Computational Complexity of the XCS Classifier System

Martin V. Butz, David E. Goldberg, and Pier Luca Lanzi

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

1 Introduction

Learning classifier systems (LCSs) are online-generalizing rule-based learning systems that use evolutionary computation techniques to evolve an optimal set of rules, that is, a *population of classifiers* (1; 2). LCSs tackle both *single-step* classification problems and *multi-step* reinforcement learning (RL) problems. Although the LCS proposal dates back over twenty years ago, there has been hardly any theory regarding convergence, computational effort, problem instances, etc. Successful applications seemed to rather rely on a "black art" of correct parameter settings, supported by powerful computers, than on actual insight.

XCS (3) can be viewed as a mile-stone of learning classifier system research. The XCS system combines an accuracy-based fitness approach with a niched genetic algorithm (4; 5). Recent results show that XCS can solve typical datamining problems in a machine-learning competitive way, providing classification accuracy that is comparable to that of well-known machine learning algorithms, such as C4.5, nearest neighbor, Naive Bayes, and support vector machines (6; 7; 8).

This chapter connects LCSs, and specifically the XCS classifier system, to important elements of computational learning theory. We focus on the most fundamental class of concept learning problems, the learning of Boolean functions. Based on previous facetwise analyses resulting in several bounding models (9; 10; 11; 12), we show that k-DNF problems that satisfy few additional properties are PAC-learnable (13; 14) by XCS. That is, XCS scales polynomially in time and space complexity learning with high probability an approximately correct solution. The proof also confirms Wilson's previous conjecture on XCS's scalability (15).

The analysis essentially proves that XCS is an effective machine learning system that learns complex machine learning problems with a computational effort that scales similarly to other machine learning systems. Moreover, since XCS is an evolutionary learning system with very generally applicable learning mechanisms, the analysis actually confirms the general learning competence of the XCS system, which searches effectively for accuracy structures in any provided problem search space.

Due to its general applicability and its online learning capability, XCS is certainly not the most effective learning algorithm to solve k-DNF problems (see

e.g. (16)). However, XCS is not particularly targeted to solve k-DNF problems. Rather, XCS is a much more general, noise-robust problem solver. The aim of this paper is to show that LCSs, and XCS in particular, can be shown to be PAC-learning algorithms. The advantage of XCS compared to more specialized PAC-learning algorithms is its flexibility and generality in their applicability to different problem types, problem representations, and problem domains as well as to online learning problems.

To prevent complete problem dependence, our analysis does not model the system behavior exactly, as could be done for example with a Markov-chain model. Only general problem features are required to predict system behavior using our approach. Consequently, the derived problem bounds are flexible, modifiable, and less problem-dependent than an exact analysis. Additionally, the facet-wise analysis improves the overall understanding of XCS providing a general theory of how XCS evolves solutions that are maximally accurate and maximally general.

The chapter first gives a short introduction to XCS. Next, we provide a short evolutionary pressure overview that shows how XCS works. Section 4 derives the computational complexity of XCS. Summary and conclusions discuss the impact and extendibility of the presented research.

2 The XCS Classifier System

XCS learns in typical RL settings interacting online with an unknown environment that provides problem instances and reward feedback. Hereby, it learns to predict future reward values accurately.

XCS (as all other LCSs) represents the problem solution by a *population* of *classifiers*. At each time step, XCS receives a problem instance. Based on its current knowledge, XCS proposes a solution for the instance. Depending on the problem instance and the solution proposed, XCS receives numerical reward characterizing the goodness of the proposed solution. In essence, XCS is designed to evolve a complete, maximally accurate, and maximally general representation of the optimal problem solution. This section gives an overview over the basic structure and mechanisms in XCS, further details can be found in (17).

For the purpose of our analysis, we introduce XCS as a pure classification system in which no reward propagation is necessary. However, the reader should keep in mind that XCS is a much more general learning system that is able to learn in more general, multistep RL problems, in which (back-)propagation of reward is necessary in order to learn an optimal problem solution.

2.1 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 consequently 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 underlying a

certain probability distribution D. If not stated differently, we assume a uniform distribution 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. In the simplest case, reward 0 indicates an incorrect classification and a non null constant reward (e.g., 1000) indicates a correct classification.

2.2 Knowledge Representation

As mentioned above, knowledge is represented by a *population* $[P]$ (that is, a set) of *classifiers* (that is, classification rules). Each classifier is characterized by 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. Effectively, a problem solution is represented by a disjunctive normal form in which each accurate classifier specifies one conjunctive clause.

2.3 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. $[M]$ is used to decide on the classification on the current problem forming fitness-weighted reward predictions of each possible classification. 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 updated according to the following equations:

$$R \leftarrow R + \beta(r - R), \tag{1}$$

$$\varepsilon \leftarrow \varepsilon + \beta(|r - R| - \varepsilon), \tag{2}$$

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0 \\ \alpha(\varepsilon/\varepsilon_0)^{-\nu} & \text{otherwise} \end{cases}, \quad \kappa' = \frac{\kappa}{\sum_{x \in [A]} \kappa_x}, \tag{3}$$

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

Parameter β denotes the learning rate, ε_0 denotes the error tolerance, α and ν are additional constants that scale the fitness evaluation. In XCS, classifier prediction R (Equation 1) corresponds to the Q-values (formal details in (18)); classifier fitness F (equations 2–4) essentially estimates the scaled, average, relative accuracy of the classifier derived from the current reward prediction error ε with respect to the competing classifiers.

2.4 Rule Evolution

Initially, the population $[P]$ is empty. When a problem instance is presented 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* $(1 - P_{\#})$. If $P_{\#}$ is close to one, XCS basically starts from very general hypotheses and basically pursues a *general-to-specific* search. If $P_{\#}$ is small, XCS starts from very specific hypotheses and the search follows a *specific-to-general* approach.

The genetic algorithm (GA) is the main rule learning component. Because classifier fitness estimates the accuracy of the reward prediction, the GA favors the evolution of classifiers which provide an accurate prediction of the expected payoffs. The genetic algorithm used is a steady-state niched genetic algorithm (19). Each problem iteration, the GA may be applied. The GA reproduces two classifiers selecting 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 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 which favors more general, accurate previous classifiers over more-specialized offspring classifiers.

3 Evolutionary Pressures in XCS

XCS is designed to evolve a complete, accurate, and maximally general representation of the reward function of the problem. More accurate classifiers are favored due to the accuracy-based selection. Since classifiers are favorably selected if they are more accurate, the GA in XCS pushes toward a population of maximally accurate classifiers. More general classifiers are favored since more general classifiers match and therefore reproduce more often. Since more general classifiers match and therefore reproduce more often, the overall effect of the GA in XCS is an evolutionary pressure toward a solution consisting of maximally accurate and maximally general classifiers. This simple principle, firstly stated by (3), has been theoretically analyzed in (10), in which a basic idea of XCS functioning is provided. The analysis characterizes the different evolutionary biases as *evolutionary pressures*, which together guide XCS in the evolution of the desired complete, maximally accurate, and maximally general problem solution representation. The four most important pressures are (1) *set pressure*, (2) *mutation pressure*, (3) *deletion pressure*, and (4) *fitness pressure*.

Set pressure refers to the generalization pressure in XCS resulting from niche reproduction but population-wide deletion. Mutation pressure generally can be regarded as a diversification pressure that results in a local search in the neighborhood of the selected classifier. Mutation essentially pushes the population towards an equal distribution of symbols in classifier conditions. In terms of specificity in the ternary case, mutation pushes towards an equal distribution

of zeroes, ones, and don't care symbols and thus towards a specificity of 2/3. Deletion pressure additionally prefers the deletion of inaccurate classifiers and classifiers that populate over-populated niches. In general, though, deletion pressure in XCS is weak and can be approximated by random deletion from the population.

Combining set pressure and mutation pressure assuming random deletion in $[P]$, we can derive a general *specificity equation* that expects the specificity change in the population.

$$\sigma[P'] = \sigma[P] + f_{ga}\frac{2(\sigma[A] + \Delta_\mu(\sigma[A]) - \sigma[P])}{N} \tag{5}$$

where $\sigma[X]$ denotes the average specificity of all classifier conditions in set $[X]$, f_{ga} denotes the frequency of GA application, Δ_μ denotes the specificity change due to mutation, and N denotes the population size. (10) evaluate the accuracy of the specificity equation in detail confirming that given no further fitness influence, the specificity in the population behaves in the derived way and converges to the predicted value.

Fitness pressure is the main pressure towards higher accuracy. In essence, since more accurate classifiers are selected for reproduction, classifiers with higher accuracy and thus classifiers with lower error are propagated. In general, the degree of fitness pressure is strongly problem dependent. In (9) it was shown that tournament selection results in a much more reliable fitness pressure towards higher accuracy.

In sum, XCS's evolutionary pressures propagate the evolution of accurate, maximally general classifiers in the population. Imagining a specificity axis along which the classifier population evolves, the pressures can be visualized as shown in Figure 1. While fitness pressure prevents over-generalization, set pressure prevents over-specialization. Mutation serves as the diversification mechanism that enables the fitness pressure to evolve more accurate classifiers. However, it also has a general specialization tendency. Crossover is not visualized since it has no immediate impact on specificity. However, the mechanism can be very important in the effective propagation of accurate sub-solutions dependent on the problem (19; 9).

4 Towards Computational Complexity

With the knowledge of XCS's learning biases at hand, we can now analyze XCS's learning complexity. Essentially, we now assume that the learning pressures are set correctly and that XCS will evolve a complete problem solution as long as enough computational resources are available. Thus, we do not investigate the chosen fitness approach, the type of offspring selection, or improvements in the applied search operators in further detail. The interested reader is referred to the available literature (20; 21; 22). Rather, we investigate the computational effort necessary to assure that the evolutionary pressures can apply.

