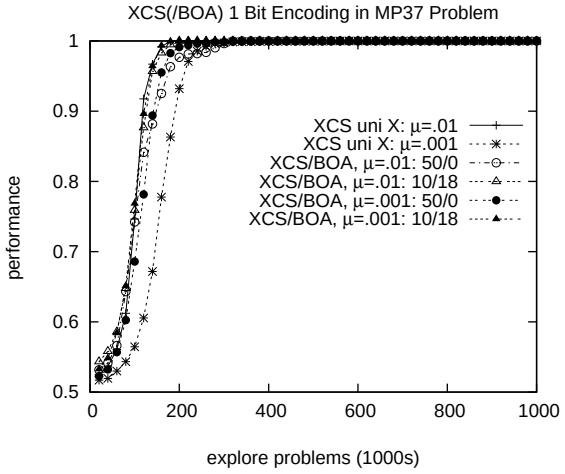

Figure 3: The 1-bit encoding prevents the encountered disruption.

Figure 3 shows performance with this encoding in the 37 multiplexer problem comparing XCS/BOA with XCS with uniform crossover. XCS/BOA now performs equally well.

3.2.2 Restricted 2-Bit Encoding

The 1-bit encoding disables the representation of dependencies between actual classifier values instead of classifier specificities (or generalities, that is, don't care symbols). It may be that a 2-bit encoding may be even more useful for the evolutionary progress. Thus, we re-use the same 2-bit encoding as before but forbid offspring generation that does not match the current problem instance. If such an offspring is generated by the model, then the mismatching attributes are flipped to the other binary value. For example, if the 2-bit encoding generates offspring code `000011` given problem instance 011, the resulting condition will be set to 01#.

Figure 4 shows that the restricted 2-bit encoding again does not cause the previously observed disruption. However, performance does not improve in comparison to the

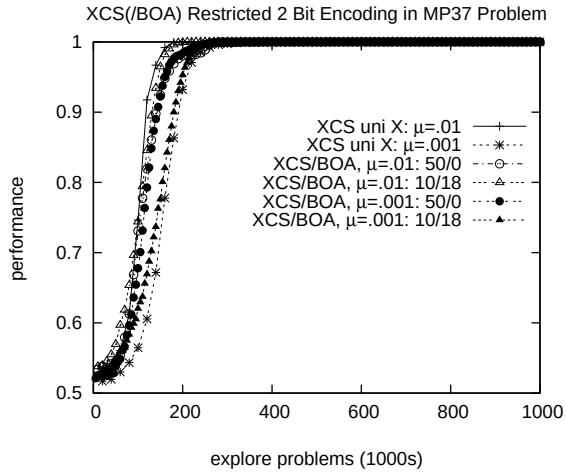

Figure 4: The restricted 2-bit encoding prevents the encountered disruption.

1-bit encoding. The additional bit is of no use in the 37 multiplexer problem. Nonetheless, performance of the simple crossover mechanism is matched.

3.2.3 Further Multiplexer Results

Besides the performance curves, population sizes (the number of macro-classifiers, that is, different classifiers in the population) can reveal how well the population is compressed respecting the optimal solution. Figure 5 shows the population sizes of XCS with uniform crossover as well as the diverse codings within XCS/BOA. It can be seen that very early in the run, the XCS/BOA application results in a more diverse population regardless of the encoding used. However, the 1-bit and restricted 2-bit encodings enable XCS/BOA to converge quickly to the optimal solution. Hereby, the differences between the two encodings are marginal. In the unrestricted 2-bit encoding diversity remains larger and convergence is slower, which indicates the suspected disruptive effects caused by the encoding.

As a final challenge, we compared the various XCS/BOA combinations in the 70 multiplexer problem ($k = 6$). Figure 6 suggests that with increasing population size, the 1-bit encoding becomes increasingly more effective. Early in the run, the 1-bit encoding provides a more directed evolutionary pressure towards more accurate classifiers - most likely propagating the specialization of the k address bits. Late in the run, though, when the dependency of the address bit values and the corresponding address location become more important for the successful detection of the accurate, maximally general classifiers, the restricted 2-bit encoding enables faster convergence.

