Extracted Global Structure Makes Local Building Block Processing Effective in XCS

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

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

Xavier Llorà
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
xllora@illigal.ge.uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
deg@illigal.ge.uiuc.edu

ABSTRACT

Michigan-style learning classifier systems (LCSs), such as the accuracy-based XCS system, evolve distributed problem solutions represented by a population of rules. Recently, it was shown that decomposable problems may require effective processing of subsets of problem attributes, which cannot be generally assured with standard crossover operators. A number of competent crossover operators capable of effective identification and processing of arbitrary subsets of variables or string positions were proposed for genetic and evolutionary algorithms. This paper effectively introduces two competent crossover operators to XCS by incorporating techniques from competent genetic algorithms (GAs): the extended compact GA (ECGA) and the Bayesian optimization algorithm (BOA). Instead of applying standard crossover operators, here a probabilistic model of the *global* population is built and sampled to generate offspring classifiers *locally*. Various offspring generation methods are introduced and evaluated. Results indicate that the performance of the proposed learning classifier systems XCS/ECGA and XCS/BOA is similar to that of XCS with informed crossover operators that is given all information about problem structure on input and exploits this knowledge using problem-specific crossover operators.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.5.1 [**Pattern Recognition**]: Models—*Statistical; Structural; Neural Nets*

GECCO'05, June 25–29, 2005, Washington, DC, USA.

General Terms

Algorithms, Experimentation, Performance, Reliability, Theory

Keywords

Learning Classifier Systems, Building Block Processing, Decomposable Classification Problems, Estimation of Distribution Algorithms, Extended Compact GA, Bayesian Optimization Algorithm

1. INTRODUCTION

Standard variation operators in genetic algorithms (GAs), such as crossover and mutation, may lead to highly inefficient evolutionary search due to their potential disruptive effects. One of the biggest challenges in genetic and evolutionary computation is the design of techniques that can adapt to a problem and improve effectiveness of variation operators. Estimation of distribution algorithms (EDAs) [24, 21, 30] use probabilistic variation operators to identify and exploit problem structure effectively. The use of advanced variation operators such as those in ECGA [13] and BOA [29, 28] provides scalable solutions for decomposable problems of bounded order. These problems are intractable using standard variation operators [12].

Most LCSs use genetic variation operators so that the lessons learned from research in genetic and evolutionary computation should carry over to LCSs. Despite impressive results achieved with LCSs, research on LCSs has hardly addressed the issue of recombination operators. This papers replaces the potentially ineffective crossover operators with competent crossover operators that adapt to the identified problem structure on the fly.

Here we focus on the well-studied XCS classifier system [37]. In XCS, the accuracy-based fitness results in an evolutionary pressure towards higher accuracy and thus higher specificity [7]. It was shown that for problems where recombination of individual problem attributes improves accuracy, uniform crossover can ensure successful learning [6].

This paper first characterizes BB-hard problems in LCSs, and the XCS system in particular. We then endow XCS with mechanisms adopted from either ECGA [13] or BOA [29, 28]. In difference to previous ad-hoc substitutions of crossover with EDA mechanisms [32], the additional mechanisms here are used to build a probabilistic model of the *global* population. The model is then continuously adjusted to the local distribution to sample local offspring. The results confirm that XCS combined with ECGA or BOA identifies and propagates BB structures effectively reaching performance comparable to an informed BB processing mechanism (that is, BB-wise uniform crossover).

The remainder of this paper first summarizes XCS and some important XCS theory. Section 3 introduces several hierarchical problems and indicates the need for more competent recombination operators. Section 4 combines XCS with mechanisms from ECGA and BOA. Section 5 presents experimental results, which confirm the validity of the approach. Section 6 concludes the paper.

2. BRIEF OVERVIEW OF XCS

Like other LCSs [17, 2], XCS evolves a set of condition-action rules, that is, a *population of classifiers*. Classifiers evolve using a steady-state, niched GA. The fitness in the XCS classifier system is based on the accuracy of reward predictions rather than on the reward predictions themselves. Thus, XCS is designed to evolve not only the representation of an optimal behavioral policy or classification, but rather to evolve a representation of the expected payoff in each possible situation-action combination.

XCS represents its knowledge using a fixed-length population $[P]$ of classifiers. Each classifier consists of five main components. The condition part specifies in which situations the classifier is applicable. The action part determines its classification or action. The payoff prediction part estimates the average expected reward. The prediction error part determines the mean absolute error of the payoff prediction. Finally, the fitness part reflects the current relative accuracy of the classifier with respect to competing classifiers.