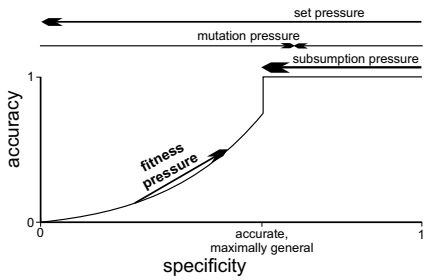

Fig. 1. The visualized evolutionary pressures give an intuitive idea how XCS evolves the intended complete, accurate, and maximally general problem representation.

Each learning iteration, all classifiers need to be monitored as matching candidates so that the computational effort in each learning iteration is bounded by the population size N. We may denote the number of iterations until an optimal solution is found by the learning time t^* so that the overall computational effort grows in Nt^*.

Several constraints need to be satisfied to ensure that the evolutionary pressures can apply, that enough time is allocated to evolve a complete solution, and that the evolved solution is sustained. In essence, the following four facets need to be satisfied.

1. *Population initialization* needs to ensure time for classifier evaluation and successful GA application.
2. *Schema supply* needs to be ensured to have better classifiers available.
3. *Schema growth* needs to be ensured to grow those better classifiers evolving a complete problem solution.
4. *Solution sustenance* needs to be ensured to sustain a complete problem solution.

We address these issues in the subsequent sections.

First, we derive the *covering bound* to ensure proper initialization. Second, we derive the *schema bound* to ensure the availability of better schema representatives for reproduction. We use the schema bound to derive initial settings for population specificity and population size. Also the time it takes to generate a better classifier at random is considered.

Next, we derive the *reproductive opportunity bound* to ensure schema growth by ensuring that better classifiers can be detected and reproduced successfully. We show that the schema bound and the reproductive opportunity bound somewhat interact since, intuitively, too much supply implies too large specificity consequently disabling reproduction.

While the reproductive opportunity bound assures that better classifiers grow, we are also interested in how long it takes them to grow. This is expressed in the learning time bound that assures solution growth.

Once enough learning time is available to make better classifiers grow until the final solution is found, we finally need to assure *sustenance* of the grown solu-

tion. XCS applies niching in that it reproduces in problem subspaces and deletes from the whole population. Larger niches are preferred for deletion. The analysis results in a final population size bound ensuring the sustenance of a complete problem solution as long as there are no severe problem solution overlaps.

Putting the results together, we are able to derive a positive computational learning theory result for an evolutionary-based learning system with respect to *k-DNF* functions. We show that XCS is able to PAC-learn *k-DNF* functions with few restrictions. However, the reader should keep in mind that XCS is a system that is much more broadly applicable and actually an online generalizing RL system. XCS's capability of PAC-learning *k-DNF* functions confirms its general learning scalability and potential widespread applicability.

4.1 Proper Population Initialization: The Covering Bound

Several issues need to be considered when intending to make time for classifier evaluation and thus the identification of better classifiers. The first bound is derived from the rather straight-forward requirement that the evolutionary algorithm in XCS needs to apply. That is, genetic reproduction in action sets and deletion in the population needs to take place.[1]

Reproduction will not occur if XCS gets stuck in an infinite covering-deletion cycle. This can happen if the initial population is filled with over-specialized classifiers. In this case, inputs may continuously not be covered since covering will continue to generate over-specialized classifiers and delete random other overspecialized classifiers. In this case, the GA will never take place and no evolutionary pressures apply. This scenario can only happen if the maximal population size N is set too small and initial specificity, controlled by the *don't care parameter* $P_\#$, is set too high.

As specified above, XCS triggers the creation of a covering classifier if the current problem instance is not matched by at least one classifier for each possible classification. If the population is already filled up with classifiers, other classifiers are deleted to make space for the new covering classifiers. Given the population is filled up with overspecialized classifiers that did not undergo any evaluation so far, and since all covering classifiers have identical fitness F and action set size estimate *as* values, classifiers are effectively selected uniformly randomly for deletion. Thus, in this case, a *covering-random deletion* cycle may continue for a long time.

Assuming a uniform problem instance distribution over the whole problem space $\mathcal{S} = \{0,1\}^l$, we can determine the probability that a given problem instance is covered by at least one classifier in a randomly generated population:

$$P(\texttt{cover}) = 1 - \left(1 - \left(\frac{2 - \sigma[P]}{2}\right)^l\right)^N, \tag{6}$$

[1] Related publications of parts of this and the following section can be found elsewhere (23; 9; 10).

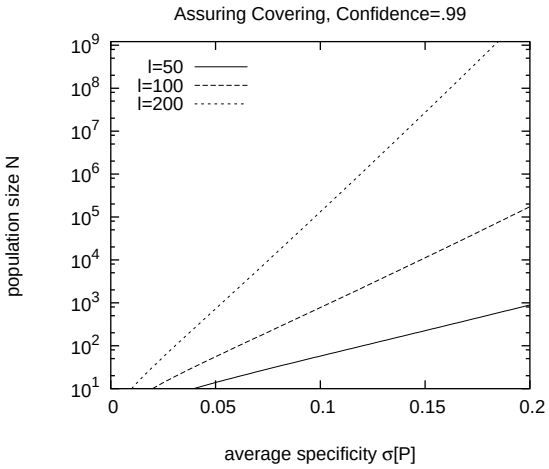

Fig. 2. To ensure that better classifiers can be identified, population size needs to be set high enough and specificity low enough to satisfy the covering bound.

where $\sigma[P]$ may be equated with $1 - P_{\#}$ in the beginning of a run. Similarly, we can derive the actual necessary maximal specificity given a certain population size using the inequality $1 - \exp^x < x$ setting $(1 - \texttt{cover})$ to $1/\exp$:

$$\sigma[P] < 2(1 - (\frac{1}{N})^{1/l}) < 2 - 2(1 - (1 - \texttt{cover})^{1/N})^{1/l}, \tag{7}$$

showing that increasing N results in an increase in maximal specificity polynomial in $1/l$ deriving an effective rule of thumb of how low the don't care probability $P_{\#}$ may be set to assure an appropriate specificity $\sigma[P]$. Figure 2 shows the resulting boundary conditions on population size and specificity for different problem lengths requiring a confidence level of 0.99.

To automatically avoid the covering bound, XCS could be enhanced to detect infinite covering and consequently increase the $P_{\#}$ value. However, we did not experiment with such an enhancement so far since usually the covering bound can be easily circumvented by setting the parameter $P_{\#}$ large enough.

Given that the problem instances are not sampled uniformly randomly in the problem space, the covering bound can be used as an upper bound. Since the covering mechanism generates classifiers that cover actual instances and the genetic algorithm mainly focuses on generating offspring classifiers that apply in the current problem niche, the smaller the set of sampled problem instances, the larger the probability that an instance is covered. In most RL problems as well as in datamining problems the number of distinct problem instances is usually much smaller than the whole problem space so that the covering bound becomes less important in these problems.

4.2 Ensuring Supply: The Schema Bound

The supply question relates to the schema supply in GAs. Similar to GAs, LCSs process *building blocks* (BBs)—small dependency structures in the overall problem structure that result in an increase in fitness. Since the fitness structure of one BB may point towards the mediocre solution (that is, a local optimum), the optimal solution to a BB as a whole needs to be processed. Thus, disregarding mutation effects, individuals that represent the optimal BB structure need to be available from the beginning to be able to propagate the BB.

The same observation applies in XCS albeit in slightly different form. The question arises, what is a BB in the XCS classifier system? We know that fitness is based on accuracy. Thus, a BB in XCS is a substructure in the problem that increases classification accuracy. As there are BBs for GAs, there are minimal substructures in classification problems that result in higher accuracy (and thus fitness).

To establish a general notion of BBs in XCS, we use the notion of a *schema* as suggested elsewhere (24; 4). A schema for an input of length l is defined as a string that specifies some of the positions and ignores others. The number of specified positions is termed the *order* k of the schema. A schema is said to be *represented* by a classifier if the classifier correctly specifies *at least* all positions that are specified in the schema. Thus, a representative of a schema of order k has a specificity of at least $\sigma(.) = k/l$. For example, a classifier with condition $C =$`##10#0` is a representative of schema `**10*0`, but also of schemata `**10**`, `**1**0`, `***0*0`, `**1***`, `***0**`, `*****0`, and `******`.

Let's consider now a specific problem in which a minimal order of at least k_m bits needs to be specified to reach higher accuracy. We call such a problem a problem that has a *minimal order of difficulty* k_m. That is, if less than k_m bits are specified in a classifier, the class distribution in the specified problem subspace is equal to the overall class distribution. In other words, the *entropy* of the class distribution decreases only if at least some k_m bits are specified. Since XCS's fitness is derived from accuracy, representatives of the schema of order k_m need to be present in the population.

Population Size Bound To assure the supply of the representatives of a schema of order k_m, the population needs to be specific enough and large enough. It is possible to determine the probability that a randomly chosen classifier from the current population is a schema representative by:

$$P(\texttt{representative}) = \frac{1}{n}\left(\frac{\sigma[P]}{2}\right)^{k_m}, \tag{8}$$

where n denotes the number of possible actions and $\sigma[P]$ denotes the specificity in the population, as before. From Equation 8 we can derive the probability of the existence of a representative of a specific schema in the current population

$$P(\texttt{representative exists}) = 1 - \left(1 - \frac{1}{n}\left(\frac{\sigma[P]}{2}\right)^{k_m}\right)^{N}, \tag{9}$$

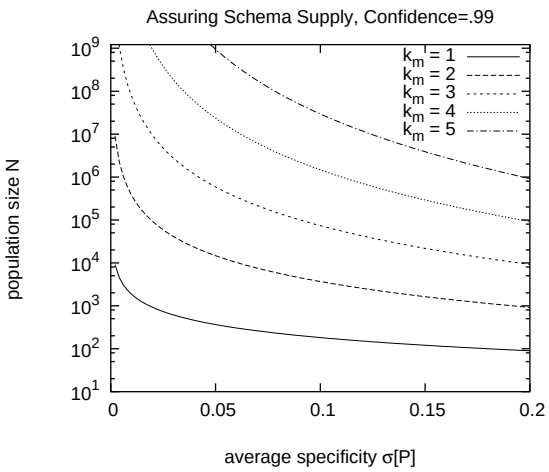

Fig. 3. The schema bound requires population size to be set sufficiently high with respect to a given specificity. The larger the population size, the lower the necessary specificity.

basically deriving the probability that at least on schema representative of order k_m exists in $[P]$.

