Problem Solution Sustenance in XCS: Markov Chain Analysis of Niche Support Distributions and the Impact on Computational Complexity

Martin V. Butz[⋆], David E. Goldberg[‡], Pier Luca Lanzi[†], Kumara Sastry[‡]

[⋆]Department of Cognitive Psychology
University of Würzburg
Röntgenring 11, 97070 Würzburg, Germany
butz@psychologie.uni-wuerzburg.de

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

[†]Artificial Intelligence and Robotics Laboratory
Dipartimento di Elettronica e Informazione
Milano 20133, Italy
pierluca.lanzi@polimi.it

Abstract

Michigan-style learning classifier systems iteratively evolve a distributed solution to a problem in the form of potentially overlapping subsolutions. Each problem *niche* is covered by subsolutions that are represented by a set of predictive rules, termed classifiers. The genetic algorithm is designed to evolve classifier structures that together cover the whole problem space and represent a complete problem solution. An obvious challenge for such an online evolving, distributed knowledge representation is to continuously sustain all problem subsolutions covering all problem niches, that is, to ensure *niche support.* Effective niche support depends both on the probability of reproduction and on the probability of deletion of classifiers in a niche. In XCS, reproduction is occurrence-based whereas deletion is support-based. In combination, niche support is assured effectively. In this paper we present a Markov chain analysis of the niche support in XCS, which we validate experimentally. Evaluations in diverse Boolean function settings, which require non-overlapping and overlapping solution structures, support the theoretical derivations. We also consider the effects of mutation and crossover on niche support. With respect to computational complexity, the paper shows that XCS is able to maintain (partially overlapping) niches with a computational effort that is linear in the inverse of the niche occurrence frequency.

Keywords: learning classifier systems, LCS, XCS, niching, Markov chain analysis, solution sustenance, mutation

1 Introduction

Michigan-style learning classifier systems (LCSs) are rule-based, evolutionary online-learning systems that can be applied to a wide range of learning problems including classification problems, function approximation problems, and reinforcement learning problems. LCSs learn online from one problem instance at a time. They evolve a solution represented by a population of condition-action-prediction rules, called *classifiers.* Problem knowledge is represented by a set of subsolutions, which should cover the whole problem space. A problem subspace in which one maximally accurate and maximally general subsolution applies is referred to as a *problem niche.* Fundamental learning challenges in LCSs are the evolution and the sustenance of all necessary problem subsolutions. This paper investigates the problem of sustaining a complete problem solution and thus essentially covering all problem niches in a problem focusing on classification problems.

The XCS classifier system [49] is known to perform well on various classification problems including general data mining problems [1, 2, 21] as well as deterministic and noisy Boolean function problems [15]. So far, several problem bounds have been identified: population size bounds that ensure problem coverage [16], population size bounds that ensure initial support and reproductive opportunities [15], and a learning time bound that estimates the time until a complete solution is found using a domino-convergence model [13]. All results confirm Wilson's original hypothesis that XCS's learning complexity scales polynomially in problem complexity and problem size [50].

Our analysis of niche sustenance in XCS allows us to derive a population size bound that ensures the sustenance of a complete solution to the underlying problem given a certain *niche occurrence frequency.* Hereby, niche occurrence frequency refers to the frequency with which problem instances from a certain environmental niche (a specific subspace in the problem) are sampled. The analysis enables us also to estimate the minimal niche occurrence frequency for which XCS is able to lean and sustain a subsolution with high probability given a certain (fixed) population size. Although the number and complexity of the niches necessary to evolve a complete and accurate final problem solution cannot be known beforehand in the general case, the derived population size bound can guarantee that XCS will maintain a complete problem solution for problems *up to* a certain problem complexity. The complexity is strongly dependent on niche size and niche occurrence frequency.

We first propose a Markov chain model to approximate niche sizes. Next, we derive the corresponding steady state equation that estimates the distribution of niche sizes. The model depends on niche occurrence frequency and the mutation type and rate used. We validate the model in non-overlapping problems, in which the experimental evaluation matches the model nearly perfectly. We also consider problems in which overlapping problem solutions are evolved. In such types of problems, additional competitive effects need to be considered. The experimental evidence confirms the theoretical model. Further niching influences due to additional XCS mechanisms, parameters, and problem properties are analyzed as well.

The model developed in this paper adds a new facet to the analysis of XCS: solution sustenance. The derived population sizing bound oftentimes dictates the success of XCS learning so that the developed model is a critical component for designing scalable XCS systems. In general, the objectives of this paper are threefold: (1) To understand the solution sustenance mechanisms in the XCS classifier system; (2) To derive a consequent population size bound that ensures niche support with high probability given a certain problem complexity; (3) To emphasize the importance of solution sustenance in the learning classifier system framework in general suggesting the necessary analysis of solution sustenance in any type of LCS.

The paper is organized as follows. Section 2 introduces the XCS classifier system. Section 3 introduces the Markov model for niche support and Section 4 derives the closed form for the niche support distribution in steady state. Section 5 introduces the approach to experimental evaluation and Section 6 and 7 experimentally evaluate the niche support equations. Section 8 relates the results to the general learning complexity in XCS. The paper ends with a discussion of the related work (Section 9), contributions of this work (Section 10), and future directions (Section 11).

2 The XCS Classifier Systems

The accuracy-based classifier system XCS was introduced by Wilson [49]. The system was further improved by Wilson [50] and Kovacs [37, 38]. We now provide a concise description of XCS focused on its knowledge representation and niche genetic algorithm approach. For more details we refer the interested reader to the original paper [49] and to the algorithmic description [18].

2.1 Knowledge Representation

As outlined in the introduction, XCS evolves a distributed knowledge representation expressed by a *population* (set) of *classifiers* (rules). As outlined by Kovacs [36, 38], XCS is designed to evolve a non-overlapping, optimal problem solution represented by an optimal set $[O]$ of accurate, maximally general classifiers. This paper investigates the maintenance of this optimal set $[O]$ under continual application of the evolutionary discovery component.

Rules, or classifiers, in XCS consist of a condition, which specifies when the classifier is applicable, an action, which specifies the suggested action or classification, and four main parameters: (i) the prediction R, which estimates the relative payoff that the system expects when the classifier is used, (ii) the prediction error ε, which estimates the mean absolute deviation of the prediction R, (iii) the fitness F, which estimates the average relative accuracy of the payoff prediction derived from ε, and (iv) the numerosity *num*, which indicates how many copies of *micro-classifiers* with the same condition and the same action are present in the population. Additionally, each classifier has a time stamp *ts* that records the last GA application in a classifier set that the classifier was part of, and the action set size estimate *as*, which estimates the average action set size the classifier is part of. The parameter is updated similarly to the reward prediction R.[1]

2.2 Performance Component

At each time step, XCS builds a *match set* [M] containing the classifiers in the population [P] whose conditions match the current sensory input. If [M] contains less than θ_{mna} actions, *covering* takes place and creates a new classifier that matches the current inputs and has a random action. For each possible action a_i in [M], XCS computes the *system prediction* $P(a_i)$, which estimates the payoff that XCS expects if action a_i is performed. The *system prediction* is computed as the fitness weighted average of the predictions of classifiers in [M], $cl \in$[M], which advocate action a_i (i.e., $cl.a{=}a_i$):

$$P(a_i) = \frac{\sum_{cl_k \in [M]|_{a_i}} R_k \cdot F_k}{\sum_{cl_k \in [M]|_{a_i}} F_k} \tag{1}$$

where, $[M]|_{a_i}$ represents the subset of classifiers of $[M]$ with action a_i, R_k identifies the prediction of classifier cl_k, and F_k identifies the fitness of classifier cl_k. Using the computed system prediction, XCS selects an action using some behavioral policy (e.g. during knowledge exploitation, it chooses the action that promises the maximum payoff). The classifiers in [M] that advocate the selected action form the current *action set* [A]. The selected action is performed in the environment and a scalar reward r is returned indicating the correctness, or quality, of the executed action. In classification this is usually 1000 for a correct action and 0 for an incorrect action. Next, the subsequent problem instance is received.