4. HIERARCHICAL CLASSIFICATION PROBLEMS

The original motivation for the introduction of the competent crossover operator was due to the observation that simple crossover operators can be disruptive in hierarchical classification problems [11]. It was shown that model-based recombination operators can solve these challenging problems. What remains to be shown is whether the model re-

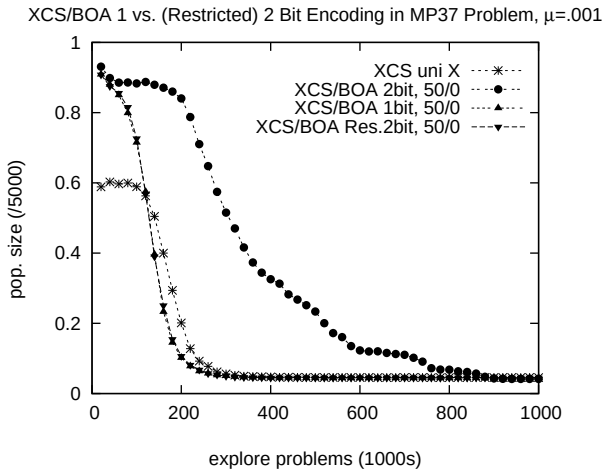

Figure 5: Population sizes show that in the 2-bit classifier encoding increased diversity causes disruption. Albeit all XCS/BOA versions initially yield a much higher diversifying recombination bias, the prevention of offspring generations that do not match the current niche cause a fast decrease in population size indicating successful convergence to the optimal solution.

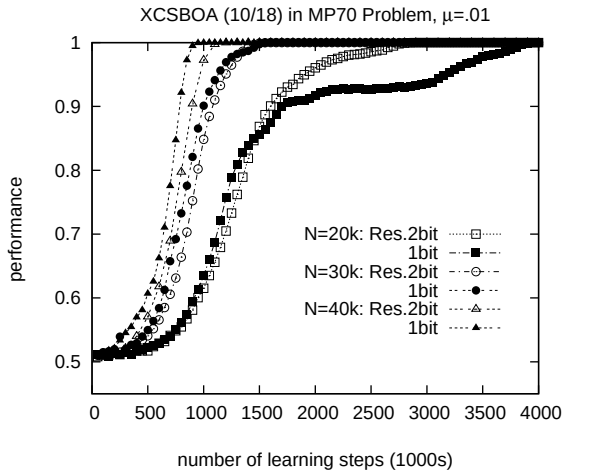

Figure 6: The Bayesian population model propagates the evolution of an optimal solution in the 70 multiplexer. While the 1-bit encoding focuses performance faster, convergence is slightly sped up by the additional information available to the 2-bit encoding case.

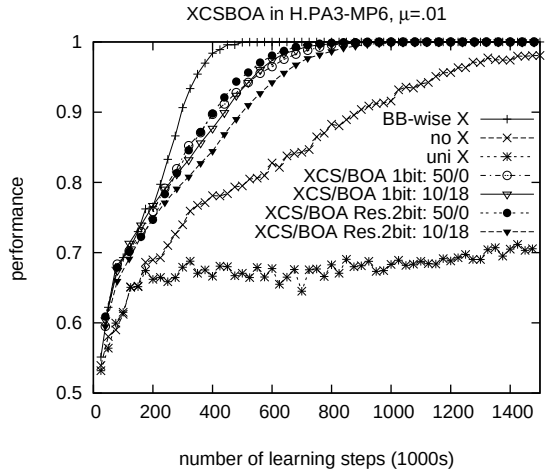

Figure 7: In the hierarchical 3-parity, 6-multiplexer problem, the 1-bit and restricted 2-bit encodings learn competitively to the problem-structure informed building block-wise uniform crossover (BB-wise X) outperforming runs without crossover as well as the highly disruptive uniform crossover.

ally supported the evolutionary process or, accidentally, the 2-bit encoding yielded a sufficient bias in the right direction on its own.

Figure 7 shows that also when using the 1-bit and restricted 2-bit encodings XCS/BOA learns competitively in the hierarchical 3-parity, 6-multiplexer problem. compared to the parity-block-structure informed building block-wise uniform crossover (BB-wise X). Although performance is not quite matched, system behavior shows that the model must have detected and recombined problem substructures. Runs without recombination are clearly outperformed as well as runs with uniform crossover, which is highly disruptive in these problems.

Runs with an even lower mutation rate ($\mu = .001$) show that the lack of diversification and the resulting specialization pressure is missing in the system, delaying learning (Figure 8). The problem is that the diversity in the population is not high enough and mutation alone is too weak to introduce sufficient specialization/diversification pressure. Thus, the supply of lower order building blocks (here: parity blocks) is insufficient for fast learning essentially facing the *schema challenge* [10].