As shown in previous publications (25; 10; 9), in a problem in which no current fitness pressure applies, specificity can be approximated by twice the mutation probability μ, that is, $\sigma[P] \approx 2\mu$. Additionally, the population may be initialized to a desired specificity value by choosing parameter $P_\#$ appropriately. It should be kept in mind, though, that albeit $P_\#$ might bias specificity further early in the run, without any fitness influence, specificity converges to a value that can be approximated by 2μ. Thus, mutation determines specificity on the long term. Well-chosen $P_\#$ values may boost initial XCS performance.

Requiring a high probability for the existence of a representative, we can derive the following population size bound using the inequality $x < -ln(1-x)$:

$$N > -n(\frac{2}{\sigma[P]})^{k_m} ln(1 - P(\texttt{rep.exists})) > \frac{\log(1 - P(\texttt{rep.exists}))}{\log(1 - \frac{1}{n}(\frac{\sigma[P]}{2})^{k_m})}, \quad (10)$$

which shows that N needs to grow logarithmically in the probability of error and exponentially in the minimal order of problem difficulty k_m given a certain specificity. Enlarging the specificity, we can satisfy the schema bound. However, schema growth may be violated if specificity is chosen too high as shown in the subsequent sections.

Figure 3 shows the schema bound plotting the required population size with respect to a given specificity and several minimal orders k_m (left-hand side). Population size is plotted in log scale due to its exponential dependence on the order k_m.

Specificity Bound Similar to the bound on population size, given a certain problem of minimal order of problem difficulty k_m, we can derive a minimal specificity bound from Equation 9 assuming a fixed population size:

$$\sigma[P] > 2n^{1/k_m}(1-(1-P(\texttt{rep.exists}))^{1/N})^{1/k_m} \tag{11}$$

Setting $(1-P(\texttt{rep.exists}))$ to $1/\exp$ we can use the inequality $1-\exp^{-x} < x$ to derive that:

$$\sigma[P] > 2\left(\frac{n}{N}\right)^{1/k_m} \tag{12}$$

Note that an identical derivation is possible determining the expected number of schema representatives in $[P]$ given specificity $\sigma[P]$:

$$E(representative) = \frac{N}{n}\left(\frac{\sigma[P]}{2}\right)^{k_m} \tag{13}$$

Requiring that at least one representative can be expected in the current population

$$E(representative) > 1, \tag{14}$$

also yields Equation 12.

We may rewrite Equation 12 using the O-notation. Given a population size of N and the necessary representation of an unknown schema of order k_m, the necessary specificity $\sigma[P]$ can be bounded by

$$\sigma[P] : \; O\left(\left(\frac{n}{N}\right)^{1/k_m}\right), \tag{15}$$

showing that the required specificity decreases polynomially with increasing population size N and increases exponentially with increasing problem complexity k_m. Since we have shown that population size N also needs to increase exponentially in k_m but necessary specificity decreases polynomially in N, the two effects cancel each other. Thus, it is possible to leave specificity and thus mutation constant and to focus only on a proper population size to assure effective schema supply.

Extension in Time Given we start with a completely general or highly general initial classifier population (that is, $P_\#$ is close to 1.0), the schema bound also extends in time. In this case, it is the responsibility of mutation to push the population towards the intended specificity generating initial supply.

Given a mutation probability μ, the probability can be approximated that a classifier is generated that has all k_m relevant positions specified given a current specificity $\sigma[P]$:

$$P(\texttt{generation of representative}) = (1-\mu)^{\sigma[P]k_m} \cdot \mu^{(1-\sigma[P])k_m} \tag{16}$$

With this probability, we can determine the expected number of steps until at least one classifier may have the desired attributes specified. Since this is a geometric distribution:

$$E(t(\texttt{generation of representative})) = \\ 1/P(\texttt{generation of representative}) = \\ \left(\frac{\mu}{1-\mu}\right)^{\sigma[P]k_m} \mu^{-k_m} \tag{17}$$

Given a current specificity of zero, the expected number of steps until the generation of a representative consequently equals to μ^{-k_m}. Thus, given we start with a completely general population, the expected time until the generation of a first representative is less than μ^{-k_m} (since $\sigma[P]$ increases over time). Requiring that the expected time until a classifier is generated is smaller than some threshold Θ, we can generate a lower bound on the mutation μ:

$$\mu^{-k_m} < \Theta \\ \mu > \Theta^{\frac{1}{-k_m}} \tag{18}$$

The extension in time is directly correlated with the specificity bound in Equation 12. Setting Θ to N/n we get the same bound (since σ can be approximated by 2μ).

As mentioned before, although supply may be assured easily by setting the specificity σ and thus $P_{\#}$ and more importantly the mutation rate μ sufficiently high, we yet have to assure that the supplied representatives can grow. This is the concern of the following section in which the the reproductive opportunity bound is derived.

4.3 Making Time for Growth: The Reproductive Opportunity Bound

To ensure the growth of better classifiers, we need to ensure that *better* classifiers get reproductive opportunities. So far, the covering bound only assures that reproduction and evaluation are taking place. This is a requirement for ensuring growth but not sufficient for it. This section derives and evaluates the *reproductive opportunity bound* that provides a population size and specificity bound that assures the growth of better classifiers.[2]

The idea is to assure that more accurate classifiers need to be ensured to undergo reproductive opportunities before being deleted. To do this, we minimize the probability of a classifier being deleted before being reproduced. The constraint effectively results in another population and specificity bound since only a larger population size and a sufficiently small specificity can assure reproduction before deletion.

[2] Related publications of parts of this section can be found elsewhere (26; 9).

General Population Size Bound To derive the bound, we first determine the expected number of steps until deletion. Assuming neither any fitness estimate influence nor any action set size estimate influence, the probability of deletion is essentially random. Thus, the probability of deleting a particular classifier in a learning iteration equals:

$$P(\texttt{deletion}) = \frac{2}{N}, \tag{19}$$

since two classifiers are deleted per iteration. A reproductive opportunity takes place if the classifier is part of an action set. As shown elsewhere (22) the introduced action-set size proportionate tournament selection bounds the probability of reproduction of the best classifier from below by a constant. Thus, the probability of being part of an action set directly determines the probability of reproduction:

$$P(\texttt{reproduction}) = \frac{1}{n} 2^{-l\sigma(cl)} \tag{20}$$

Note that as before this derivation assumes binary input strings and further uniform random sampling from the problem space. Combining Equation 19 with Equation 20, we can determine that neither reproduction nor deletion occurs at a specific point in time:

$$\begin{aligned} P(\texttt{no rep., no del.}) &= (1 - P(\texttt{del.}))(1 - P(\texttt{rep.})) = \\ &= (1 - \frac{2}{N})(1 - \frac{2^{-l\sigma[P]}}{n}) = 1 - \frac{2}{N} - \frac{2^{-l\sigma[P]}}{n}(1 - \frac{2}{N}) \end{aligned} \tag{21}$$

Together with equations 19 and 20, we can now derive the probability that a certain classifier is part of an action set before it is deleted:

$$\begin{aligned} P(\texttt{rep.before del.}) &= P(\texttt{rep.})(1 - P(\texttt{del.})) \sum_{i=0}^{\infty} P(\texttt{no rep., no del.})^i = \\ & P(\texttt{rep.})(1 - P(\texttt{del.})) \frac{1}{1 - P(\texttt{no rep., no del.})} = \\ & \frac{\frac{1}{n} 2^{-l\sigma[P]}(1 - \frac{2}{N})}{\frac{2}{N} + \frac{1}{n} 2^{-l\sigma[P]}(1 - \frac{2}{N})} = \frac{\frac{1}{n} 2^{-l\sigma[P]}}{\frac{2}{N-2} + \frac{1}{n} 2^{-l\sigma[P]}} = \frac{N-2}{N-2+n2^{l\sigma[P]+1}} \end{aligned} \tag{22}$$

Requiring a certain minimal reproduction-before-deletion probability and solving for the population size N, we derive the following bound:

$$N > \frac{2n2^{l\sigma[P]}}{1 - P(\texttt{rep.before del.})} + 2 \tag{23}$$

This bounds the population size by $O(n2^{l\sigma})$. Since specificity σ can be set proportional to $\sigma = 1/l$, the bound actually diminishes usually. However, in problems in which the problem complexity $k_m > 1$, we have to ensure that classifiers that represent order k_m schemata have reproductive opportunities.