2.3 Reinforcement Component

Once the reward r is received the parameters of the classifiers in [A] are updated in the following order [18]: prediction, prediction error, and finally fitness. In classification problems, classifier prediction R is updated with learning rate β $(0 \leq \beta \leq 1)$:

$$R \leftarrow R + \beta(r - R). \tag{2}$$

Next, the prediction error ε is updated as: $\varepsilon \leftarrow \varepsilon + \beta(|r - R| - \varepsilon)$. Both parameters are usually updated using the *moyenne adaptative modifiée* technique [46], which averages the encountered values as long as the resulting update is larger than the updated caused be learning rate β (that is, the number of updates encountered so far is smaller than $1/\beta$).

Fitness F is updated in three steps. First, the *raw accuracy* κ of the classifiers in [A] is computed as:

$$\kappa \quad = \quad \begin{cases} 1 & \text{if } \varepsilon \leq \varepsilon_0 \\ \alpha(\varepsilon/\varepsilon_0)^{-\nu} & \text{otherwise} \end{cases} \tag{3}$$

The *raw accuracy* κ is used to calculate the *relative accuracy* κ' as:

$$\kappa' \quad = \quad \frac{(\kappa \cdot num)}{\sum_{cl \in [A]}(cl.\kappa \cdot cl.num)} \tag{4}$$

[1]All parameters used in the paper are listed in Appendix A.

where $cl.\kappa$ is the *raw accuracy* of classifier *cl*, as computed in equation 3; *cl.num* is the *numerosity* of classifier *cl*. Finally the *relative accuracy* κ' is used to update the classifier fitness as:

$$F \leftarrow F + \beta(\kappa' - F) \tag{5}$$

Thus, the fitness estimates the average relative accuracy of this (macro-)classifier, which consists of *num* identical micro-classifiers.

2.4 Discovery Component

If the mean time since the previous GA application to the classifiers in action set $[A]$ is larger than θ_{GA}, genetic reproduction is applied in $[A]$. Two classifiers are selected with probability *proportional to their fitness*. (The recently introduced tournament selection is not applied in the experiments herein. However, comparisons show that the results also hold for tournament selection [12].) Genetic reproduction generates two offspring classifiers performing crossover with probability χ and mutation with probability μ. Fitness in the offspring classifiers is usually decreased by 0.1 compared to the parents.

The resulting offspring are inserted into the population and two classifiers are deleted to keep the population size constant. Deletion is done deleting classifiers proportional to their action set size estimate *as*. Additionally, we investigate Kovacs' deletion addition [37] that increases the deletion probability of a classifier further if it is (1) experienced ($exp > \theta_{del}$) and (2) its fitness F is smaller than a fraction of the current average fitness $\bar{F}$ in the population ($F < \delta\bar{F}$).

Thus, while reproduction is constrained to the current problem niche, which is essentially determined by the current problem instance, deletion is applied to the whole population. Effectively, the probability of reproduction in a particular problem niche is constrained by the niche occurrence (controlled by the probability of occurrence of a problem instance belonging to that niche) whereas the probability of deletion is mainly constrained by the current niche size. This insight forms the basis for the theory developed in the following section.

3 Markov Chain Model for Niche Support

We now develop a Markov chain model for niche support in XCS. Essentially, we assume a Markov chain based on the niche size of one particular niche, solely dependent on the niche occurrence probability p. The model assumes a number of simplifications to enable the mathematical analysis. First, we focus on the support of one niche only and initially neglect interactions with other niches. Second, we assume that the genetic algorithm is always applied ($\theta_{GA} = 0$) and that its application can only add to or delete from a niche at most one classifier. Third, we assume that inputs are encountered according to a uniform distribution Finally, we approximate classifier deletion with the uniform probability.

Subsequently, we relax several of these constraints. In particular, we consider niche interactions caused by mutation experimentally and analytically. Moreover, we investigate the biases caused by XCS's GA application threshold θ_{GA} as well as by the enhanced classifier deletion technique, in which deletion is proportional to classifiers' action set size estimates and is biased on their fitness estimates.

Before introducing our model, we first need to define a *problem niche* and a *representative of a problem niche*. We define a problem niche using the notion of a schema [29] in genetic algorithms and in classifiers [30]. Given a binary classification problem of length l, a schema defines the relevant attributes in the problem for a particular problem niche. Additionally, an action needs to be specified. The number of specified positions is termed the *order* o of the schema. This actually corresponds to the notation of the condition parts in LCSs where the number of specialized attributes (i.e. the non-don't care symbols) corresponds to the order.

In the spirit of Holland's definition [30], we define a *schema representative* as a classifier that represents at least all o specified attributes in the schema and specifies the correct action [15]. For example, given a schema `01*0*`, the following classifier conditions are representatives of this schema: `01#0#`, `0100#`, `0110#`, `01#00`, `01000`, `01100`, `01#01`, `01001`, and `01101`. Similarly, a particular problem instance is part of a particular

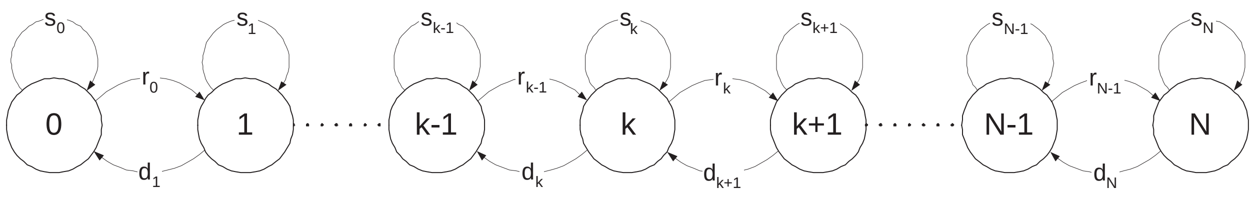


Figure 1: Markov chain model for the support of one niche in XCS: k is the number of classifiers in the niche; N is the maximum population size; r_k is the probability that a classifier is added to a niche represented by k classifiers; s_k is the probability that the niche containing k classifiers is not modified by reproduction; d_k is the probability that a classifier is deleted from a niche containing k classifiers.

schema, if all specified attributes in the schema are equal to the corresponding attributes in the problem instance. Thus, in our example problem instances `01000`, `01100`, `01001`, and `01101` are part of schema `01*0*`.

Suppose we have a particular problem niche represented by k classifiers. If we neglect additional deletion biases, the probability of deleting one of the classifier representatives from a niche is k/N. Suppose that the probability of encountering a problem instance that is part of the niche is equal to p. Given that XCS learned the problem, we can assume that the action set resulting from the problem instance and an (e.g. random) action choice consists solely of representatives of the currently relevant niche. Thus, given the GA is applied, the probability that a representative is reproduced equals p. Ignoring potential disruptive effects by mutation for now, the probability of generating an offspring that is a representative of the niche in question also equals to p. We also assume that both genetic reproduction and deletion are applied in each iteration. This is always the case once the population size reaches its maximum size N, which is true when a solution was found except for in the most trivial problems.

With these assumptions, we can now derive the probabilities that the size of a particular niche, currently represented by k classifiers, increases, decreases, or stays the same. At each time step, (i) with probability r_k the size of the niche increases because a classifier was added by the genetic algorithm, while another classifier was deleted from another niche; (ii) with probability s_k the niche size remains constant either because no classifier was added nor deleted from the niche or because one classifier was added to the niche while another one was deleted from the same niche; (iii) with probability d_k the size of the niche decreases because genetic reproduction was applied to another niche, while a classifier was deleted from this niche to keep the population size constant.