To further determine if XCS/BOA is able to exploit the additional value information contained in the restricted 2-bit encoding, we performed runs in the x,y-biased multiplexer introduced in [10]. This function combines multiple multiplexer functions by having one multiplexer function with x address bits decide on which biased multiplexer with y address bits is executed. The challenge in this function is that each multiplexer needs to be learned independently from each other. The x address bits decide which one of the biased multiplexers is currently relevant. Although one might suspect that the additional information contained in the 2-bit encoding is beneficial in this case, the results show that actually the 1-bit coding yields superior performance in this problem (Figure 9). The 2-bit encoding seems to prevent an efficient coding of the dependencies, possibly also due to

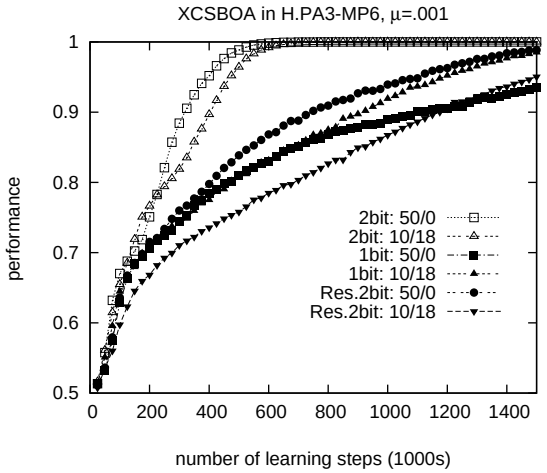

Figure 8: The additional diversification pressure induced by the original 2-bit encoding is slightly advantageous in hierarchical problems.

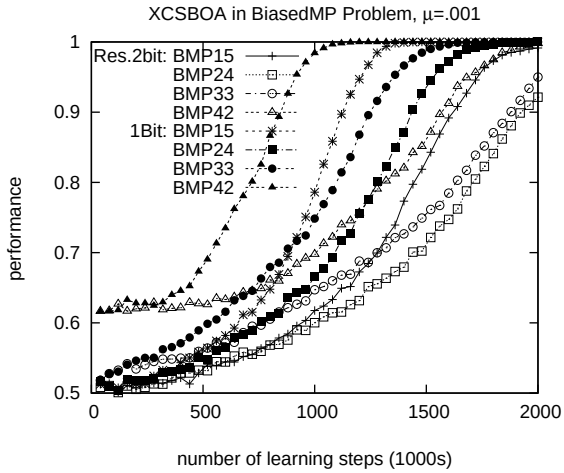

Figure 9: XCS/BOA with low mutation rate ($\mu = .001$) requires sufficient diversification resulting in a better performance for the 1-bit case.

the size of the model. The major drawback here, however, seems to be the lack of diversification in the population. The algorithm only executes 18 random bit flips which might not be enough in these biased multiplexer problems where the problem lengths range from $l = 73$ up to $l = 84$. Within the restricted 2-bit encoding, 18 flips actually affect only 9 bits (on average) providing an alternative explanation for the observed decrease in learning efficiency.

5. RANDOM K-DNF PROBLEMS

Apart from the typically investigated multiplexer and hierarchical classification problems, the question arose how well XCS will do in general kDNF problems. Trusting XCS theory, we know that XCS is able to evolve a complete and accurate problem solution with a population size and iteration time requirement that is polynomial in problem complexity as long as sufficient fitness guidance is available and the solutions are sufficiently non-overlapping [8].

Thus, we now evaluate XCS and XCS/BOA performance on randomly generated kDNF problems. We are interested in whether XCS is able to evolve a near-complete problem solution reliably. Moreover, we are interested in comparing the learning speed and accuracy of different XCS variants in the randomly generated problems. Finally, we are interested in the influence of crossover and the model-building-based recombination operator.

Table 1 shows performance of XCS without and with uniform crossover as well as XCS/BOA with 1-bit and (unrestricted) 2-bit encoding, sampling from the probabilistic model (setting 20/0) or applying MCMC to reproduced offspring (setting 20/10). The performance deviations show that the ten randomly generated problems are equally difficult for XCS. Deviations are small. A control of the maximum and minimum performance values (not shown) confirm that none of the maxima/minima deviated from the average performance values more than three standard deviations. Thus, the randomly generated kDNF problems are equally difficult to XCS.