Given a current population-wide specificity $\sigma[P]$, the expected specificity of a representative of an order k_m schema can be estimated by

$$E(\sigma(\texttt{repres.of schema of order } k_m)) = \frac{k_m + (l - k_m)\sigma[P]}{l}, \tag{24}$$

assuming that the specificity in the other $l - k_m$ attributes equals the specificity in the population. Substituting $\sigma(cl)$ from Equation 23 with this expected specificity of a representative of a schema of order k, the population size N can be bounded by

$$\begin{aligned} N &> 2 + \frac{n2^{l \cdot \frac{k+(l-k)\sigma[P]}{l}+1}}{1 - P(\texttt{rep.before del.})} \\ N &> 2 + \frac{n2^{k+(l-k)\sigma[P]+1}}{1 - P(\texttt{rep.before del.})} \end{aligned} \tag{25}$$

This bound ensures that the classifiers necessary in a problem of order of difficulty k_m get reproductive opportunities. Once the bound is satisfied, existing representatives of an order k_m schema are ensured to reproduce before being deleted and XCS is enabled to evolve a more accurate population.

Note that this population size bound is actually exponential in schema order k_m and in string length times specificity $l\sigma[P]$. This would mean that XCS scales exponentially in the problem length, which is certainly highly undesirable. However, since specificity in $[P]$ decreases with larger population sizes as shown in Equation 15, we derive a general *reproductive opportunity bound (ROP-bound)* below that shows that population size needs to grow in $O(l^{k_m})$.

Figure 4 shows several settings for the reproductive opportunity bound. We show the resulting dependency of population size N on specificity $\sigma[P]$ requiring a confidence value of .99. In comparison to the covering bound, shown in Figure 2, it can be seen that the reproductive opportunity bound is always stronger than the covering bound making the latter nearly obsolete. However, the covering bound is still useful to set the don't care probability $P_\#$ appropriately. Mutation rate, and thus the specificity the population converges to, however, is bound by the reproductive opportunity bound.

General Reproductive Opportunity Bound While the above bound ensures the reproduction of *existing* classifiers that represent a particular schema, it does not assure the actual presence or generation of such a classifier. Thus, we need to *combine schema bound* and *reproductive opportunity bound*. Figure 5 shows a *control map* of certain reproductive opportunity bound values and schema bound values requiring a probability of success of 50% (plotting equations 10 and 25). The corresponding intersections denote the best value of specificity and population size to ensure supply and growth with high probability. Initial specificity may be set slightly larger than the value of the intersection to boost initial performance as long as the covering bound is not violated.

We can also quantify the interaction. Substituting the O-notated specificity bound in Equation 15 of the schema bound in the O-notated dependence of the

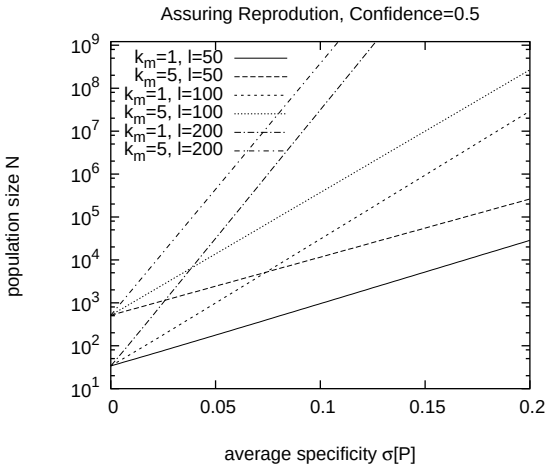

Fig. 4. To ensure successful identification and reproduction of better classifiers, population size needs to be set high enough with respect to a given specificity.

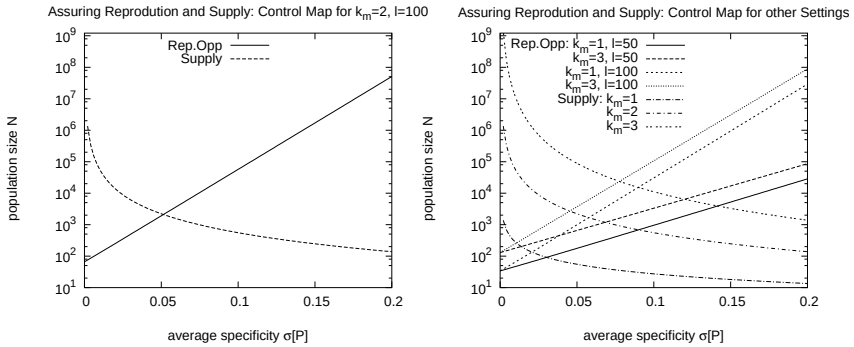

Fig. 5. The shown control maps clarify the competition between reproduction and supply. The shown boundaries assure a 50% probability of success. High specificity ensures supply but may hinder reproduction. Vice versa, low specificity ensures reproduction but lowers the probability of supply.

representative bound on string length l ($N : O(2^{l\sigma[P]})$) and ignoring additional constants, we can derive the following enhanced population size bound:

$$\begin{aligned} N &> 2^{l(\frac{n}{N})^{1/k_m}} \\ \log_2 N &> l\left(\frac{n}{N}\right)^{1/k_m} \\ N^{1/k_m}\log_2 N &> l n^{1/k_m} \\ N(\log_2 N)^{k_m} &> n l^{k_m} \end{aligned} \tag{26}$$

This general *reproductive opportunity bound* (ROP-bound) essentially shows that population size N needs to grow approximately exponentially in the minimal order of problem difficulty k_m and polynomially in the string length.

$$N : O(l^{k_m}) \tag{27}$$

Note that in the usual case, k_m is rather small and can often be set to one. Essentially, when k_m is greater than one, other classification systems in machine learning have also shown a similar scale up behavior. For example, the inductive generation of a decision tree in $C4.5$ (27) would not be able to decide on which attribute to expand first (since any expansion leads to the same probability distribution and thus no information gain) and consequently would generate an inappropriately large tree.

Sufficiently Accurate Values Although we usually assume that classifier parameter estimates are sufficiently accurate, we need to note that this assumption does not hold necessarily. All classifier parameters are only an approximation of the average value. The higher parameter β is set, the higher the expected variance of the parameter estimates. Moreover, while the classifiers are younger than $1/\beta$, the estimation values are approximated by the average of the so far encountered values. Thus, if a classifier is younger than $1/\beta$, its parameter variances will be even higher than the one for experienced classifiers.

Requiring that each offspring has the chance to be evaluated at least $1/\beta$ times to get as close to the real value as possible, the reproductive opportunity bound needs to be increased by the number of evaluations we require.

From Equation 19 and Equation 20, we can derive the expected number of steps until deletion and similarly, we can derive the expected number of evaluations during a time period t.

$$E(\texttt{\# steps until deletion}) = \frac{1}{P(\texttt{deletion})}/2 = \frac{N}{2} \tag{28}$$

$$E(\texttt{\# of evaluations in } t \texttt{ steps}) = P(\texttt{in } [A]) \cdot t = \frac{1}{n} 0.5^{l\sigma(cl)} t \tag{29}$$

The requirement for success can now be determined by requiring that the number of evaluations before deletion must be larger than some threshold Θ where Θ could for example be set to $1/\beta$.

$$\begin{aligned} E(\texttt{\# of evaluations in (\# steps until deletion)}) &> \Theta \\ \frac{1}{n} 0.5^{l\sigma(cl)} \frac{N}{2} &> \Theta \\ N &> \Theta n 2^{l\sigma(cl)+1} \end{aligned} \tag{30}$$

Setting Θ to one, Equation 30 is basically equal to Equation 25 since one evaluation is equal to at least one reproductive opportunity disregarding the confidence value in this case. It can be seen that the sufficient evaluation bound only

increases the reproductive opportunity bound by a constant. Thus, scale-up behavior is not affected.

As a last point in this section, we want to point out that fitness is actually not computed by averaging, but the Widrow-Hoff delta rule is used from the beginning. Moreover, fitness is usually set to 10% of the parental fitness value (to prevent disruption). Thus, fitness is derived from two approximations and it starts off with a disadvantage so that the early evolution of fitness strongly depends on fitness scaling and on accurate approximations of the prediction and prediction error estimates. To ensure a fast detection and reproduction of superior classifiers it is consequently necessary to choose initial classifier values as accurate as possible. Alternatively, the expected variance in fitness values could be considered to prevent potential disruption. For example, 28) suggests the usage of variance sensitive bidding. Accordingly using the estimated expectable variance, the fitness of young classifiers could be modified for selection to prevent disruption but also to enable the earlier detection of better classifiers.

Bound Verification The derived bounds are experimentally confirmed elsewhere (23; 10; 9). In essence, it was shown that that the derived bounds hold using a Boolean function problem of order of difficulty k_m. The hidden parity function (a set of k of the l attributes are evaluated by a parity (or XOR) operator), originally investigated in XCS in 29), is very suitable to manipulate k_m since at least all k relevant parity bits need to be specified to increase classification accuracy. Thus, the minimal order of problem difficulty equals the size of the parity ($k_m = k$)..

The results confirmed the computational dependency on minimal order of difficulty k_m as well as the problem length l and the chosen specificity reflected in the mutation probability μ. In particular, population size needs to grow polynomially in the string length l as well as exponential in k_m in order to assure quick and reliable learning.

4.4 Estimating Learning Time

Given that schema, covering, and reproductive opportunity bounds are satisfied, we addressed three facets of the four aspects introduced: Problem initialization is appropriate if the covering bound is satisfied. Schema supply is assured if the schema bound is respected. Growth can be assured if the reproductive opportunity bound is considered. Thus, it is assured that better classifier structures can grow in the population.

However, it was not addressed, yet, how long it may take to evolve a complete problem solution by the means of this growing process. In essence, the reproductive opportunity bound ensures that better classifiers *can* grow. This section investigates *how long* it takes to grow a complete solution from this growing process.[3]

[3] Related publications of parts of this and the following section can be found in 11).

Assuming that the other problem bounds are satisfied by the choice of mutation and population size, we can estimate how long it takes to discover successively better classifiers until the maximally general, accurate classifiers are found. To do this, we assume a domino convergence model (30) estimating the time until each relevant attribute can be expected to be specialized to the correct value. Considering mutation only, we estimate the time until reproduction and the time until generation of the next best classifier in each problem niche. Using this approach we can show that learning time scales polynomially in problem length and problem complexity.