According to the above derivations, probabilities r_k, s_k, and d_k can be computed as follows:

$$r_k = p\left(1-\frac{k}{N}\right) \tag{6}$$

$$s_k = (1-p)\left(1-\frac{k}{N}\right)+p\frac{k}{N} \tag{7}$$

$$d_k = (1-p)\frac{k}{N} \tag{8}$$

The consequent Markov chain is depicted in Figure 1. States in the model represent the different niche support sizes. State labels indicate the number of representatives of a niche. Arcs correspond to increasing, maintaining, and decreasing the number of representatives of the niche in question, labeled according to the above equations with r_k, s_k, and d_k, respectively.

In the limiting case when $k = 0$ we have $r_0 = p$, $s_0 = 1-p$, and $d_0 = 0$. When $k = 0$, 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), a classifier will be generated through *covering* (assuming there are no over-general classifiers present), therefore the approximation $r_0 = p$ is appropriate; (ii) since the niche has no classifiers, deletion cannot take place, therefore $d_0 = 0$; finally, (iii) the probability that the niche will remain unrepresented is $1 - r_0$, that is $s_0 = 1 - p$. Similarly, for $k = N$ we have $r_N = 0$, $s_N = p$, and $d_N = 1 - p$.

When $k = N$, 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 will be presented, a classifier from the niche will be reproduced, while another one from the niche will be 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 will be deleted from the niche to allow the insertion of the new classifier to the other niche, therefore $d_N = 1 - p$.

Note that over-general classifiers are not considered as representatives of any niche. The experimental investigations will show how over-general classifiers can further influence the resulting niche support distributions. In addition, covering is assumed to create a (non-over-general) representative so that $r_0 = p$ is an approximation. However, as outlined elsewhere [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. Given the above assumptions, we are now able to derive a probability distribution over niche support.

4 Closed-Form Solution

We now derive a closed-form solution for the Markov model derived in the previous section. Basically, we compute for each possible state k the probability u_k that the niche will have k representatives. As the very first step, we write the following fixed point equation [43] for the Markov chain in Figure 1:

$$u_k = r_{k-1}u_{k-1} + s_k u_k + d_{k+1}u_{k+1},$$

which expresses the probability u_k that the niche has k classifiers (i.e., of being in state k) as the union of three events: (i) reaching state k from state $k - 1$, with probability $r_{k-1}u_{k-1}$, by means of a reproductive event; (ii) remaining in state k with probability $s_k u_k$; (iii) reaching state k from state $k+1$, with probability $d_{k+1}u_{k+1}$, by means of a deletion event. We can rewrite the same equation as:

$$(r_k + d_k)u_k \quad = \quad r_{k-1}u_{k-1} + d_{k+1}u_{k+1},$$

which equates the probability of leaving state k, computed as $(r_k + d_k)u_k$ to the probability of entering state k, computed as $r_{k-1}u_{k-1} + d_{k+1}u_{k+1}$. By replacing d_{k+1}, d_k, r_k, and r_{k+1} with the actual values we obtain:

$$\left[p\left(1-\frac{k}{N}\right)+\frac{k}{N}\left(1-p\right)\right]u_k = p\left(1-\frac{k-1}{N}\right)u_{k-1} + (1-p)\left(\frac{k+1}{N}\right)u_{k+1} \tag{9}$$

where p is the probability that an input belonging to the niche will be encountered, N is the population size, and k is the niche size. Equation 9 is a second order difference equation whose parameters are dependent on k, i.e., on the current state. We use Equation 9 and the condition for a probability distribution:

$$\sum_{k=0}^{k=N} u_k = 1$$

to derive the following closed-form equation for probability u_k (see Appendix B for details):

$$u_k = \binom{N}{k} p^k (1-p)^{N-k} \tag{10}$$

The equation shows that the constant probability of reproduction p in combination with a linear increasing probability of deletion k/N, results in a binomial distribution over niche support sizes in steady state.

In the next sections we validate Equation 10 experimentally. We also consider the influence of mutation on niche support, which is strongly problem dependent. Moreover, we investigate the impact of over-general classifiers, the r_0 approximation, additional deletion biases, overlapping niches, and the impact of parameter θ_{GA}.

5 Design of the Experimental Validation

To validate the proposed Markov model for niche support, we perform two sets of experiments. In the first set of experiments (Section 6), we apply XCS to a class of problems, the Boolean multiplexer, whose solutions involve non-overlapping niches. In the second set of experiments (Section 7), we extend previous results and apply XCS to a class of problems, the carry function, whose solutions involve overlapping niches. For each experiment, we collect the statistics regarding the size of all the niches, and compare the distribution of niche sizes estimated from the experiments with the model distribution provided by Equation 10.

5.1 Design of Experiments

Each experiment consists of a set of trials in which XCS is applied to solve a task with a specific setting. Each trial consists of a number of problems that XCS must solve. At the end of each trial we measure the niche sizes as the number of micro-classifiers that belong to each niche. As outlined in Section 3, we represent a niche by a schema and an action; for example, the term "`000***:0`" identifies the niche containing classifiers whose condition represent the schema `000***` and that have specified action `0`. A classifier condition represents a schema, if it is at least as specific as the schema. Accordingly, a classifier with condition `000##1` and action `0` belongs to the niche identified by "`000***:0`"; while a classifier with condition `#00##1` (regardless of the action) does not. The distribution of the niche size values collected throughout the trials is used to estimate the values u_k for every k, that is, to estimate the density function over niche sizes. All the experiments were conducted on `xcslib` [39] and were duplicated using Butz's implementation [10].

5.2 Design of Statistical Analysis

To analyze the experimental results reported, we use various statistical procedures involving both (i) simple descriptive statistics (e.g., mean and standard error), (ii) parametric and nonparametric statistics to compare the mean of the distributions derived from the experiments with the mean predicted by Equation 10; and also (iii) nonparametric statistics to compare the distribution of niche sizes from the experiments with the distribution provided by the model. More specifically, for each experiment, we provide a table showing the following descriptive statistics: the sample mean and standard error (column "$\overline{x}{\pm}s$"), and the sample median (column "$\overline{x}_{0.5}$"); the mean μ and standard deviation σ as predicted by the model (column "$\mu \pm \sigma$"). To compare the sample mean with the model mean we also provide the p-value for the (parametric) *one sample t-test for unequal variance* (column `tt`) and the p-value for the (non-parametric) one sample *Wilcoxon test* (column `wt`). These two statistical procedures test the null hypothesis H_0 that the mean of the experimental data $\overline{x}$ is equal to the mean of the theoretical distribution μ (i.e., $H_0 : \overline{x} = \mu$), against the alternative hypothesis H_1 that the sample mean and the model mean differ (i.e., $H_1 : \overline{x} \neq \mu$).

To test whether the distribution of the experimental data is *compatible* with a binomial distribution we use *Fisher's Binomial Dispersion* test [22], and report the corresponding *p-value* in the table as "`bdt`". To test whether the distribution of the experimental data actually fits the theoretical distribution provided by the model we employ the *goodness of fit for discrete distributions* introduced in [23], and report the corresponding p-value in the table as column "`gof`".