Learning usually starts with an empty population. Given the current input consisting of the values of all attributes, a match set $[M]$ is formed that consists of all classifiers in $[P]$ whose conditions match the input. If an action is not represented in $[M]$, a covering mechanism is applied. Given a match set, XCS estimates the payoff for each possible action and computes a prediction array $P(A)$ that reflects the fitness-weighted averages of the matching classifiers' reward predictions for each action. 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. All classifiers in $[M]$ that specify the chosen action comprise the action set $[A]$.

XCS iteratively updates $[P]$ with respect to the successive problem instances. After the classification is selected and executed, the problem provides scalar feedback. Classifier parameters are updated in $[A]$ with respect to the immediate feedback. After rule evaluation and possible GA invocation, the next iteration starts.

The aforementioned covering mechanism ensures that all actions in a particular problem instance are represented by at least one classifier. XCS applies a GA for rule evolution. The GA selects two parental classifiers from the current action set using set-size relative tournament selection [8]. Two offspring are generated by applying crossover and mutation to the two selected parents. Parents remain in the population, competing with their offspring. 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. 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.

XCS strives to evolve a complete, accurate, and maximally general problem solution represented by maximally accurate classifiers. Each classifier specifies a solution (that is, a class) for the problem subspace defined by its condition part. In combination, the population evolves a complete problem solution.

Hereby, the combination of niche reproduction with population-wide deletion results in a distributed, local search for the global problem solution. Locally in each action set, XCS searches for substructures that yield maximal accuracy. Globally, the combination of the identified substructures forms a global classification (or payoff prediction) landscape. Thus, crossover needs to combine globally effective substructures locally searching for more accurate substructures. Simple crossover can only re-sample the current local classifier distribution. Mutation (as long as general mutation is applied) may extend the current structural information to other syntactically neighboring problem subspaces.

3. XCS'S SEARCH FOR STRUCTURE

Except for in very small problems, XCS may only begin its evolutionary search with very general classifier conditions that are then specialized via mutation and classifier recombinations to evolve a complete, maximally accurate problem solution.[1] Specializations that yield higher accurate classifiers are then propagated via reproduction. Thus, XCS evolves progressively higher accurate classifiers as long as the problem structure provides sufficiently accurate information. The following examples clarify XCS search and introduce BB-hard classification problems.

The multiplexer problem [10, 37, 38] is a suitable problem to illustrate accuracy levels and consequent problem search in XCS. Table 1 shows some exemplar classifier condition parts and the corresponding average reward prediction and reward prediction error estimates for classifiers with action part 1. Accuracy somewhat guides towards the correct specializations. Initially, though, only the specialization of the value bits increases accuracy. Once some value bits are specialized in a condition, specialization of the address bits decreases accuracy further.When starting with complete generality, relying on mutation for the first specializations, specificity initially raises more in the value attributes of the classifiers. Only later, specificity in the address attributes takes over [6].

The hidden parity problem [18] is harder than the multiplexer problem because only once all relevant attributes are specialized, accuracy increases. Thus, we need to ensure a sufficient initial supply of BBs, which in this case consist of classifiers that specialize all parity bits [6]. This is similar to

[1]Several mechanisms in XCS additionally assure the search for a maximally general problem solution [37, 7]. This feature of XCS is not further investigated in this paper but explains the found highly general problem solutions.

Table 1: Expected reward prediction and reward prediction error measures for exemplar condition parts in several typically-used Boolean function problems for classifiers with action part $A = 1$.

6-Multiplexer Problem			Hidden 4-Parity Problem		
C	p	ϵ	C	p	ϵ
`######`	500.0	500.0	`#####`	500.0	500.0
`1#####`	500.0	500.0	`1####`	500.0	500.0
`0#####`	500.0	500.0	`0####`	500.0	500.0
`##1###`	625.0	468.8	`11###`	500.0	500.0
`##0###`	375.0	468.8	`1##1#`	500.0	500.0
`##11##`	750.0	375.0	`00###`	500.0	500.0
`##00##`	250.0	375.0	`111##`	500.0	500.0
`0#1###`	750.0	375.0	`000##`	500.0	500.0
`0#0###`	250.0	375.0	`101##`	500.0	500.0
`0#11##`	1000.0	0.0	`1110#`	1000.0	0.0
`001###`	1000.0	0.0	`0100#`	1000.0	0.0
`10##1#`	1000.0	0.0	`0000#`	0.0	0.0
`000###`	0.0	0.0	`1010#`	0.0	0.0
`01#0##`	0.0	0.0	`1111#`	0.0	0.0

the needle-in-the-haystack problem studied in optimization where all solutions are equal except for the global optimum. Table 1 shows the four hidden parity problem (the fifth bit is irrelevant). Error only drops to zero once all four attributes are specified.