Time Bound Derivation To derive our learning time bound, we estimate the time until reproduction of the current best classifier as well as the time until creation of the next best classifier via mutation given a reproductive event of the current best classifier. The model assumes to start with an initially completely general population (that is, $P_{\#} = 1.0$). Initial specializations are randomly introduced via mutation. Problem-specific initialization techniques or higher initial specificity in the population may speed-up learning time (as long as the covering bound is not violated).

Further assumptions are that the current best classifier is not lost and selected as the offspring when it is part of an action set (assured by the ROP-bound in conjunction with tournament selection). The time model assumes domino convergence (30) in which each attribute is successively specified. This means that only once the first attribute is correctly specified in a classifier, the second attribute influences fitness and so forth.

Using the above assumptions, we can estimate the probability that mutation correctly specifies the next attribute

$$P(\texttt{perfect mutation}) = \mu(1-\mu)^{l-1} \tag{31}$$

where l specifies the number of attributes in a problem instance. This probability can be relaxed in that we only require that the k already correctly set features are not unset (changed to don't care), the next feature is set, and we do not care about the others:

$$P(\texttt{good mutation}) = \mu(1-\mu)^{k} \tag{32}$$

Whereas Equation 31 specifies the lower bound on the probability that the next best classifier is generated, Equation 32 specifies an optimistic bound.

As seen before, the probability of reproduction can be estimated by the probability of occurrence in an action set. The probability of taking part of an action set again, is determined by the current specificity of a classifier. Given a classifier which specifies k attributes, the probability of reproduction is

$$P(\texttt{reproduction}) = \frac{1}{n}\frac{1}{2}^{k}\ , \tag{33}$$

where n denotes the number of actions in a problem. The best classifier has a minimal specificity of k/l. With respect to the current specificity in the population $\sigma[P]$, the specificity of the best classifier may be expected to be $k+\sigma[P](l-k)$

assuming a uniform specificity distribution in the other $l-k$ attributes. Taking this expected specificity into account, the probability of reproduction is

$$P(\texttt{rep.in [P]}) = \frac{1}{n}\frac{1}{2}^{k+\sigma[P](l-k)} \tag{34}$$

Since the probability of a successful mutation assumes a reproductive event, the probability of generating a better offspring than the current best is determined by:

$$P(\texttt{generation of next best cl.}) = P(\texttt{rep.in [P]})\ P(\texttt{good mutation}) = \frac{1}{n}\frac{1}{2}^{k+\sigma[P](l-k)} \mu(1-\mu)^{l-1} \tag{35}$$

Since we assume uniform sampling from all possible problem instances, the probability of generating a next best classifier conforms to a geometric distribution (memoryless property, each trial has an independent and equally probable distribution), the expected time until the generation of the next best classifier is

$$\begin{gathered} E(\texttt{time until gen.of next best cl.}) = \\ 1/P(\texttt{time until gen.of next best cl.}) = \\ \frac{1}{\frac{1}{n}\frac{1}{2}^{k+\sigma[P](l-k)}\mu(1-\mu)^{l-1}} = \frac{n2^{k+\sigma[P](l-k)}}{\mu(1-\mu)^{l-1}} \leq \frac{n2^{k+\sigma[P]l}}{\mu(1-\mu)^{l-1}} \end{gathered} \tag{36}$$

Given now a problem in which k_d features need to be specified and given further the domino convergence property in the problem, the expected time until the generation of the next best classifier can be summed to derive the time until the generation of the global best classifier:

$$\begin{gathered} E(\texttt{time until generation of maximally accurate cl}) = \\ \sum_{k=0}^{k_d-1} \frac{n2^{k+\sigma[P]l}}{\mu(1-\mu)^{l-1}} = \frac{n2^{\sigma[P]l}}{\mu(1-\mu)^{l-1}} \sum_{k=0}^{k_d-1} 2^k < \frac{n2^{k_d+\sigma[P]l}}{\mu(1-\mu)^{l-1}} \end{gathered} \tag{37}$$

This time bound shows that XCS needs an exponential number of evaluations in the order of problem difficulty k_d. As argued above, the specificity and consequently also mutation needs to be decreased indirect proportional to the string length l. In particular, since specificity $\sigma[P]$ grows as $O((\frac{n}{N})^{\frac{1}{k_m}})$ (Equation 15) and population size grows as $O(l^{k_m})$ (Equation 27), specificity essentially grows as

$$O(\frac{n}{l}) \tag{38}$$

Using the O-notation and substituting in Equation 37, we derive the following adjusted time bound making use of the inequality $(1+\frac{n}{l})^l < e^n$:

$$O\left(\frac{l2^{k_d+n}}{(1-\frac{n}{l})^{l-1}}\right) = O\left(\frac{l2^{k_d+n}}{e^{-n}}\right) = O\left(l2^{k_d+n}\right) \tag{39}$$

Thus, learning time in XCS is bound mainly by the order of problem difficulty k_d and the number of problem classes n. It is linear in the problem length l. This derivation essentially also validates Wilson's hypothesis that XCS learning time grows polynomially in problem complexity as well as problem length (15). The next section experimentally validates the derived learning bound.

Experimental Validation In order to validate the derived bound, performance was evaluated on an artificial problem in which domino convergence is forced to take place. Similar results are expected in typical Boolean function problems in which similar fitness guidance is available, such as in the layered multiplexer problem (3; 9). In other problems, additional learning influences may need to be considered such as the influence of crossover or the different fitness guidance in the problem (9).

The results in 11) as well as the further experimental evaluations in 22) confirmed the learning time bound. Essentially, the dependency on problem difficulty k_d and on mutation rate μ was confirmed. Moreover, it was shown that the population size needs to be chosen sufficiently high to satisfy the above problem bounds. Moreover, it was shown that mutating the action part as well as allowing free mutation was able to further speed-up learning. A higher GA threshold slightly delayed learning speed as expected.

4.5 Assuring Solution Sustenance: The Niche Support Bound

The above bounds assure that problem subsolutions represented by individual classifiers evolve. The time bound additionally estimates how long it takes the evolutionary process to evolve a complete problem solution. Since the time bound and all other bounds consider individual classifiers integrated in the whole population, the population as a whole is required to evolve a complete problem solution supplying, evaluating, and growing currently best subsolutions in parallel. What remains to be assured is that the final problem solution, represented by a set of maximally accurate and maximally general classifiers, can be sustained in the population. This is expressed in the sixth point of the facetwise theory approach to LCS success: Niching techniques need to assure the sustenance of a complete problem solution.

Thus, we now derive a population size bound that assures the niche support of all necessary problem subsolutions with high probability. To derive the niche support bound, we develop a simple Markov chain model of classifier support in XCS. Essentially, we model the change in niche size of particular problem subsolutions (that is, niches) using a Markov chain.

To derive the bound, we focus on the support of one niche only disregarding potential interactions with other niches. Again we assume that problem instances are encountered according to a uniform distribution over the whole problem space. Additionally, we assume random deletion from a niche. Given the Markov chain over the niche size, we then determine the steady state distribution that estimates the expected niche distribution.

Using the steady state distribution, we derive the probability that a niche is lost. This probability can be bound by minimizing this loss probability. The result is a final population size bound. The bound assures the maintenance of a low-error solution with high probability. The experimental evaluations show that the assumptions hold in non-overlapping problems. In problems that require overlapping solution representations, the population size may need to be increased further.

Markov Chain Model As already introduced for the schema bound in Section 4.2, we define a problem niche by a schema of order k. A representative of a problem niche is defined as a classifier that specifies at least all k bits correctly. The Markov chain model constructs a Markov chain over the number of classifier representatives in a particular problem niche.

Suppose we have a particular problem niche represented by k classifiers; let p be the probability that an input belonging to the niche is encountered, and let N be the population size. Since classifiers are deleted with probability proportional to their action set size estimate, a classifier will be deleted from the niche with probability k/N. Assuming that the GA is always applied (this can be assured by setting $\theta_{GA} = 0$) and disregarding any disruptive effects due to mutation or crossover, the probability that a new classifier is added to the niche is exactly equal to the probability p that an input belonging to the niche is encountered.

However, overgeneral classifiers might inhabit the niche as well so that also an overgeneral classifier might be chosen for reproduction decreasing the reproduction probability p of a niche representative. However, as shown elsewhere (22), due to the action set size relative tournament selection, the probability of selecting a niche representative for reproduction is larger than some constant dependent on the relative tournament size τ. Given that τ is chosen sufficiently large and given further that the population mainly converged to the niche representatives, the probability approaches one.

In the Markov chain model, we assume that at each time step both the GA reproduction and the deletion are applied. Accordingly, we derive three transition probabilities for a specific niche. Given the niche is currently represented by j classifiers, at each time step, (i) with probability r_j the size of the niche is increased (because a classifier has been reproduced from the niche, while another classifier has been deleted from another niche); (ii) with probability d_j the size of the niche is decreased (because genetic reproduction took place in another niche, while a classifier was deleted from this niche); (iii) with probability s_j the niche size remains constant (either because no classifier has been added nor deleted from the niche or because one classifier has been added to the niche while another one has been deleted from the same niche).