To compare the distribution of niche sizes estimated from XCS, with that predicted by the model, we apply both the two-sided and the one-sided Kolmogorov-Smirnov test [19]. To accomplish this, we first generate a sample from the model distribution provided by Equation 10 of the same size as the data extracted by the experiments. Then for each experiment, we apply three different versions of the *two independent samples* Kolmogorov-Smirnov test. In particular, we apply the two-sided Kolmogorov-Smirnov (K), that tests the hypothesis H_0 that for every value x the empirical cumulative distribution function[2] estimated from the experiments ($F_{XCS}(x)$) and the cumulative distribution function of the model ($F_M(x)$) are the same (i.e., $\forall x \in (-\infty, +\infty) F_{XCS}(x) = F_M(x)$) against the alternative hypothesis H_1 that $F_{XCS}(x) \neq F_M(x)$ for at least one value of x. We apply the one-sided Kolmogorov-Smirnov test (K^+) to test the hypothesis $H_0 : \forall x \in$

[2]We remind the reader that given a random variable X the cumulative distribution function $F(x)$ is defined as $F(x) = P(X \leq x)$.

column	description
$\overline{x}$	sample mean
s	standard error
$\overline{x}_{0.5}$	sample median
$\mu \pm \sigma$	average and standard deviation for the model
bdt	p-value for *Fisher's Dispersion Test* for the binomial distribution
tt	p-value for the one-sample *t-test* for unequal variance
wt	p-value for the one-sample *Mann-Whitney* test
gof	p-value for the χ^2 goodness of fitness test for the Binomial distribution
K^+	p-value for the one-way Kolmogorov-Smirnov test $H_0 : F_{XCS}(x) \geq F_M(x)$
K	p-value for the two-way Kolmogorov-Smirnov test $H_0 : F_{XCS}(x) = F_M(x)$
K^-	p-value for the one-way Kolmogorov-Smirnov test $H_0 : F_{XCS}(x) \leq F_M(x)$

Table 1: Statistics reported for experiments discussed in this paper.

$(-\infty, +\infty) : F_{XCS}(x) \geq F_M(x)$ against the hypothesis $H_1 : \exists x : F_{XCS}(x) < F_M(x)$. Finally, we apply the one-sided Kolmogorov-Smirnov test (K^-) to test the hypothesis $H_0 : \forall x \in (-\infty, +\infty) : F_{XCS}(x) \leq F_M(x)$ against the hypothesis $H_1 : \exists x : F_{XCS}(x) > F_M(x)$. For each of tests ($K$, K^+, and K^-) we report the corresponding p-value in the table. Note that in all these three tests, the null hypothesis H_0 is rejected as soon as it does not hold for *one* value. Accordingly, we shall expect that in most of the cases the null hypotheses will be rejected.

In Table 1 we provide a reference of all the statistics that we will report for the experiments discussed. Note that although the Kolmogorov-Smirnov test should be applied to continuous distributions, the test is *conservative* when applied to discrete distributions; in particular, when applied to discrete distributions the true p-value is 1/3 of the computed p-value, see [19] for details. All the statistical tests reported in this paper were conducted using the R environment for statistical computing [45].

6 Non-overlapping Subsolutions

In the first set of experiments we apply XCS to a problem involving non-overlapping niches. We compare the empirical niche size distributions with the theoretical niche size distributions predicted by the model in Equation 10. We show the partially disruptive effects of mutation and we investigate the influence of action set size-based deletion as well as fitness-based deletion. Finally, we show that results also hold for larger values of the GA activation threshold θ_{GA}.

6.1 Boolean Multiplexer

This class of problems has been widely studied in the learning classifier system literature (see for example [34, 47, 49]). Boolean multiplexer are defined for strings of $l = k + 2^k$ bits. The first k bits, $x_0, \ldots, x_{k-1}$, represent an address which indexes the remaining 2^k bits, $x_k, \ldots, x_{k+2^k-1}$; the function returns the value of the indexed bit. For instance, in the 6-multiplexer function, mp_6, we have that $mp_6(100010) = 1$ while $mp_6(000111) = 0$. More formally, the 6-multiplexer can be represented by the following disjunctive normal form:

$$mp_6(x_0, x_1, y_0, \ldots, y_3) = \overline{x_0}\ \overline{x_1} y_0 \vee\ \overline{x_0}\ x_1 y_1 \vee x_0\ \overline{x_1} y_2 \vee x_0 x_1 y_3$$

For a Boolean multiplexer of size l, with k address bits, the size of the optimal population $[O]$ in XCS is 2^{k+2} since XCS represents the expected payoff for the correct as well as the incorrect action. For instance, the optimal solution for the 11-multiplexer (i.e., $l = 11$ and $k = 3$) consists of 32 (i.e., 2^{3+2}) classifiers.

In this section, we validate our model using the 11-multiplexer. More difficult multiplexer instances up to the multiplexer 70 were solved with XCS [16]. The purpose of the following evaluations in the smaller

0000*******:0	100****0***:0	0000*******:1	100****0***:1
0001*******:0	100****1***:0	0001*******:1	100****1***:1
001*0******:0	101*****0**:0	001*0******:1	101*****0**:1
001*1******:0	101*****1**:0	001*1******:1	101*****1**:1
010**0*****:0	110******0*:0	010**0*****:1	110******0*:1
010**1*****:0	110******1*:0	010**1*****:1	110******1*:1
011***0****:0	111*******0:0	011***0****:1	111*******0:1
011***1****:0	111*******1:0	011***1****:1	111*******1:1

Table 2: Niches for the 11-multiplexer; each niche is represented by a schema and an action.

multiplexer instance is to validate the derived model and investigate further XCS parameter influences.[3] Successful XCS applications to larger Boolean function problems as well as to real-world data mining problems can be found elsewhere [16, 1, 2, 21].

6.2 Accurate Model Approximation

We applied XCS to the 11-multiplexer with three different population sizes, $N = 500$, $N = 1000$, and $N = 1500$. Since in our model we assume that the genetic algorithm is applied at each time step, we set $\theta_{GA} = 0$. All other parameters are set as usual (see [18] for details): $\beta = 0.2$; $\theta_{mna} = 2$; $\alpha = 0.1$; $\varepsilon_0 = 1$; $\nu = 5$; $\chi = 0.8$, $\theta_{del} = 20$; $\delta = 0.1$; GA-subsumption is on with $\theta_{sub} = 20$; while action-set subsumption is off. For each value of N, we run 1000 trials, each consisting of 100,000 problems. To eliminate the effects of crossover and mutation, 50,000 steps of condensation [50] are performed at the end of each trial. During condensation the genetic algorithm is acting but crossover and mutation are turned off. Thus, exactly as hypothesized in our model, in condensation only selection and deletion are applied. From the final population, we measure the size of the 32 problem niches (that is, the number of representatives for each problem niche).

Table 2 shows the 32 niches that need to be covered by classifier representatives for a complete, maximally-accurate, maximally-general, non-overlapping problem solution for the 11-multiplexer, in correspondence to Kovacs' definition of an optimal set $[O]$ of classifiers [36]. As previously outlined, each niche is represented by a schema and an action. Since there are 32 niches of the same size and we sample problem instances uniformly randomly, the probability p that a particular niche is encountered is 1/32.

Figure 2a reports (i) with dashed lines, the three theoretical distributions provided by our Markov model (Equation 10) when $p = 1/32$, and $N = 500$, $N = 1000$, $N = 1500$; (ii) with solid lines, the distributions of niche sizes estimated through the experiments when $N = 500$, $N = 1000$, and $N = 1500$. Since all 32 niches have the same p, the curves reported are obtained by measuring, for each of the 1000 trials, the size of all the 32 possible niches. Thus, the empirical distributions are obtained from 32000 data entries. As Figure 2a shows, when condensation is applied at the end of the experiments, the empirical and theoretical distributions match almost completely.

The data in Figure 2a, are also reported in Figure 2b as *empirical cumulative distribution functions* (ECDFs), obtained from the experimental data, and *cumulative distribution functions* (CDFs) predicted by the model. As the curves in Figure 2b show that also ECDFs and CDFs almost overlap. The analysis of final populations shows that in the vast majority of the cases populations contain exactly the 32 necessary classifiers that represent the niches in Table 2. A closer look shows that the model's CDFs overestimate the ECDFs for small values of the niche size. Then ECDFs and CDFs cross so that for larger values of niche size, the model (CDFs) underestimate actual XCS behavior (ECDFs).