The two problems illustrate that the existence of accuracy guidance, that is, substructures that yield progressively higher accuracy, depends on the problem structure. As exhibited in the hidden-parity problem, sometimes it might not be sufficient to specify one attribute to gain accuracy but several attributes may need to be specified. In this case, learning time becomes polynomial in the number of irrelevant attributes where the order of the polynomial equals the number of attributes that need to be specified at once [6, 5]. Thus, learning may be significantly delayed.

Even worse, what if several of those accuracy blocks need to be combined to reach optimal performance? In this case, many subsolutions need to be maintained in parallel. Mutation alone, however, would not be able to combine subsolutions but would need to detect the accuracy blocks rather independently for each subsolution. Thus, mutation can be expected to take a very long time until the complete solution is found. Only a recombination operator that combines substructures effectively can improve learning speed in such problems.

To illustrate the necessity of block processing, we define a problem structure constructing a two-level hierarchy as has been done for standard GAs and EDAs [36, 28]. On the lower level, small-order Boolean functions are evaluated to provide the input to the higher level. The higher level is evaluated using the output of these functions as input. As an example, we can consider a parity, multiplexer combination in which the lower-level blocks are evaluated by the parity function. The results of the parity functions are then fed into the higher-level multiplexer function, which determines the overall class of the problem instance.

Note that we are not interested in creating a problem to force BB processing for its own sake. In fact, many indications in nature and engineering suggest that typical natural problems are structured in a hierarchical, decomposable structure [35, 11, 12]. Similarly, in the world of classification and prediction, interacting factors can be expected to determine the result where the factors should be expected to be modularized as well. Thus, we believe that the introduced class of problems is an important problem class and should be solvable by a rather general machine learning system such as the XCS system.

Table 2: Expected reward prediction and reward prediction error measures for exemplar condition parts in the hierarchical 3-parity / 6-multiplexer problem for classifiers with action part $A = 1$. For readability reasons, the lower level 3-parities are tightly coded and separated by spaces.

C	p	ε
`### ### ### ### ### ###`	500.0	500.0
`111 ### ### ### ### ###`	500.0	500.0
`#1# ### #11 #1# #11 ###`	500.0	500.0
`### ### 111 ### ### ###`	625.0	468.8
`### #1# ### 100 ##1 ###`	625.0	468.8
`### 0## ### ### 000 ###`	375.0	468.8
`### 111 ### 010 ### ###`	750.0	375.0
`##1 111 ##0 100 #0# ###`	750.0	375.0
`101 ### 111 ### ### ###`	750.0	375.0
`### 000 ### ### 000 ###`	250.0	375.0
`101 111 ### 100 ### ###`	1000.0	0.0
`101 000 111 ### ### ###`	1000.0	0.0

How can XCS solve this problem? Clearly, the lower level parity blocks need to be identified first to enable the discovery of the higher level function. Table 2 shows exemplar conditions with corresponding average reward predictions and prediction errors for the hierarchical 3-parity, 6-multiplexer problem. In contrast to the plain multiplexer problem or count ones problem, in these hierarchical problems, the lower-level BBs (here, for example, parity blocks) need to be identified and then processed effectively. The next section shows that only if the detected blocks are not disrupted, XCS is able to solve the problem. Additionally, only if the BBs are recombined effectively, XCS can solve the problem efficiently.

In the remainder of this paper we focus on XCS performance in the parity, multiplexer combination. Nonetheless, any other type of Boolean function combination in the proposed hierarchical manner is possible. Additionally, it is not necessary that all BBs on the lower level are evaluated by the same Boolean function nor do they need to be of equal length. Certainly, though, all these potential manipulations may lead to different population size and learning-time requirements as outlined elsewhere [4].

Exemplar performance of XCS in the hierarchical 3-parity, 6-multiplexer problem is shown in Figure 1.[2] It can be seen that XCS is not able to solve the problem if uniform

[2]If not stated differently, all results are averaged over ten experiments. Performance is assessed by test trials in which no learning takes place and the better classification is chosen. During learning, classifications are chosen at random. If not stated differently, parameters were set as follows: $N = 20000$, $\beta = 0.2$, $\alpha = 1$, $\varepsilon_0 = 10$, $\nu = 5$, $\theta_{GA} = 25$, $\chi = 1.0$, $\mu = 0.01$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = 20$, and $P_{\#} = 0.6$ (see e.g. [4] for details). GA subsumption was applied.