Comparisons between the different settings show that in these randomly generated problems crossover is not necessarily useful. In fact, XCS with uniform crossover performs worse than XCS without crossover indicating the disruptive effects of recombination. Further analysis of classifier populations showed that although the specificity in the population is generally lower with crossover application, the population size (number of different classifiers) is actually larger. This confirms the hypothesized crossover disruption: Simple uniform crossover tends to combine overlapping clauses in the kDNF problem. The combination yields a partially accurate classifier that results in a blow up of population size but hinders the evolution of maximally accurate classifiers. In the XCS/BOA settings, the 1-bit encoding is again superior to the (unrestricted) 2-bit encoding. Moreover, MCMC yields higher learning performance preventing disruption and propagating important classifier substructures. The setting with larger population size ($N = 8000$) shows that a larger population size clearly beats all other settings in performance, confirming that reproductive opportunities and even more so solution sustenance are a challenging problem in these highly overlapping kDNF problems.

6. SUMMARY AND CONCLUSIONS

This study showed that the substitution of simple crossover operators with enhanced statistics-based recombination operators yields robust learning in diverse Boolean function problems. Even in randomly generated k-DNF problems accurate performance was achieved in a wide range of settings. However, we also saw that in larger random k-DNF problems, additional niching techniques may be required to ensure complete convergence to the accurate, maximally general target solution (represented in the optimal population [O] in Kovacs' terms [19]).

The replacement of the simple crossover with building and sampling probabilistic models showed to be particularly useful in hierarchical problems. The additional bias induced by the originally used unrestricted 2-bit coding of classifier condition attributes showed to be useful in highly symmetrically structured Boolean function problems, such as the parity problem, and hierarchical problems involving parity blocks. However, the encoding can be disruptive in other

Av.& Dev. Perf. in 10 Random kDNF Problems with l=20, k=5, and 22 clauses										
5	**10**	**20**	**40**	**60**	**80**	**100**	**120**	**150**	**200**	**steps (1000s)**
N=4000: XCS/BOA, 1-bit, Setting 20/0										
0.63070	0.67776	0.70678	0.76164	0.80792	0.84224	0.86396	0.88430	0.90000	0.91710	Averages
0.02158	0.01890	0.01986	0.02272	0.02035	0.02530	0.02195	0.02180	0.01864	0.01733	Deviations
N=4000: XCS/BOA, 1-bit, Setting 20/10										
0.63858	0.69306	0.74166	0.82906	0.88094	0.90612	0.92492	0.93386	0.94572	0.95390	Averages
0.02078	0.02112	0.01877	0.01424	0.01597	0.01437	0.01170	0.01014	0.01242	0.00727	Deviations
N=4000: XCS/BOA, 2-bit, Setting 20/0										
0.62806	0.66512	0.67890	0.69394	0.71676	0.73298	0.75060	0.77046	0.78650	0.81458	Averages
0.02257	0.01834	0.01823	0.01686	0.02115	0.02049	0.02083	0.02253	0.02451	0.02982	Deviations
N=4000: XCS/BOA, 2-bit, Setting 20/10										
0.63170	0.68734	0.73666	0.81720	0.86752	0.89868	0.91854	0.93058	0.94086	0.95210	Averages
0.02506	0.01872	0.02140	0.01525	0.01804	0.01388	0.01381	0.01515	0.01492	0.01290	Deviations
N=4000: XCS with uniform Xover										
0.63768	0.68434	0.73352	0.80328	0.85572	0.88168	0.90058	0.91050	0.92732	0.93842	Averages
0.02392	0.01964	0.01865	0.01917	0.01957	0.01983	0.01767	0.01496	0.01036	0.00980	Deviations
N=4000: XCS without Xover										
0.62228	0.67842	0.74496	0.82151	0.87273	0.90762	0.92375	0.93486	0.94589	0.95392	Averages
0.01924	0.02351	0.01818	0.01476	0.01253	0.01287	0.01050	0.00927	0.00780	0.00883	Deviations
N=8000: XCS/BOA, 1-bit, Setting 20/10										
0.63580	0.69834	0.79575	0.91700	0.95997	0.97293	0.97907	0.98185	0.98277	0.98189	Averages
0.01921	0.01810	0.00937	0.00867	0.00748	0.00464	0.00378	0.00401	0.00249	0.00342	Deviations

Table 1: In randomly generated kDNF problems, crossover is mainly disruptive. Proper XCS/BOA combinations are as effective as runs with mutation only. Different problem instances appear equally difficult for XCS seeing the low standard deviation values.

Boolean functions. The proposed specificity-focused 1-bit encoding or the 2-bit encoding with the restriction of generating matching offspring yield more robust performance. The marginal performance differences between 1-bit and 2-bit encoding additionally suggest that the model mainly encodes lower level dependencies focused on specificity rather than on the actual values of specified condition attributes. Thus, a 1-bit specific/general encoding of classifier conditions may be sufficient for an efficient recombination operator in XCS.