Table 3 summarizes the results of the statistical analysis collected for all three trials. The table shows that according to the t-test, `tt`, we cannot reject the null hypothesis, that is, we cannot determine whether the sample mean and the model mean differ. The empirical data are *compatible* with a binomial distribution (because of the high value of `bdt`). The Kolmogorov-Smirnov tests (columns K^+, K, and K^-) show that

[3]The same analysis on the even smaller 6-multiplexer is available in [14].

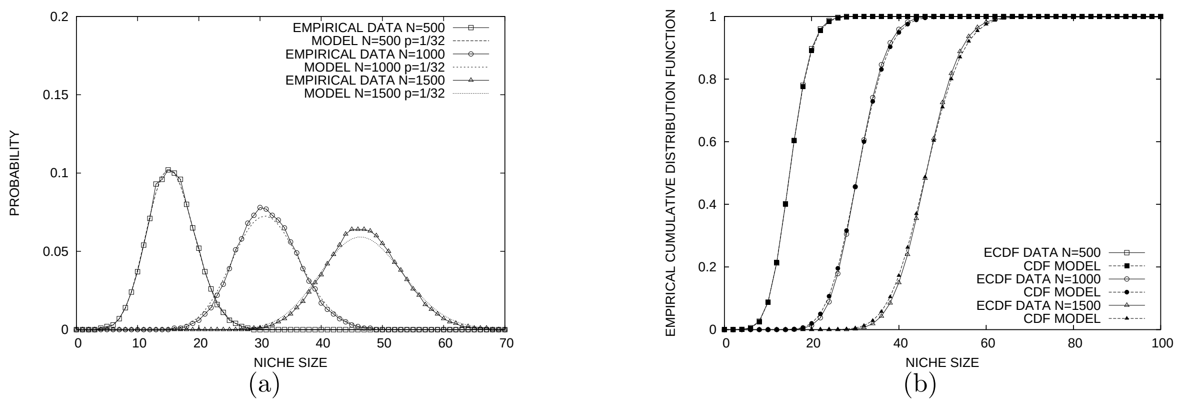


Figure 2: Distributions and corresponding cumulative distributions of niche sizes for XCS with 50,000 condensation steps in the 11-multiplexer (solid line) and for the model provided by Equation 10 (dashed line) when $\theta_{GA} = 0$, the population sizes are 500, 1000, 1500, and 50,000 condensation steps are performed at the end of each trial.

`fig`	N	$\overline{x} \pm s$	$x_{0.5}$	$\mu \pm \sigma$	`bdt`	`tt`	`wt`	`gof`	K^+	K	K^-
2	500	15.61 ± 3.89	15.00	15.62 ± 3.89	0.431	0.490	0.000	0.00	0.699	0.994	0.741
	1000	31.25 ± 5.17	31.00	31.25 ± 5.50	1.000	0.997	0.017	0.00	0.000	0.000	0.001
	1500	46.87 ± 6.19	47.00	46.88 ± 6.74	1.000	0.935	0.080	0.00	0.000	0.000	0.000
3	500	13.70 ± 3.67	14.00	15.62 ± 3.89	0.105	0.000	0.000	0.00	1.000	0.000	0.000
	1000	28.52 ± 5.00	28.00	31.25 ± 5.50	1.000	0.000	0.000	0.00	1.000	0.000	0.000
	1500	43.71 ± 5.97	44.00	46.88 ± 6.74	1.000	0.000	0.000	0.00	1.000	0.000	0.000

Table 3: XCS in the 11-multiplexer when $\theta_{GA} = 0$. Statistics for the distribution of niche sizes depicted in Figure 2 and Figure 3. Column `fig` indicates the figure to which the statistics refer to and N is the population size. Table 1 specifies the meanings of the other columns.

none of the three hypotheses can be rejected for $N = 500$; while for $N = 1000$ and $N = 1500$ all the hypotheses are rejected, so that for $N = 1000$ and $N = 1500$ we can state that empirical distributions and model distributions are significantly different. The χ^2 goodness of fitness test (`gof`) does not confirm a complete match. This effect can be explained by the focusing effect on the action-set size proportionate deletion in XCS. Currently, overrepresented niches are more likely to encounter deletions whereas currently underrepresented niches are less likely. The result is a slight focusing effect around the mean also observable in Figure 2 especially for the case with larger population sizes. Since deletion is proportional to the action set size estimates as, which are updated iteratively, this additional pressure is noisy and very hard to model. Nonetheless, the results show that the influence on niche size distribution is minor.

In sum, our statistical analysis confirms the very close match of the theory with the actual empirical values. When the effects of mutation and crossover are switched off, the results validate the modeled Markov chain behavior of niche support. The expected mean support is obtained and the resulting niche sizes are Binomially distributed. The exact fit cannot be rejected in the case of $N = 500$ according to the Kolmogorov-Smirnov tests.

6.3 The Influence of Mutation

The introduced Markov chain model does not take into account the impact of mutation and crossover on the niche size distribution. However, when a representative of a niche is reproduced, mutation may disrupt the structure so that the offspring may actually not be a representative of the current niche. This disruptive effect of mutation depends on the type of mutation used as well as on the mutation rate but it is also dependent on the order of the schema that characterizes a problem niche. Thus, the amount of disruption

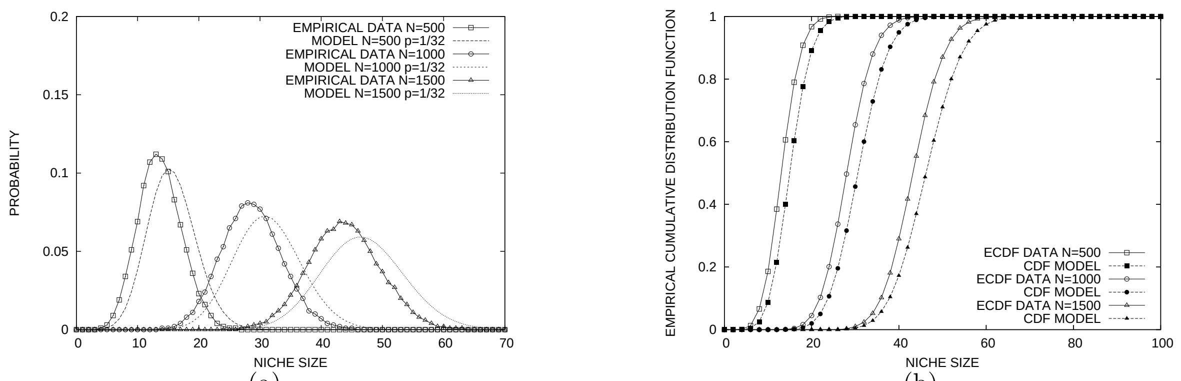


Figure 3: XCS in the 11-multiplexer when $\theta_{GA} = 0$. Distributions and corresponding cumulative distributions of niche sizes for XCS without condensation steps (solid line) and for the model provided by Equation 10 (dashed line) when $\theta_{GA} = 0$ and the population sizes are 500, 1000, 1500.

is also problem dependent.

6.3.1 Offspring Disruption due to Mutation

To analyze the effects of mutation in the 11-multiplexer we repeat the previous set of experiments without the final 50,000 condensation steps. Figure 3a reports (i) with dashed lines the three theoretical distributions provided by our Markov model (Equation 10) when $p = 1/32$, and $N = 500$, $N = 1000$, $N = 1500$; (ii) with solid lines the distributions of niche sizes estimated through the experiments when $N = 500$, $N = 1000$, and $N = 1500$; as in the previous experiments, the empirical distributions are obtained from 32000 data entries.