The Markov chain associated to the model is depicted in Figure 6. States in the model indicate the niche size determined by the number of representatives in a niche. Arcs labeled with r_j represent the event that the application of the GA and deletion results in an increase of the niche size. Arcs labeled with s_j represent the event that the application of the genetic algorithm and deletion

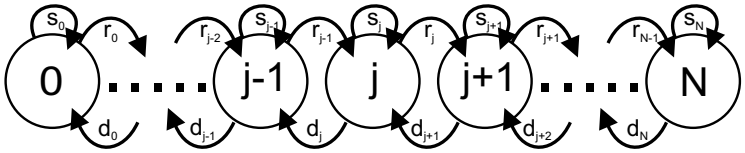

Fig. 6. Markov chain model for the support of one niche in XCS: j is the number of classifiers in the niche; N is the population size; r_j is the probability that a classifier is added to a niche containing j representatives; s_j is the probability that the niche containing j representatives is not modified through reproduction; d_j is the probability that a classifier is deleted from a niche containing j representatives.

results in no overall effect on the niche size. Arcs labeled with d_j represent the event that the application of the genetic algorithm and deletion results in a decrease of the niche size.

More formally, since the current problem instance is part of a particular niche with probability p, a niche representative will be generated via GA reproduction with approximately probability p. Assuming random deletion, a representative of a niche is deleted with probability j/N since there are by definition j representatives in the current population of size N. Accordingly, we compute the probabilities r_j, s_j, and d_j as follows:

$$r_j = p\left(1 - \frac{j}{N}\right) \tag{40}$$

$$s_j = (1-p)\left(1 - \frac{j}{N}\right) + p\frac{j}{N} \tag{41}$$

$$d_j = (1-p)\frac{j}{N} \tag{42}$$

For $j = 0$ we have $r_0 = p$, $s_0 = 1 - p$, and $d_0 = 0$. When $j = 0$, the niche is not represented in the population, therefore: (i) when an input belonging to the niche is presented to the system (with probability p), one classifier is generated through *covering*, therefore $r_0 = p$; (ii) since the niche has no classifiers, deletion cannot take place, therefore $d_0 = 0$; finally, (iii) the probability that the niche remains unrepresented is $1 - r_0 - s_0$, that is $s_0 = 1 - p$. Similarly, when $j = N$ all the classifiers in the population belong to the niche, accordingly: (i) no classifier can be added to the niche, therefore $r_N = 0$; (ii) with probability p an input belonging to the niche is encountered so that a classifier from the niche is reproduced while another one from the niche is deleted, leaving the niche size constant, therefore $s_N = p$; (iii) when an input that does not belong to the niche is presented to the system (with probability $1-p$), a classifier is deleted from the niche to allow the insertion of the new classifier to the other niche, therefore $d_N = 1 - p$. Thus, for $j = N$ we have $r_N = 0$, $s_N = p$, and $d_N = 1 - p$.

Note that our approach somewhat brushes over the problem of overgeneral classifiers in that overgeneral classifiers are not considered as representatives of any niche. In addition, covering may not be sufficient in the event of an empty

niche since overgeneral classifiers might still be present so that $r_0 = p$ is an approximation. However, as pointed out in 31), as long as a sufficiently large population size is chosen, chopping off or approximating the quasi absorbing state r_0 approximates the distribution accurately enough. This is also confirmed by our experimental investigations. However, overgeneral classifiers and more importantly overlapping classifiers can influence the distribution as evaluated below. Given the above assumptions, we are now able to derive a probability distribution over niche support.

Steady State Derivation To estimate the distribution over the number of representatives of a problem niche, we derive the distribution when the Markov chain is in steady state. Essentially, we derive probabilities u_j that the niche has j representatives. To derive the steady state distribution, we first write the fixed point equation for our Markov chain:

$$u_j = r_{j-1}u_{j-1} + s_j u_j + d_{j+1}u_{j+1},$$

which equates the probability that the niche has j representatives with the probability that the niche will have j representatives in the next time step. In the next time step, three events will contribute to the probability of having j representatives: (i) reaching state j from state $j-1$, with probability $r_{j-1}u_{j-1}$, by means of a reproductive event; (ii) remaining in state j with probability $s_j u_j$; (iii) reaching state j from state $j+1$, with probability $d_{j+1}u_{j+1}$, by means of a deletion event. The same equation can be rewritten by acknowledging the fact that in steady state the incoming proportion needs to be equal to the outgoing proportion in each state in the Markov chain:

$$(r_j + d_j)u_j = r_{j-1}u_{j-1} + d_{j+1}u_{j+1} \tag{43}$$

Replacing d_{j+1}, d_j, r_j, and r_{j+1} with the actual values from the previous section (equations 40, 41, and 42) we get the following:

$$\left[p\left(1 - \frac{j}{N}\right) + \frac{j}{N}(1-p)\right] u_j = (1-p)\left(\frac{j+1}{N}\right) u_{j+1} + p\left(1 - \frac{j-1}{N}\right) u_{j-1} \tag{44}$$

Equation 44 is a second order difference equation whose parameters are dependent on j, i.e., on the current state. We use Equation 44 and the condition:

$$\sum_{j=0}^{j=N} u_j = 1 \tag{45}$$

to derive the steady state distribution. Multiplying Equation 44 by $N/((1-p)u_{j-1})$, we derive the following:

$$\left[\frac{p}{1-p}(N-j) + j\right] \frac{u_j}{u_{j-1}} = (j+1)\frac{u_{j+1}}{u_{j-1}} + \frac{p}{1-p}(N-j+1) \tag{46}$$

To derive the steady state distribution of probabilities u_j we use Equation 46 to derive an equation for the ratio $\frac{u_j}{u_0}$. Next, we use the equation for $\frac{u_j}{u_0}$ and the condition in Equation 45 to derive the steady state distribution.

As the very first step, we write the following fixed point equation for the transition between state 0 and state 1

$$u_0 = s_0 u_0 + d_1 u_1 \tag{47}$$

substituting the values of r_0 and d_1 we obtain:

$$\begin{aligned} u_0 &= (1-p)u_0 + (1-p)\frac{1}{N}u_1 \\ pu_0 &= (1-p)\frac{1}{N}u_1 \end{aligned} \tag{48}$$

from which we derive:

$$\frac{u_1}{u_0} = \frac{p}{1-p}N \tag{49}$$

To derive the equation for u_2/u_0 we start from Equation 46 and set $j = 1$:

$$\left[\frac{p}{1-p}(N-1)+1\right]\frac{u_1}{u_0} = 2\frac{u_2}{u_0} + \frac{p}{1-p}N \tag{50}$$

so that,

$$\frac{u_2}{u_0} = \frac{1}{2}\left[\left(\frac{p}{1-p}(N-1)+1\right)\frac{u_1}{u_0}\frac{p}{1-p}N\right] \tag{51}$$

We replace u_1/u_0 with Equation 49:

$$\begin{aligned} \frac{u_2}{u_0} &= \frac{1}{2}\left[\left(\frac{p}{1-p}(N-1)+1\right)\frac{u_1}{u_0}\frac{p}{1-p}N\right] \\ &= \frac{1}{2}\left[\left(\frac{p}{1-p}(N-1)+1\right)\frac{p}{1-p}N\frac{p}{1-p}N\right] \\ &= \frac{1}{2}\left[\frac{p}{1-p}(N-1)\frac{p}{1-p}N\right] \\ &= \frac{N(N-1)}{2}\left(\frac{p}{1-p}\right)^2 \\ &= \binom{N}{2}\left(\frac{p}{1-p}\right)^2 \end{aligned} \tag{52}$$

This leads us to the hypothesis that

$$\frac{u_j}{u_0} = \binom{N}{j}\left(\frac{p}{1-p}\right)^j, \tag{53}$$

which we prove by induction. Using Equation 53, we can first derive that

$$u_{j+1} = \frac{N-j}{j+1}\frac{p}{1-p}u_j \tag{54}$$

$$u_j = \frac{N-j+1}{j}\frac{p}{1-p}u_{j-1} \tag{55}$$

$$u_{j-1} = \frac{1-p}{p}\frac{j}{N-j+1}u_j. \tag{56}$$

With Equation 46 substituting Equation 56 as the inductive step, we now derive

$$\begin{aligned} u_{j+1} &= \frac{\left(\left(\frac{p}{1-p}(N-j)+j\right)\frac{u_j}{u_{j-1}} - \frac{p}{1-p}(N-j+1)\right)u_{j-1}}{j+1} \\ &= \frac{\left(\frac{p}{1-p}\right)^2 \frac{(N-j)(N-j+1)}{j}u_{j-1}}{j+1} \\ &= \frac{N-j}{j+1}\frac{p}{1-p}u_j, \end{aligned} \tag{57}$$

which proves the hypothesis.

We can now derive the steady state distribution from Equation 45 dividing both sides by u_0

$$\sum_{j=0}^{N}\frac{u_j}{u_0} = \frac{1}{u_0}, \tag{58}$$

substituting Equation 53 we derive

$$\begin{aligned} \sum_{j=0}^{N}\frac{u_j}{u_0} &= \sum_{j=0}^{N}\binom{N}{j}\left(\frac{p}{1-p}\right)^j \\ &= \left[\sum_{j=0}^{N}\binom{N}{j}p^j(1-p)^{N-j}\right]\frac{1}{(1-p)^N}, \end{aligned} \tag{59}$$

where the term $\sum_{j=0}^{N}\binom{N}{j}p^j(1-p)^{N-j}$, equals to $[p+(1-p)]^N$", that is 1, so that:

$$\sum_{j=0}^{N}\frac{u_j}{u_0} = \frac{1}{(1-p)^N}, \tag{60}$$

accordingly,

$$u_0 = (1-p)^N \tag{61}$$

Finally, combining Equation 53 and Equation 61, we derive the steady state distribution over u_j as follows:

$$\begin{aligned} u_j &= \binom{N}{j} \left(\frac{p}{1-p}\right)^j u_0 \\ &= \binom{N}{j} \left(\frac{p}{1-p}\right)^j (1-p)^N \\ &= \binom{N}{j} p^j (1-p)^{N-j} \end{aligned} \tag{62}$$

Note that the same derivation is possible noting that the proposed Markov chain results in an *Engset* distribution (32).