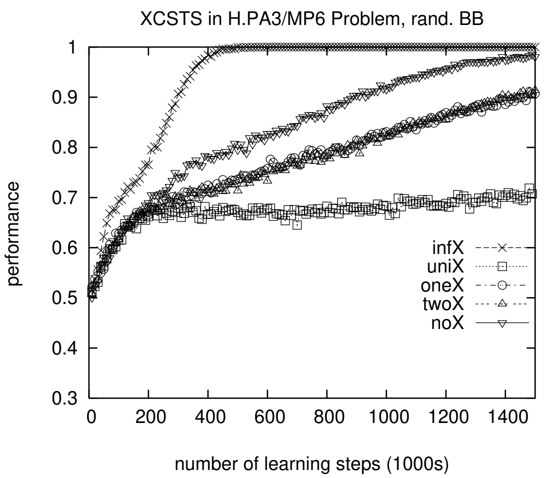

Figure 1: In the loosely coded hierarchical 3-parity, 6-multiplexer problem all simple crossover operators cause disruption (uniX= uniform crossover, oneX = one-point crossover, twoX = two-point crossover). Mutation alone is able to solve the problem but performance is strongly delayed (noX = mutation only). Only BB-wise uniform crossover (infX = informed crossover) solves the problem quickly, reliably, and effectively.

crossover is applied. Due to the disruptive effects of uniform crossover—as already suggested in Holland's original schema theory [16]—XCS is not able to process the lower level building blocks but tends to disrupt them. Additionally to standard crossover operators of uniform, one-point, and two-point crossover, we applied an informed crossover operator, called BB-wise uniform crossover. BB-wise uniform crossover is designed using the information about BBs so that it creates new classifiers by exchanging all attributes in each BB partition between the two parents with probability 50%. The BB-wise uniform crossover operator thus proceeds similarly to uniform crossover, but instead of exchanging individual attributes, it exchanges groups of attributes according to the BB structure of the problem.

XCS with BB-wise uniform crossover solves the problem efficiently. This indicates that processing subsets of attributes as opposed to individual attributes can sometimes allow solutions to problems intractable with standard recombination operators. Since many complex real-world systems are composed of smaller sub-systems, many real-world problems can be expected to contain decomposable structures. Decomposition exploited in XCS with standard crossover operators is severely limited and does not yield efficient solutions for problems where blocks of attributes must be processed, as illustrated in the investigated hierarchical problems. Thus, a mechanism in XCS is necessary that is able to identify the lower-level BB structure. Once identification is successful, effective BB processing and recombination can be applied.

Further studies of the complexity of the problem showed that the performance of XCS with BB-wise uniform crossover is nearly independent of the mutation type used [4]. The results were also replicated in several other hierarchical problem representations [4].

4. BUILDING BLOCK IDENTIFICATION

To meet the challenge of effective exploration in XCS by identifying and processing important BBs, it is necessary to develop a mechanism that can do this online. Estimation of distribution algorithms [24, 30, 19] seem to be most appropriate for this purpose for three basic reasons: (1) EDAs enable automated identification of BBs given only a limited sample of candidate classifiers, (2) EDAs enable the use of prior information to make BB identification more efficient, and (3) EDAs provide information about BBs in the form of a probabilistic model, which can be used and modified in several different ways in order to ensure effective exploration.

The evolutionary component in XCS differs from the usual GA application in several respects, though. Due to XCS's niched reproduction in action sets and since action sets are generally rather small compared to the whole population, it is difficult to identify BB structures using an action set alone. On the other hand, identifying BBs using the whole population results in a model that may not accurately encode classifiers in an action set. Thus, the inclusion of an EDA mechanism in XCS is not completely straight-forward.

This section integrates the BB-identification mechanism used in ECGA [13] to identify and process BBs. Next, the section integrates the more powerful Bayesian network models, which are used in BOA [29]. We show that both mechanisms are suitable to learn the *global* lower-level problem structure and can be used to generate or improve *local* classifier offspring. The generation and improvement of the local offspring depends on the current action set similar to the original XCS crossover operators.

To ensure that the learned model is of appropriate complexity and that it encodes only relevant dependencies, ECGA as well as BOA use measures based on Occam's razor and penalized marginal likelihood in model construction to balance model accuracy with model complexity.

4.1 BB-identification in the ECGA

To identify and combine BBs, ECGA uses marginal product models (MPMs), which can encode non-overlapping BBs. Considering the binary-string representation of individuals, an MPM clusters all string positions into groups of positions. The goal of clustering string positions is to identify groups of string positions so that interactions between string positions in each group are of much higher magnitude than interactions between the groups. To learn the model, the best individuals in the current population are used where the best individuals can be selected by any standard GA selection method. The model is then learned greedily to fit the dependencies identified in the selected set of solutions and the resulting model is sampled to generate offspring.