As Figure 3a shows, without condensation (crossover and mutation apply) the curves corresponding to the theoretical distribution and the curves estimated from the experimental data are still highly overlapping. Since mutation may result in classifier offspring that is not a representative of any niche, the classifier distributions shown in Figure 3a suggest the existence of over-general classifiers that do not belong to any of the 32 niches reported in Table 2 (e.g., `00#00######:0`). As a result, the size of the niches in Figure 3a is smaller than predicted by the model, so that the empirical distributions are shifted to the left. This observation is confirmed by the empirical cumulative distributions in Figure 3b. As the curves show, when mutation and crossover are involved the model cumulative distribution functions (CDFs) always underestimates the empirical cumulative distribution functions (ECDFs) derived from the experiments (Figure 3b).

Table 3 summarizes the statistics collected in the three settings. When mutation and crossover are involved, the mean of the empirical distributions is significantly different from that of the model, the p-values for `tt` and `wt` are 0. The Kolmogorov-Smirnov tests (columns K^+, K, and K^-) show that: (i) we accept the null hypothesis for K^+ for all values of N, that is, the ECDF is greater than the CDF ($\forall x : F_{XCS}(x) \geq F_M(x)$); (ii) we can reject the null hypothesis for K that is, the ECDF and the model CDF are significantly different.

6.3.2 Quantification of Disruptive Events

Can we quantify the observed disruptive effects of mutation and crossover? It seems that two of our assumptions are violated when mutation and crossover continue to be applied: (1) Mutation can cause disruption so that given a representative is chosen for reproduction the offspring may not necessarily be a representative. (2) Due to this disruption, over-general classifiers stay in the population. Thus, the probability of selecting a representative for reproduction is smaller than the probability of niche occurrence. Both effects lower the probability of generating a new niche representative, expressed in the niche reproduction probability p.

However, mutation and crossover are not only disruptive. Creative effects of mutation and crossover can be expected in that reproduced classifiers might become representatives of a niche due to specializations and value variations caused by mutation or due to the recombination of relevant condition substructures. However, once the problem is learned and mainly representatives are reproduced, the probability of not selecting a representative for reproduction becomes small. Thus, creational effects of mutation and crossover diminish since mainly representatives are selected for reproduction. More technically, once the optimal solution is found, the probability of selecting a representative for reproduction will be close to one so that the probability of selecting a non-representative and creating a representative out of it will be very small.

Disruptive Mutation. The potentially disruptive effects of mutation depends both (i) on the rate and type of (e.g., niche or free; see [16]) mutation involved and (ii) on the problem solution structure, that is, the schema structure of the necessary subsolutions. Mutation is disruptive if a niche representative is selected and the resulting offspring classifier is not a representative of the current niche anymore. Two cases need to be distinguished: (1) the resulting classifier becomes a representative of another niche; (2) the resulting classifier is over-general and thus not a representative of any niche. These disruptive effects depend on the structure of the problem solution space. Therefore, we now analyze these events for the 11-multiplexer.

In the 11-multiplexer, a selected representative classifier becomes a representative of another niche if its index attribute is flipped from zero to one or vice versa or if the action is mutated. Since the occurrence probability for this event is equal for all niches in the 11-multiplexer, the overall p is not influenced by such an event. However, the classifier may become either over-general or overlapping with another niche by either generalizing any of the four relevant attributes in the 11-multiplexer or by flipping one of the three address attributes from zero to one (or vice versa), respectively.

We investigate the effect of mutation for two different types of mutation [16]. The first one is *niche mutation*, in which a specialized attribute is always mutated to a don't care symbol and a don't care symbol to the current bit value in the problem instance (as defined in the algorithmic description of XCS in [18]). The second one is *free mutation*, in which an attribute is mutated to one of the two other possible values with equal probability.

In the case of niche mutation, with mutation rate μ, the probability of disruption in the 11-multiplexer equals $1-(1-\mu)^4$ since the generalization of any of the four relevant bits results in an over-general classifier. In the case of free mutation, the probability of disruption equals $1-(1-\mu)^3(1-\mu/2)$ since a mutation of any of the three attributes results most likely in a classifier that is not a representative and only the generalization of the index attribute results in disruption. Note that we slightly simplify the analysis in that we disregard potentially creative effects of mutation by, for example, the mutation of two relevant attributes. Moreover, we disregard crossover effects. Finally, we disregard specialization effects of mutation, which can create representatives that are only $1/2^{aspec}$ times part of a niche where *aspec* denotes the number of additionally specialized attributes compared to the corresponding schema.

Probability of Non-Representative Reproduction. The estimation of the probability of selecting a classifier for reproduction that is not a representative is much harder to derive mathematically. It depends on the expected number of non-representatives, which depends on the disruptive and creative effects of mutation and crossover as well as (recursively) on the current number of non-representatives. Moreover, the probability of selecting a non-representative depends on current classifier fitness values, which directly depend on classifier age, the taken fitness scaling approach, the dynamics in the population, and the problem sampling. Thus, it seems nearly impossible to estimate the probability of selecting a representative mathematically accurately and an approximation is necessary.

However, it is possible to consider the final population and derive the average probability of selecting a representative for reproduction from that population. To investigate the disruptive effects, we derive the probability of selecting a representative for reproduction by determining the average fitness proportion of representatives in the final population for each niche as follows:

$$\frac{\sum_{\texttt{cl is representative}} .5^{aspec(cl)} cl.F}{\sum_{\texttt{cl matches schema}} .5^{aspec(cl)} cl.F} \tag{11}$$

Hereby, a classifier matches a schema if all its specialized attributes correspond to the specialized (non "$*$" symbols) attributes in the schema. The operator $cl.aspec$ is used to determine the number of additionally specified attributes in the classifier cl, that is, all specialized (non don't care) attributes in the classifier that are denoted by a "$*$" in the investigated schema. The addition of this proportion assures that the classifiers that participate only sometimes in the niche under investigation are accounted for appropriately.

By multiplying the probability of selecting a representative with the probability of maintaining the representative (no disruption due to mutation), we effectively derive the expected additional disruption due to mutation. Thus, the new probability p of generating an offspring representative can be adjusted by multiplying it with this probability. Since the expected mean of the binomial distribution equals Np, the expected mean decreases when additionally considering the probability of selecting a representative and not disrupting it by mutation.

6.3.3 Experimental Evaluation

To validate the proposed effects of mutation on the sustenance of the final population, we repeat the previous experiments in the 11-multiplexer for both types of mutation (niche mutation and free mutation) and for four different types of deletion: (1) random deletion, (2) action set size deletion, (3) random deletion with the additional fitness bias [37], and (4) action set size deletion with the additional fitness bias [37].

Table 4 reports the results for all eight sets of experiments. For each experiment, Table 4 reports the derived proportions of mismatch between (i) the mean observed niche support and (ii) the theoretically derived niche support. In particular, columns "$\overline{x} \pm s$" and "$\mu \pm \sigma$" denote the actual and theoretical mean and standard deviation in the niche size distribution, without considering mutation. Column "*mut.maint.*" denotes the (mutation type dependent) probability that mutation is not disruptive; column "*sel.maint.*" denotes the probability (derived from the final population using Equation 11) that a selected classifier is a representative; column "*theor.result*" is the product of these two probabilities, that is, the probability that a representative will be reproduced and will not be disrupted; column "*empir.result*" lists the actual empirical fraction of the mean of representatives (dividing $\overline{x}$ by μ), that is, the actual supply of representatives compared to the mean supply expected by the Markov chain model without considering mutation; column "*t.r./e.r.*" denotes the fraction between the theoretical result and the experimental result, that is, the fraction between the theoretically expected fraction of representatives and the empirically observed fraction of representatives. This measure effectively denotes the accuracy of our model when we consider the additional disruptive effects of mutation.