With respect to the slightly inaccurate results in the investigated k-DNF problems, it should be kept in mind that XCS is designed to evolve a complete problem solution by specifying classifications for both sides: correct classifications and incorrect classifications. In this sense, XCS does not only strive to learn the clauses of a k-DNF problem but also the potentially much more complex subspace of the negation of the clauses. Thus, the problem XCS learns is essentially harder. Other approaches using a default rule [17] for the incorrect classifications may be worth exploring. Similarly, advancements within the ZCS system [28, 3] may yield a more robust solution than XCS since ZCS only learns the correct classifications (dependent on the reward scheme used). Future research needs to shed further light on the niching constraints in XCS induced by the fitness sharing mechanism as well as the action set-based reproduction.

Acknowledgments

This work was support by the European commission contract no. FP6-511931 is acknowledged. Additional support came from the National Science Foundation under NSF CAREER grant ECS-0547013 and the University of Missouri under the Research Board and Research Award programs.

7. REFERENCES

[1] E. Bernadó, X. Llorà, and J. M. Garrell. 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, editors, *Advances in Learning Classifier Systems (LNAI 2321)*, pages 115–132. Springer-Verlag, Berlin Heidelberg, 2002.

[2] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis, and applications to classification tasks. *Evolutionary Computation*, 11:209–238, 2003.

[3] L. Bull and J. Hurst. ZCS redux. *Evolutionary Computation*, 10(2):185–205, 2002.

[4] M. V. Butz. Anticipation for learning, cognition, and education. *On the Horizon*, 2004. in press.

[5] M. V. Butz. Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 2*, pages 1835–1842, 2005.

[6] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Bounding learning time in XCS. *Proceedings of the Sixth Genetic and Evolutionary Computation Conference (GECCO-2004): Part II*, pages 739–750, 2004.

[7] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Gradient Descent Methods in Learning Classifier Systems: Improving XCS Performance in Multistep

Problems. *IEEE Transactions on Evolutionary Computation*, 9:452– 473, 2004.

[8] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. *Foundations of Learning Classifier Systems*, chapter Computational Complexity of the XCS Classifier System, pages 91–126. Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg, 2005.

[9] M. V. Butz, D. E. Goldberg, and K. Tharakunnel. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11:239–277, 2003.

[10] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. Toward a theory of generalization and learning in XCS. *IEEE Transactions on Evolutionary Computation*, 8:28– 46, 2004.

[11] M. V. Butz, M. Pelikan, X. Llorà, and D. Goldberg. Extracted global structure makes local building block processing effective in XCS. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 1*, pages 655–662, 2005.

[12] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996)*, pages 253–272. Springer-Verlag, Berlin Heidelberg, 2001.

[13] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[14] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL report 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1999.

[15] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 149–163. Elsevier, Amsterdam, London, New York, 1988.

[16] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in theoretical biology*, volume 4, pages 263–293. Academic Press, New York, 1976.

[17] J. H. Holland. *Hidden Order: How Adaptation Builds Complexity*. Perseus Book, 1995.

[18] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*, pages 313–329. Academic Press, New York, 1978.

[19] T. Kovacs and M. Kerber. What makes a problem hard for XCS? In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996)*, pages 80–99. Springer-Verlag, Berlin Heidelberg, 2001.

[20] P. Larrañaga. A review on estimation of distribution algorithms. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms*, chapter 3, pages 57–100. Kluwer Academic Publishers, Boston, MA, 2002.

[21] X. Llorà, K. Sastry, and D. E. Goldberg. The compact classifier system: Motivation, analysis, and first results. *GECCO 2005: Genetic and Evolutionary Computation Conference*, pages 1893–1894, 2005.

[22] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, 1996.

[23] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[24] M. Pelikan. *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer-Verlag, 2005.

[25] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. BOA: The Bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 525–532, 1999.

[26] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[27] B. Widrow and M. Hoff. Adaptive switching circuits. *Western Electronic Show and Convention*, 4:96–104, 1960.

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

[29] S. W. Wilson. Generalization in the XCS classifier system. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674, 1998.

[30] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1:211–234, 2002.

[31] S. W. Wilson. Classifier systems for continuous payoff environments. *Proceedings of the Sixth Genetic and Evolutionary Computation Conference (GECCO-2004): Part II*, pages 824–835, 2004.