The greedy model building algorithm is controlled by a minimum description length (MDL) metric, in which models that allow higher compression of data (including the description of the model) are favored. The algorithm starts with each string position forming a separate group. In each iteration two groups are merged that yield highest increase in the model quality measured by the MDL metric. If no two groups can be merged to improve the model quality, the learning is terminated. New candidate solutions are then generated by sampling each group of positions independently according to the distribution of instances of this group in the selected population.

ECGA was shown to be able to identify BBs and solve problems with BBs of bounded order (such as deceptive trap problems) in a scalable manner [13, 33]. Due to its rather straight-forward approach and the various successful applications, it appears a valuable candidate for integration into XCS. For the integration we made use of the relevant parts of the available ECGA implementation [22].

4.2 BB-identification in BOA

BOA uses a more powerful class of models called Bayesian networks in order to identify, encode, and recombine BBs. The overall learning mechanism in BOA is similar to the one applied in ECGA. A Bayesian network is first built from a selected population of individuals. The build network is then sampled to generate offspring.

Bayesian networks (BNs) [26, 23] combine statistics with graph theory providing a modular graphical model of the analyzed data. BNs can be used to estimate and sample probability distributions as well as to do inference. A Bayesian network is defined by its structure and parameters. The structure is usually encoded by a directed acyclic graph with the nodes corresponding to the attributes and the edges corresponding to conditional dependencies. The parameters are represented by a set of conditional probability tables (CPTs) specifying a conditional probability for each variable given any instance of the variables that the variable depends on.

As the ECGA structure assumes the independence of the blocks, also a Bayesian network encodes a set of implicit independence assumptions. Each variable in a BN is independent of all its non-descendants given the values of this variable's parents [23]. To encode an MPM with a Bayesian network, the network would for example contain an edge between every pair of nodes in one cluster of variables but no edges between pairs of nodes from two different clusters of variables. Thus, BNs are more expressive that MPMs.

Like in ECGA, a greedy algorithm is used to learn a BN. The algorithm starts with an empty BN and iteratively adds that edge to the network that improves the quality of the network maximally. Network quality is measured by a combination of the Bayesian Dirichlet metric with likelihood equivalence [9, 14] and the Bayesian information criterion [34]. The metrics weigh network complexity against gained data compression following the MDL principle. Learning terminates when no more improvement is possible.

The sampling of a BN can be done 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 CPT for x_i using the known values of Π_i. We made us of Pelikan's BOA implementation available on the net [27], which uses decision graphs—a more efficient variant of BNs.

4.3 Learning Dependency Structures in XCS

BB-identification mechanisms of ECGA and BOA can be used to identify building blocks in a population of high-quality classifiers selected from the global population. However, relative accuracy may not be an appropriate measure since different problem niches may currently be differently populated resulting in different relative accuracies. Consequently, relative accuracy and thus fitness as the selection criterion may be misleading in this case.

Table 3: Sample classifiers (from the multiplexer problem) and their corresponding binary encoding for the structure learning mechanism. Spaces are added for clarity. If an attribute is a don't care symbol, the second bit in the corresponding binary code is chosen randomly. The class bit is flipped, if the reward prediction is below 500.

C	A	p	ε	binary encoding
##11##	1	750.0	375.0	10 11 01 01 11 10 1
##00##	1	250.0	375.0	11 11 00 00 10 11 0
0#1###	0	250.0	375.0	00 01 11 11 11 10 1
0#0###	0	750.0	375.0	00 00 10 11 10 10 0
0#11##	1	1000.0	0.0	00 11 01 01 11 10 1
0#11##	0	1000.0	0.0	00 11 01 01 11 11 0
001###	1	1000.0	0.0	00 00 01 10 10 10 1
10##0#	0	1000.0	0.0	01 00 11 11 00 10 0
000###	1	0.0	0.0	00 00 00 11 10 10 0
01#1##	0	0.0	0.0	00 01 11 01 11 11 1

To eliminate this problem, we use a filtering mechanism that extracts the most accurate classifiers out of the current population. Additionally, we require a minimum experience to avoid misleading information in young classifiers (as has been done before in XCS). The mechanism selects those classifiers that have a minimum experience θ_{be}, a minimum numerosity θ_{bn}, and a minimum error $\theta_{b\varepsilon}$. The parameters were set to $\theta_{be} = 20$, $\theta_{bn} = 1$, $\theta_{b\varepsilon} = 400$ throughout the subsequent experiments filtering out the young and high-error classifiers. Since predictions below the average reward of 500 can be considered as predictions of the opposite class with higher reward, we switch the class of those classifiers that predict a reward of less than 500. Note that this class switching can only be applied in classification problems in which only two types of reward (e.g. 1000/0) are possible.