As can be noted, Table 4 shows that if random deletion or action set size-based deletion is applied (as in the original XCS implementation [49]), the incorporation of disruption via selection and disruption via mutation yields a near perfect match compared to the observed mean support (column *t.r./e.r.*). The close match with the theory clearly confirms the theorized disruptive effects of mutation.

When Kovacs' [37] additional fitness-based deletion bias is included, the approximation does not hold anymore. Due to the additional bias of deleting low fitness classifiers, over-general classifiers are detected and deleted much faster than in the former case. The impact of this effect increases within higher population sizes since then over-general classifiers are more likely to survive sufficiently long so that fitness-based deletion can apply. The addition of the action set size estimate bias does not alter this effect significantly.

The results confirm that mutation and offspring selection can disrupt the model approximation. While mutation depends on problem type, mutation type and mutation rate, classifier fitness estimates and fitness distributions depend not only on the problem type but also highly on the fitness derivation itself. This derivation is dependent on population dynamics as well as on the exact choice of parameter settings and operator implementation. To name a few, fitness estimates depend on the exact choice of offspring parameter initialization, the learning rate β, the usage of the moyenne adaptive modifiée technique, the type and exact implementation of subsumption deletion, and the choice of fitness scaling, specified in parameters α, ϵ_0, and ν. Thus, a derivation of the exact influence of the fitness bias in deletion and the additional bias due to the action set size estimate would not only be highly problem type and problem sampling dependent but also highly dependent on very particular details of the XCS implementation. Nonetheless, the study of fitness-based deletion showed that the additional bias is beneficial for the detection of over-general, inaccurate

Random deletion

m.type	N	$\overline{x} \pm s$	$\mu \pm \sigma$	mut.maint.	sel.maint.	theor.result	empir.result	t.r./e.r.
	500	12.69± 4.204	15.62±3.89	.8493	.9610	.8162	.8124	1.005
niche	1000	25.92± 5.935	31.25±5.50	.8493	.9792	.8316	.8294	1.003
	1500	39.09± 7.275	46.88±6.74	.8493	.9845	.8361	.8340	1.003
	500	13.17±4.218	15.62±3.89	.8670	.9730	.8436	.8427	1.001
free	1000	26.74±5.995	31.25±5.50	.8670	.9850	.8540	.8557	.9981
	1500	40.27±7.302	46.88±6.74	.8670	.9889	.8663	.8591	1.008

Action set size estimate-based deletion

m.type	N	$\overline{x} \pm s$	$\mu \pm \sigma$	mut.maint.	sel.maint.	theor.result	empir.result	t.r./e.r.
	500	12.77± 3.660	15.62±3.89	.8493	.9642	.8189	.8176	1.002
niche	1000	26.00± 4.978	31.25±5.50	.8493	.9806	.8329	.8319	1.001
	1500	39.16± 5.982	46.88±6.74	.8493	.9855	.8370	.8355	1.002
	500	13.23± 3.664	15.62±3.89	.8670	.9742	.8447	.8470	.997
free	1000	26.73± 4.984	31.25±5.50	.8670	.9852	.8542	.8554	.999
	1500	40.28± 5.949	46.88±6.74	.8670	.9893	.8666	.8594	1.008

Fitness-based deletion

m.type	N	$\overline{x} \pm s$	$\mu \pm \sigma$	mut.maint.	sel.maint.	theor.result	empir.result	t.r./e.r.
	500	13.48± 4.303	15.62±3.89	.8493	.9585	.8140	.8624	.9439
niche	1000	28.46± 6.240	31.25±5.50	.8493	.9741	.8273	.9106	.9085
	1500	43.71± 7.696	46.88±6.74	.8493	.9790	.8315	.9324	.8917
	500	13.67± 4.299	15.62±3.89	.8670	.9728	.8435	.8747	.9642
free	1000	28.52± 6.160	31.25±5.50	.8670	.9828	.8521	.9127	.9336
	1500	43.70± 7.653	46.88±6.74	.8670	.9867	.8644	.9323	.9271

Action set size estimate times fitness-based deletion

m.type	N	$\overline{x} \pm s$	$\mu \pm \sigma$	mut.maint.	sel.maint.	theor.result	empir.result	t.r./e.r.
	500	13.52± 3.709	15.62±3.89	.8493	.9612	.8164	.8654	.9433
niche	1000	28.50± 5.009	31.25±5.50	.8493	.9755	.8285	.9120	.9084
	1500	43.73± 6.079	46.88±6.74	.8493	.9799	.8322	.9330	.8920
	500	13.70± 3.664	15.62±3.89	.8670	.9734	.8444	.8765	.9634
free	1000	28.52± 5.000	31.25±5.50	.8670	.9835	.8527	.9125	.9344
	1500	43.71± 5.967	46.88±6.74	.8670	.9873	.8649	.9324	.9276

Table 4: The derived proportions of mismatch between mean observed niche support and theoretically derived niche support validate the disruptive effect of mutation with respect to niche support distribution. Columns $\overline{x}\pm s$ and $\mu\pm\sigma$ denote the actual and theoretical (without considering mutation) mean and standard deviation in the niche size distribution. Entry *mut.maint.* denotes the probability that mutation is not disruptive; entry *sel.maint.* denotes the probability (derived from the final population using Equation 11) that a selected classifier is a representative; entry *theor.result* multiplies the two probabilities deriving the expected fraction of the mean of representatives for each niche present in the final population; entry *empir.result* lists the actual fraction of representatives (dividing $\overline{x}$ by μ); entry *t.r./e.r.* denotes the fraction between the expected fraction and the actually observed fraction of representatives for each niche, effectively denoting the accuracy of our model when we consider the additional disruptive effects of mutation.

classifiers. The consequent population is more focused on the support of maximally accurate, maximally general classifiers yielding a niche support that is closer to the mathematically derived support without the effects of disruptive mutation. Thus, the consideration of the disruptive effects of mutation yields a lower bound on niche support, which can be improved upon by the means of fitness-based deletion.

6.4 The Influence of the GA Threshold θ_{GA}

So far, we assumed that the GA is applied on every step, which was easily accomplished by setting $\theta_{GA} = 0$. To analyze the distribution of niche size in XCS when this hypothesis does not hold, we apply XCS to the 11-multiplexer when $\theta_{GA} = 200$ (all other parameters are the same as in the previous experiments) and we compare the empirical distributions with the distributions predicted by the model. Values of θ_{GA} closer to zero, such as $\theta_{GA} = 25$, did hardly show any significant distribution differences in comparison with the runs when $\theta_{GA} = 0$ and are consequently not shown here, but are reported elsewhere [14].

Figure 4a reports the theoretical (dashed lines) and empirical (solid lines) distributions for $N = 500$, $N = 1000$, and $N = 1500$ and $\theta_{GA} = 200$. Figure 4b shows the same results in cumulative distribution form.