Essentially, we see that the constant probability of reproduction p in combination with a linear increasing probability of deletion j/N, results in a binomial distribution over niche support sizes in steady state. In the next sections we validate Equation 62 experimentally, and evaluate the assumptions made such as the influence of mutation, overgeneral classifiers, the r_0 approximation, and overlapping niches.

Evaluation of Niche Support Distribution Evaluations were undertaken in three Boolean function problems including (1) the layered count ones problem, (2) the multiplexer problem, and (3) the carry problem. While the layered count ones problem requires a non-overlapping solution representation, the multiplexer problem allows overlapping subsolutions. The carry problem requires an overlapping solution representation. The experimental evaluations in (12; 22) confirmed the Markov chain model and the resulting niche support bound. Hereby, the chosen selection mechanism slightly influences the result in that tournament selection sometimes facilitated the maintenance of overlapping niches (in the multiplexer problem). In the carry problem where the final subsolutions are overlapping it was possible to confirm the support of one macro-niche that included a class of overlapping niches. Due to the overlap, the support for one niche was smaller that if it was not overlapping. Thus, as mentioned, dependent on the degree of overlap, population size needs to be increased further. Increasing the GA threshold decreases the niche size distribution since the threshold prevents over-reproduction of a frequently occurring niche,

In sum, it was shown that solution spaces interfere with each other during selection in problems that require an overlapping solution representation. The overlap causes a decrease in niche sizes. However, the influence was not as significant as originally feared. Further extensions to balance the niches are imaginable such as taking into consideration the degree of overlap among competing (fitness sharing) classifiers. Nonetheless, the model is able to predict the general behavior of XCS's final solution. Additionally, the model can be used to estimate the probability of a niche loss. The next paragraphs derive this probability and extend the model to a general population size bound that ensures the maintenance of a low-error solution with high probability.

Population Size Bound The reported results show that our model for niche support in XCS can predict the distribution of niche size for problems involving non-overlapping niches. Effectively, our model provides an asymptotic prediction of niche support distribution. It applies once the problem has been learned and there is no further influence from genetic operators. Besides such predictive capabilities, we can use our model to derive a population size bound that ensures that a complete model is maintained with high probability. In particular, from the probability u_0 we can derive a bound to guarantee that a niche is not lost with high probability.

In essence, the model can approximate the probability that a niche is lost. Using Equation 62, we can derive a bound for the population size N that ensures with high probability that XCS does not loose any of the problem niches, that is, any subsolutions. From the derivation of the probability of being in state u_0 (which means, that the respective niche was lost), which is $u_0 = (1-p)^N$, we see that the probability of loosing a niche decreases exponentially with the population size. Given a problem with 2^k problem niches, that is, the perfect solution $[O]$ (33) is represented by 2^k schemata of order k, the probability of loosing a niche equates $u_0 = \left(1 - \frac{1}{2^k}\right)^N$.

Requiring a certainty θ that no niche will be lost (that is, $\theta = 1 - u_0$), we can derive a concrete population size bound

$$N > \frac{\log(1-\theta)}{\log(1-p)} > \frac{\log(1-\theta)}{\log(1-\frac{1}{2^k})}, \tag{63}$$

effectively showing that population size N grows logarithmically in the confidence value and polynomially in the solution complexity 2^k. Figure 7 shows the population size bound that assures niche support. Since the population size scales as the inverse of the probability of niche occurrence, the $\log - \log$-sale shows a straight line.

Thus, the bound confirms that once a problem solution was found, XCS is able to maintain the problem solution with high probability requiring a population size that grows polynomially in solution complexity and logarithmically in the confidence value. This bound confirms that XCS does not need more than a polynomial population size with respect to the solution complexity consequently pointing to the PAC learning capability of XCS confirming Wilson's original hypothesis (15).

4.6 Towards PAC Learnability

The derivations of the problem bounds in the previous sections enables us to connect learning in the XCS classifier system to fundamental elements of computational learning theory (COLT). COLT is interested in showing how much computational power an algorithm needs to learn a particular problem. To derive an overall computational estimate of XCS's learning capabilities, we focus

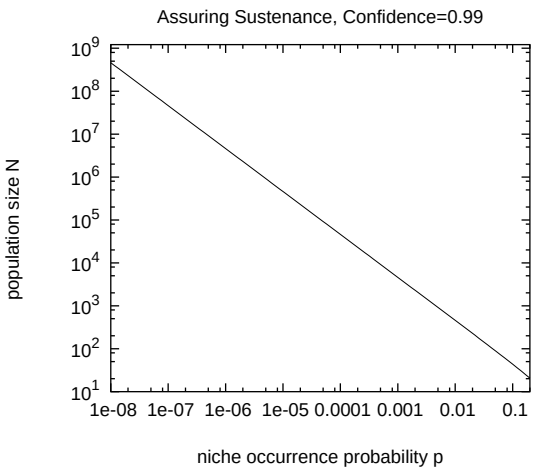

Fig. 7. To sustain a complete solution with high probability, population size needs to grow as the inverse of the niche occurrence probability p. In a uniformly sampled, binary problem, p corresponds to $1/2^k$ where k specifies the minimal number of attributes necessary to do maximally accurate predictions in all solution subspaces.

on the problem of learning *k-DNF* functions. In particular, we show that *k-DNF* problems that satisfy few additional properties are PAC-learnable (13; 14) by XCS. In essence, we also confirm Wilson's previous conjecture that XCS scales polynomially in time and space complexity (15).

XCS is certainly not the most effective *k-DNF* learning algorithm. 16) shows that an algorithm especially targeted to solve noise-free uniformly sampled *k-DNF* problems is able to reach a much more effective performance. However, this thesis does not only show that XCS is able to PAC-learn *k-DNF* problems. Rather, it shows that it is able to learn a large variety of problems including nominal and real-valued problems, noisy problems, as well as general RL problems. Restricting the problem to *k-DNF* problems, we can show that XCS is a PAC-learning algorithm confirming the effectiveness as well as the generality of XCS's learning mechanism.

To approach the PAC-learning bound, we reflect on the previous bounds evaluating their impact on computational complexity. The successive chapters provide a variety of evidence for XCS's successful and broad applicability as well as its effective learning and scale-up properties.

Problem Bounds Revisited In Section 3, we analyzed how the evolutionary pressures in XCS bias learning towards the evolution of a complete, maximally accurate, and maximally general problem solution ensuring *fitness guidance* and *appropriate generalization*. This chapter investigated the requirement on population size and learning time in order to supply better classifiers, make time to detect and grow those better classifiers until the final population is reached, and

finally, to sustain the final problem solution with high probability. Satisfying these bounds, we can ensure with few additional constraints that XCS learns the underlying problem successfully.

We now revisit the bounds considering their resulting computational requirement.

Covering Bound. The covering bound ensures that the GA is taking place establishing a covering probability (Equation 6). To ensure a high probability of covering, the specificity can be chosen very low by setting the initial specificity (controlled by $P_\#$) as well as mutation rate sufficiently low. Given a fixed specificity that behaves in $O(\frac{n}{l})$ as necessary to supply better classifiers, as derived above (Equation 38), the population size can be bounded as follows using the approximation $x < -\log(1-x)$:

$$\frac{-\log(1-P(\texttt{cov.}))}{-\log\left(1-\left(\frac{2-\sigma[P]}{2}\right)^l\right)} < \frac{-\log(1-P(\texttt{cov.}))}{\left(1-\frac{n}{2l}\right)^l} < -\log(1-P(\texttt{cov.}))e^{n/2} < N(64)$$

Thus, to satisfy the covering bound, the population size needs to grow logarithmically in the probability of error and exponentially in the number of problem classes n. With respect to PAC-learnability, the bound shows that to ensure that the GA is successfully applied in XCS with probability $1-\delta_P$ (where $\delta_P = (1-P(\texttt{cov.}))$ the population size scales logarithmically in the error probability δ_P as well as exponentially in the number of problem classes.

Schema Bound. The *schema bound* ensures that better classifiers are available in the population. Given a problem with a minimal order of difficulty k_m and requiring a high probability that representatives of this order are supplied (Equation 8), we were able to bound the population size N in Equation 10 showing that population size N needs to grow logarithmically in the probability of error δ_P (that is, $\delta_P = 1 - P(\texttt{rep.exists})$) and exponentially in the order of the minimal order complexity k_m given a certain specificity and thus polynomial in concept space complexity.

Reproductive Opportunity Bound. In addition to the existence of a representative, we showed that it is necessary to ensure reproduction and thus growth of such representatives. This is ensured by the general reproductive opportunity bound which was shown to require a population size growth of $O(l^{k_m})$ (Equation 27) with respect to the minimal order complexity k_m of a problem. The reproductive opportunity bound was generated with respect to one niche. However, since XCS is evolving the niches in parallel and the probability of niche occurrence as well as the probability of deletion underly a geometric distribution (memoryless property and approximately equal probabilities), we can assure with high confidence $1-\delta_P$, that all relevant niches receive reproductive opportunities. Thus, we can assure with high probability that lower-order representatives grow leading to higher accuracy within a complexity that is polynomial in the concept space complexity.

Time Bound. The time bound estimates the number of problem instances necessary to learn a complete problem solution using XCS. Given a problem that requires classifiers of maximal schema order k_d (a k_d-conjunction in a *k-DNF*) and given further the domino convergence property in the problem, the expected time until generation of an optimal classifier was approximated in Equation 37 yielding a population size requirement of $O(l2^{k_d+n})$ (Equation 39). The estimation approximates the expected time til creation of the best classifier of a problem niche of order k_d. As in the reproductive opportunity bound, we can argue that since XCS evolves all problem niches in parallel and since the generation of the next best classifier underlies a geometric distribution (memoryless property and equal probability), given a certain confidence value δ, the time until a particular classifier of order k is generated with high confidence δ grows within the same limits. Similarly, assuming a probability p of problem occurrence, we can bound the time requiring a maximal error in the final solution ϵ with low probability δ by $O(l\frac{1}{\epsilon}2^n)$.