In order to use the available code for learning and sampling probabilistic models, classifiers need to be transformed into a binary string representation of fixed length. Since specificity is of high importance in XCS, an explicit distinction between a specific (zero or one) and a general (don't care) attribute was expected to provide relevant structural information. Thus, we decided to code each conditional attribute by two bits: The first bit encodes if the attribute is general (that is, don't care) or specific. The second bit encodes the value of the attribute. If the attribute is a don't care symbol, we choose zero or one uniformly randomly for the second bit. The classification (class) is coded as yet another bit. If there are more than two classes, more bits will have to be used to encode the class. Encoding of the classification part may yet play a special role and future work may indicate that it is better to build models for each classification separately. Table 3 shows a set of classifiers and the corresponding encoding that is used to learn the probability structure.

With a binary coded set of individuals at hand, we are able to identify building blocks by applying the learning algorithms used in ECGA or BOA to the filtered population of binary-string classifiers. Note that since XCS applies a steady-state, niched GA, rebuilding the model in each iteration would be computationally expensive and unnecessary. Since the changes in each time step are small, it is more efficient to re-build the network after a fixed number of time steps θ_{bs}, usually set to $10,000$ in our experiments. Experiments showed that the threshold is only slightly problem dependent. In general, the lower the threshold, the more often the model is rebuilt resulting in a potential faster adjustment to newly discovered dependencies but also a potential waste of computation time for rebuilding unchanged dependency structures.

4.4 Sampling from the Dependency Structures

The learned dependency structure is used to generate new classifiers instead of using standard crossover operators. As long as the learned model reflects important BBs, it can be expected that the resulting recombination will be less disruptive and much more directed towards generating offspring that combines already successfully learned substructures effectively searching in the neighborhoods defined by the substructures. Additionally, since the model is built from the global population, the resulting recombination can be expected to bias the offspring towards global structural dependencies.

We investigate two approaches to sample the built model: (1) sample classifiers directly from the model using Probabilistic Logic Sampling (PLS) [15], and (2) use the Markov Chain Monte Carlo (MCMC) [25] method to update the structure of a selected classifier probabilistically based on the model. We also consider another variant in which the probabilities in the global model are reset to values reflecting the local distribution. This is done by selecting several classifiers in the current action set via tournament selection and using those classifiers to determine the probabilities. In this way, XCS uses the dependency structure encoded in the whole population but still biases the generation of new classifiers to the current action set distribution. Figure 2 shows the different potential methods for offspring generation using a probabilistic model.

PLS starts by ordering the attributes based on the built Bayesian network so that no attribute can follow an attribute that depends on it. Then, the values of all attributes are generated one by one in this ordering. Because of the ordering, all attributes that an attribute depends on must have been generated once this attribute is going to be sampled.

MCMC sampling starts with one classifier from the action set. It proceeds by making small changes to this classifier, accepting each change with a probability that is equal to the ratio of the likelihood after the modification and the sum of the likelihoods before and after the modification.

In XCS we select the classifier to serve as the starting point for MCMC via tournament selection in the current action set. We then apply MCMC by flipping bits in the binary classifier representation at random and determining the likelihood of the classifier structure before and after the flip using the probabilistic model. If the likelihood increases by flipping the particular bit, the probability of accepting the change is higher than the probability of accepting a flip that decreases the likelihood. To avoid zero likelihoods, all conditional probabilities are linearly normalized to values ranging from 0.05 to 0.95.

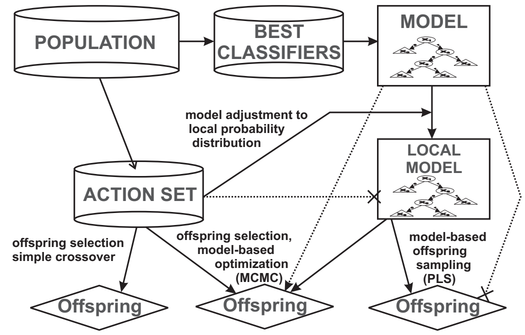

Figure 2: Potential offspring generations by the means of the built probabilistic model

When a global model with global parameters is used with PLS, sampling is not biased by the action set at all and the exploration suffers. When a global model with global parameters is used with MCMC, too many update iterations can also be expected to be ineffective because the resulting classifier will reflect the global population; on the other hand, too few updates will result in no exploration at all. The use of the global model with local probabilities is much more robust and performs well under all tested settings.

5. EXPERIMENTS

We evaluate XCS performance on the proposed hierarchical problems to evaluate and compare both offspring generation methods. XCS with Bayesian networks with decision graphs of BOA will be denoted by XCS/BOA, whereas XCS with marginal product models of ECGA will be denoted by XCS/ECGA. XCS builds a new model structure every 10,000 learning steps ($\theta_{bs} = 10,000$). The model is built from the filtered population as explained above. If the filtered population is empty, no model is learned. As long as no model is learned, XCS applies uniform crossover instead of the model-based crossover. Mutation is applied to the offspring classifiers generated by the model.

Figure 3 shows XCS/ECGA performance in the hierarchical 3-parity, 6-multiplexer problem. Model 50 or 10 specifies whether the number of classifiers used to set the model parameters to the current action set is 50 or 10. XCS/ECGA performs slightly worse than XCS with BB-wise uniform crossover, because XCS/ECGA is not given information about the overall problem structure and must discover this structure automatically. However, the difference between the performance of XCS/ECGA and XCS with BB-wise crossover is small. Furthermore, XCS/ECGA clearly outperforms XCS with no crossover or uniform crossover. Due to potential slight specialization effects when optimizing model structure, XCS/ECGA becomes more mutation rate independent.

XCS/BOA yields similarly successful solutions to the 3-parity, 6-multiplexer problem (Figure 4). The two numbers in the BOA settings indicate the offspring generation type: The first number refers to the number of selected classifiers used to set the probabilities to the local distribution (0 denotes the case with probabilities computed from the global population). The second number denotes the number of updates applied to a selected parental classifier using MCMC sampling (0 stands for sampling directly from the

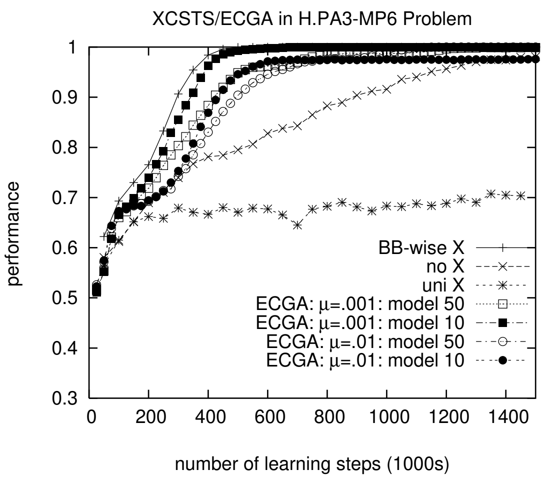

Figure 3: The application of the XCS/ECGA yields competent performance.

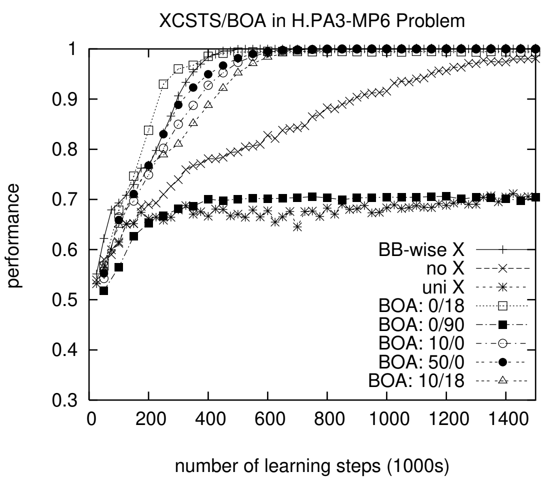

Figure 4: Also XCS/BOA learns the decomposable problem reliably.

model using PLS). BOA actually slightly outperforms BB-wise uniform crossover when optimizing the offspring with the global probabilities (0/18). As predicted, too many optimization steps with the global model probabilities are disruptive (0/90) because the offspring is too similar to the global population. This problem does not occur when using local probabilities independently of the sampling method used. All settings exhibit similar performance, which is nearly as good as that with BB-wise uniform crossover.

The results confirm that XCS can be successfully combined with advanced recombination operators capable of identifying, propagating, and combining important BBs. The implemented XCS/ECGA and XCS/BOA combinations showed to be able to achieve performance similar to the performance with BB-wise uniform crossover, which relies on explicit problem knowledge. XCS/ECGA and XCS/BOA do not require any prior problem knowledge and thus allow XCS to flexibly adjust its recombination operators to the particular problem at hand.