Niche Support Bound. To ensure the support of the final solution, we finally established the niche support bound. Given a problem whose solution is expressed as a disjunction of distinct *subsolutions*, XCS tends to allocate distinct rules to each *subsolution*. To ensure a complete problem solution, it needs to be assured that all subsolutions are represented with high probability. Deriving a Markov model over the number of representatives for a particular niche, we were able to derive the steady state niche distribution given a niche occurrence probability p (Equation 62).

Requiring that all niches with at least niche occurrence probability p are expected to be present in the population with high probability, we were able to derive a bound on the population size N requiring that the probability of no representative in a niche with more than p occurrence probability is sufficiently low. With respect to PAC-learnability this bound requires that with high probability $1-\delta$ we assure that our solution has an error probability of less than ϵ (where ϵ is directly related to p). Using this notation, we can derive the following equation from Equation 61 substituting ϵ for p and δ for u_0 using again the approximation $x < -\log(1-x)$:

$$\frac{\log \delta}{\log(1-\epsilon)} < -\frac{1}{\epsilon}\log\frac{1}{\delta} < N \tag{65}$$

This bound essentially bounds the population size showing that it needs to grow logarithmically in $\frac{1}{\delta}$ and linear in $\frac{1}{\epsilon}$. Approximating ϵ by $(\frac{1}{2})^{k_d}$ assuming a uniform problem instance distribution, we see that to prevent niche loss, the population size needs to grow linearly in the concept space complexity 2^{k_d}.

PAC-Learning with XCS With the bounds above, we are now able to bound computational effort and number of problem instances necessary to evolve with high probability $(1-\delta)$ a low error ϵ solution of an underlying *k-DNF* problem.

Additionally, the *k-DNF* problem needs to be maximally of minimal order of difficulty k_m as discussed in sections 4.2 and 4.3.

Thus, Boolean function problems in *k-DNF* form with l attributes and a maximal order of problem difficulty k_m are PAC-learnable by XCS using the ternary representation of XCS conditions. That is, XCS evolves with high probability $(1 - \delta)$ a low error ϵ solution of the underlying *k-DNF* problem in time polynomial in $1/\delta$, $1/\epsilon$, l, and concept space complexity.

The bounds derived in Section 4.6 show that the computational complexity of XCS, which is bounded by the population size N, is linear in $1/\delta$, $1/\epsilon$ and l^{k_m}. Additionally, the time bound shows that the number of problem instances necessary to evolve a low-error solution with high probability grows linearly in $1/\delta$, $1/\epsilon$, l and the solution complexity. Consequently, we showed that Boolean functions that can be represented in *k-DNF* and have a maximal order of problem difficulty k_m, are PAC-learnable by XCS using the ternary representation of XCS conditions as long as the assumptions in the bound derivations hold.

The following further assumptions about the interaction of the bounds have been made. First, crossover is not modeled in our derivation. While crossover can be disruptive as already proposed in 4), crossover may also play an important innovative role in recombining currently found subsolutions effectively as proposed in 19) and experimentally confirmed for XCS in 9). The next chapter provides a more detailed investigation on the impact of crossover.

Second, the specificity derivation from the mutation rate assumes no actual fitness influence. Subtle interactions of various niches might increase specificity further. Thus, problems in which the specificity assumption does not hold might violate the derived reproductive opportunity bound.

Third, if the probability of reproduction p is approximated by $(\frac{1}{2})^{k_d}$, niche support assumes a non-overlapping representation of the final solution. Thus, overlapping solution representations require an additional increase in population size as evaluated in Section 4.5.

5 Summary and Conclusions

This chapter showed *when* XCS is able to learn a problem. Along our facetwise theory approach to LCSs, we derived population size specificity, and time bounds that assure that a complete, maximally accurate, and maximally general problem solution can evolve and can be sustained.

In particular, we derived a *covering bound* that bounds population size and specificity to ensure proper XCS initialization making way for classifier evaluation and GA application. Next, we derived a *schema bound* that bounds population size and specificity to ensure *supply* of better classifiers. Better classifiers were defined as classifiers that have higher accuracy on average. They can be characterized as *representatives* of minimal order schemata or BBs—those BBs, that increase classification accuracy in the problem at hand. Next, we derived a *reproductive opportunity bound* that bounds specificity and population size to assure *identification* and *growth* of better classifiers. The subsequently derived

time bound estimates the learning time needed to evolve a complete problem solution given the other bounds are satisfied. Finally, we derived a *niche bound* that assures the *sustenance* of a low-error solution with high probability.

Along the way, we defined two major problem complexities: (1) the minimal order complexity k_m, which specifies the minimal number of features that need to be specified to decrease class entropy (that is, increase classification accuracy), and (2) the general problem difficulty k_d, which specifies the maximal number of attributes necessary to specify a class distribution accurately. While the former is relevant for supply and growth, the latter is relevant for the sustenance of a complete problem solution.

Putting the bounds together, we showed that XCS can *PAC-learn* a restricted class of *k-DNF* problems. However, the reader should keep in mind that XCS is an online generalizing, evolutionary-based RL system and is certainly not designed to learn *k-DNF* problems particularly well. In fact, XCS can learn a much larger range of problems including DNF problems but also multistep RL problems as well as datamining problems as validated in subsequent chapters.

Before the validation, though, we need to investigate the last three points of our facetwise LCS theory approach for single-step (classification) problems. Essentially, it needs to be investigated if search via mutation and recombination is effective in XCS and how XCS distinguishes between local and global solution structure. The next chapter consequently considers problems which are hard for XCS's search mechanism since whole BB structures need to be processed to evolve a complete problem solution. We consequently improve XCS's crossover operator using statistical methods to detect and propagate dependency structures (BBs) effectively.

Acknowledgment

We are grateful to Xavier Llorà, Kei Onishi, Kumara Sastry, and the whole IlliGAL lab for their help and the useful discussions. We are also in debt to Stewart W. Wilson who initiated the analysis of XCS with the covering and schema considerations. 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] Holland, J.H.: Adaptation. In Rosen, R., Snell, F., eds.: Progress in theoretical biology. Volume 4. Academic Press, New York (1976) 263–293

[2] Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. In Waterman, D.A., Hayes-Roth, F., eds.: Pattern directed inference systems. Academic Press, New York (1978) 313–329

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

[4] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975) second edition, 1992.

[5] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)

[6] Bernadó, E., Llorà, X., Garrell, J.M.: XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems (LNAI 2321). Springer-Verlag, Berlin Heidelberg (2002) 115–132

[7] Dixon, P.W., Corne, D.W., Oates, M.J.: A preliminary investigation of modified XCS as a generic data mining tool. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321). Springer-Verlag, Berlin Heidelberg (2002) 133–150

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

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

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

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

[12] Butz, M.V., Goldberg, D.E., Lanzi, P.L., Sastry, K.: Bounding the population size to ensure niche support in XCS. IlliGAL report 2004033, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (2004)

[13] Valiant, L.: A theory of the learnable. Communications of the ACM **27** (1984) 1134–1142

[14] Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston, MA (1997)

[15] Wilson, S.W.: Generalization in the XCS classifier system. Genetic Programming 1998: Proceedings of the Third Annual Conference (1998) 665–674
[16] Servedio, R.A.: Efficient Algorithms in Computational Learning Theory. PhD thesis, Harvard University, Cambridge, MA (2001)
[17] Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. Soft Computing **6** (2002) 144–153
[18] Lanzi, P.L.: Learning classifier systems from a reinforcement learning perspective. Soft Computing: A Fusion of Foundations, Methodologies and Applications **6** (2002) 162–170
[19] Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Boston, MA (2002)
[20] Kovacs, T.: Strength or Accuracy: Credit Assignment in Learning Classifier Systems. Springer-Verlag, Berlin Heidelberg (2003)
[21] Butz, M.V., Sastry, K., Goldberg, D.E.: Tournament selection in XCS. Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003) (2003) 1857–1869
[22] Butz, M.V.: Rule-based evolutionary online learning systems: Learning bounds, classification, and prediction. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL (2004)
[23] Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.W.: How XCS evolves accurate classifiers. Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 927–934
[24] Holland, J.H.: Processing and processors for schemata. In Jacks, E.L., ed.: Associative Information Techniques, New York, American Elsevier (1971) 127–146
[25] Butz, M.V., Pelikan, M.: Analyzing the evolutionary pressures in XCS. Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001) (2001) 935–942
[26] Butz, M.V., Goldberg, D.E.: Bounding the population size in XCS to ensure reproductive opportunities. Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003) (2003) 1844–1856
[27] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA (1993)
[28] Goldberg, D.E.: Probability matching, the magnitude of reinforcement, and classifier system bidding. Machine Learning **5** (1990) 407–425
[29] Kovacs, T., Kerber, M.: What makes a problem hard for XCS? In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996). Springer-Verlag, Berlin Heidelberg (2001) 80–99
[30] Thierens, D., Goldberg, D.E., Pereira, A.G.: Domino convergence, drift, and the temporal-salience structure of problems. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence, New York, NY, IEEE Press (1998) 535–540

[31] Horn, J.: Finite Markov chain analysis of genetic algorithms with niching. Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 110–117

[32] Kleinrock, L.: Queueing Systems: Theory. John Wiley & Sons, New York (1975)

[33] Kovacs, T.: XCS classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. In Roy, Chawdhry, Pant, eds.: Soft computing in engineering design and manufacturing. Springer-Verlag, London (1997) 59–68