6. SUMMARY AND CONCLUSIONS

In this paper we showed that also in LCSs, and the XCS classifier system in particular, effective building block (BB) processing can be necessary for an efficient and reliable evolution of a complete problem solution. We showed that, as in GAs, if there are strong dependencies between different attributes, it is necessary to identify and exploit these dependencies to enable more effective exploration. Mutation alone is strongly handicapped taking a very long time to find the optimal problem solution. Uninformed, simple crossover operators are prone to disrupt the evolutionary search. If XCS uses recombination operators based on probabilistic modeling adopted from ECGA and BOA, it becomes capable of identifying and processing important BBs. Further confirmatory results in various other binary classification problems can be found elsewhere [4]. These results also show that the final solution is of a generality equal to the one achieved with informed crossover.

Additionally, we highlighted the difference between LCSs and GAs in that the model in LCSs must be biased to the current action set, which is usually rather small. To alleviate this problem, a combination of model learning mechanisms is used where the dependency model is learned from the global population in order to ensure a sufficiently large sample for learning dependencies, but the parameters of this model are set to the local distribution in the current action set.

In their current form, the proposed LCSs are restricted to the binary realm. However, similarly as in GAs [3, 20, 1, 31], real-valued extensions of XCS/ECGA and XCS/BOA for the real-valued XCSR system [39] are imaginable.

In conclusion, XCS/ECGA and XCS/BOA may be termed *competent LCSs*, that is, LCSs that are able to solve boundedly difficult problems—those with a bounded BB size—efficiently. The competent XCS is ready to be applied to other problem scenarios and environments. It is expected to work particularly well in problems in which predictions differ in different problem subspaces but the structural properties of the subspaces are related over the whole problem space. Future research needs to confirm these conjectures in binary, real-valued, as well as real-world problem domains.

Acknowledgments

We are grateful to Pier Luca Lanzi, Kei Onishi, Kumara Sastry, and the whole IlliGAL lab for their help and the useful discussions.

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Butz's contribution received additional funding from the German research foundation (DFG) under grant DFG HO1301/4-3 as well as from the European commission contract no. FP6-511931. Pelikan's contribution was supported by the Research Award and the Research Board at the University of Missouri.

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 ex-

pressed or implied, of any of the organizations mentioned above.

7. REFERENCES

[1] C. W. Ahn, R. S. Ramakrishna, and G. Goldberg. Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. *Genetic and Evolutionary Computation Conference (GECCO-2004)*, 3102:840–851, 2004.

[2] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[3] P. A. Bosman and D. Thierens. Mixed IDEAs. Utrecht University Technical Report UU-CS-2000-45, Utrecht University, Utrecht, Netherlands, 2000.

[4] M. V. Butz. *Rule-based evolutionary online learning systems: Learning bounds, classification, and prediction*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2004.

[5] 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.

[6] 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.

[7] 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.

[8] M. V. Butz, K. Sastry, and D. E. Goldberg. Tournament selection in XCS. *Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1857–1869, 2003.

[9] G. F. Cooper and E. H. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[10] K. A. De Jong and W. M. Spears. Learning concept classification rules using genetic algorithms. *IJCAI-91 Proceedings of the Twelfth International Conference on Artificial Intelligence*, pages 651–656, 1991.

[11] J. J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1979.

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

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

[14] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research, Redmond, WA, 1994.

[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 Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[17] 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.

[18] T. Kovacs. Deletion schemes for classifier systems. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 329–336, 1999.

[19] 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.

[20] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Pena. Optimization in continuous domains by learning and simulation of Gaussian networks. In A. Wu, editor, *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 201–204, San Fransisco, CA, 2000. Morgan Kaufmann.

[21] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[22] F. Lobo and G. Harik. Extended compact genetic algorithm in C++. IlliGAL report 99016, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1999.

[23] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, MA, 1997.

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

[25] 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.

[26] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[27] M. Pelikan. Bayesian optimization algorithm, decision graphs, and Occam's razor. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 511–518, 2001.

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

[29] 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.

[30] 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.

[31] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert. *Information Sciences*, 156(3–4):36–45, 2003.

[32] J. P. Rivera and R. Santana. Improving the discovery component of classifier systems by the application of estimation of distribution algorithms. In *Proceedings of the students sessions, ACAI'99*, pages 43–44, Chania, Greece, 1999.

[33] K. Sastry and D. E. Goldberg. On extended compact genetic algorithm. IlliGAL report 2000026, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2000.

[34] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[35] H. A. Simon. *Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[36] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. *Parallel Problem Solving from Nature*, pages 97–106, 1998.

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

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

[39] S. W. Wilson. Get real! XCS with continuous-valued inputs. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning classifier systems: From foundations to applications (LNAI 1813)*, pages 209–219. Springer-Verlag, Berlin Heidelberg, 2000.