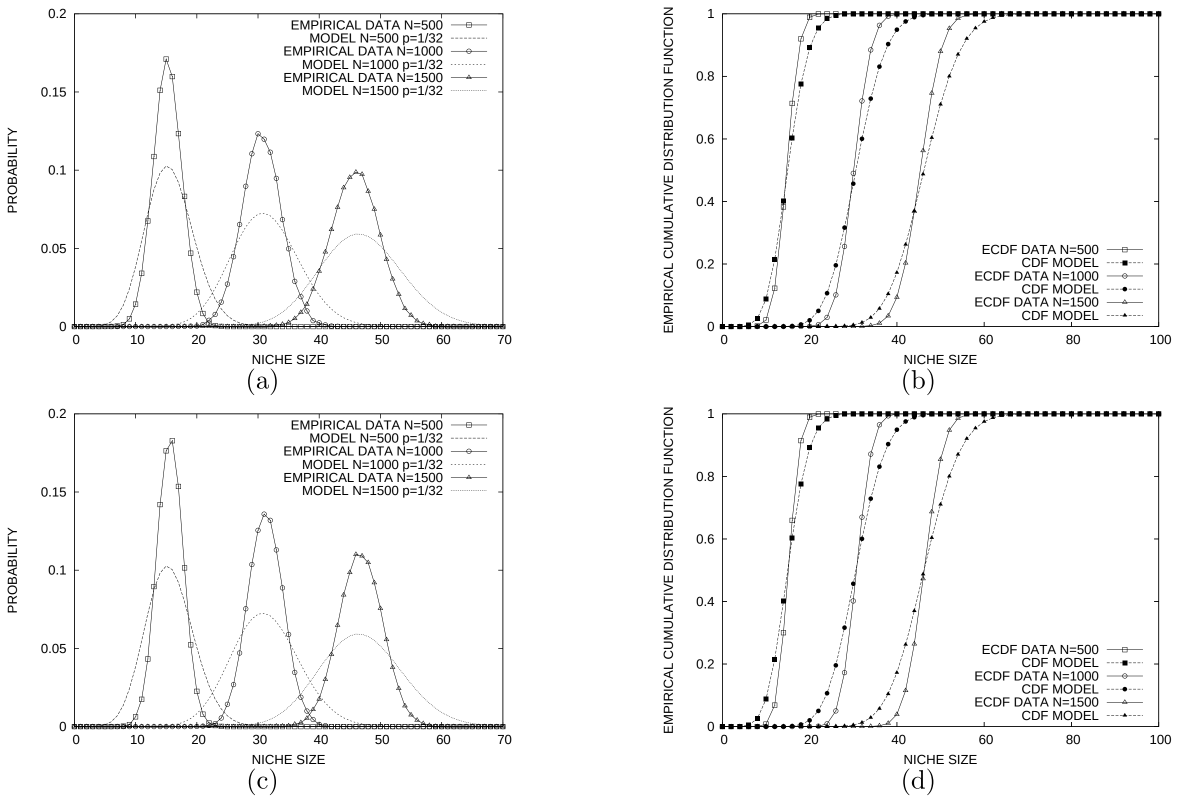


Figure 4: XCS in the 11-multiplexer when $\theta_{GA} = 200$. Distributions and corresponding cumulative distributions of niche sizes for XCS without condensation steps (a,b) and with condensation steps (c,d) (solid lines) and for the model provided by Equation 10 (dashed lines) for population sizes 500, 1000, 1500 reveal the strong focusing effect of the restricted GA application.

Figure 4c,d reports the same experiments after crossover and mutation have been turned off for additional 50,000 steps. Table 5 lists the statistics collected for all six settings.

The results show that for high values of θ_{GA}, the empirical distributions have a lower standard deviation than the deviations predicted by the model, both when crossover and mutation are turned on (4a,b), and when crossover and mutation are turned off for the last 50,000 problems (4c,d). The mean niche size for XCS ($\overline{x}$) and for the model (μ) are quite similar; when mutation and crossover are turned off, the difference is not statistically significant anymore at least for smaller population sizes (row 4c,d in Table 5). However, the distribution around the mean is much more focused on the mean than predicted by the model.

When θ_{GA} is large, classifier parameters are updated more often between reproductive events, so that the accuracy estimate provided by the classifier fitness is more precise. More importantly, over-reproduction in one niche is alleviated with a higher θ_{GA} value. Given that a reproductive event occurred in a niche, another niche reproduction event will not occur before θ_{GA} iterations have passed. Thus, if sampling generates several samples of the same niche by chance in close succession, θ_{GA} prevents additional reproductive events preventing over-reproduction in the niche and consequently avoiding additional niche support decrease of currently infrequently occurring niches. The reproduction probability is not uniform over niche occurrence any longer since more frequently occurring niches have a smaller probability of reproduction given niche occurrence. Thus, a higher GA threshold alleviates potential skews in niche occurrence frequencies.

The statistical analysis confirms the observations. Table 5 shows that when crossover and mutation are turned off the sample and model mean are not significantly different according to the *t-test* (`tt`), when $N = 500$ and $N = 1000$; while when $N = 1500$ the sample and model mean are significantly different

fig	N	$\overline{x} \pm s$	$x_{0.5}$	$\mu \pm \sigma$	bdt	tt	wt	gof	K^+	K	K^-
4a,b	500	15.20 ± 2.34	15.00	15.62 ± 3.89	1.000	0.000	0.000	0.00	0.000	0.000	0.000
	1000	30.61 ± 3.26	31.00	31.25 ± 5.50	1.000	0.000	0.000	0.00	0.000	0.000	0.000
	1500	45.83 ± 4.03	46.00	46.88 ± 6.74	1.000	0.000	0.000	0.00	0.000	0.000	0.000
4c,d	500	15.62 ± 2.14	16.00	15.62 ± 3.89	1.000	0.923	0.000	0.00	0.000	0.000	0.000
	1000	31.24 ± 2.91	31.00	31.25 ± 5.50	1.000	0.380	0.231	0.00	0.000	0.000	0.000
	1500	46.75 ± 3.57	47.00	46.88 ± 6.74	1.000	0.000	0.808	0.00	0.000	0.000	0.000

Table 5: XCS in the 11-multiplexer when $\theta_{GA} = 200$. Statistics for the distribution of niche size depicted in Figure 4. Entry fig indicates the figure to which the statistics refers to and N is the population size. The other entries are explained in more detail in Table 1.

according to the *t-test* but not according to the non-parametric Wilcoxon's test (wt). In all the experiments reported the empirical data are compatible with a Binomial distribution (column bdt), although they do not fit the model (column gof). In addition, in all cases the empirical and model distributions are significantly different according to the Kolmogorov-Smirnov test (columns K^-, K, and K^+).

In sum, the higher the GA threshold, the more uniform niche reproduction becomes and the more focused the classifier niche support distribution becomes. This may prevent XCS from loosing infrequently occurring niches but also has a negative effect on learning speed since the GA application frequency is decreased overall. Thus, a larger GA threshold may yield larger classification accuracies with smaller population sizes. However, to reach this accuracy more learning iterations will likely be required. Seeing these constraints, an approach that adapts the GA threshold over a learning run may be advantageous.

6.5 Discussion

This section confirmed that the proposed model is able to estimate resulting niche distributions. Despite several additional assumptions, the data fits the model nearly perfectly when condensation steps are applied, in which mutation and crossover are switched off. The disagreements in the runs without condensation steps were explainable by the disruptive effects of mutation. Once the solution is found, mutation is mainly disruptive. The addition of mutative disruption to the derived theory (by decreasing the probability of reproduction p by a mutation-dependent factor) was able to account for the influence of mutation on niche support, as experimentally validated. Crossover showed to have only a minor influence on niche support. Moreover, we observed benefits of applying fitness-based deletion mechanisms proposed elsewhere [37]. The mechanism helps in the detection of over-general classifiers for deletion consequently focusing the population on the accurate, maximally general ones.

We also analyzed the influence of the GA application threshold θ_{GA}. The model is correct for small θ_{GA} values. For larger values, the parameter causes additional niching pressure that does not change the mean but decreases the variance of the niche support distribution. Essentially, the threshold prevents the over-reproduction in currently frequently occurring niches, ensuring a more balanced niche reproduction and a consequently more balanced niche support distribution.

The evaluations in this section focused on the problem case in which solutions can be represented by non-overlapping subsolutions. In the next section we analyze how the model fits when this assumption does not hold, that is, when subsolutions overlap. In this case, a continuous interaction among the overlapping classifiers can be expected since the classifiers have to compete for fitness resources and reproductive opportunities.

7 Overlapping Subsolutions

A main assumption in our model is that the required subsolutions in the optimal problem solution $[O]$ do not overlap. In general, though, subsolutions may overlap and classifiers that are representatives of different but overlapping niches may have to share their reproductive opportunities with each other. In this section, we

1	`0**0**:0`	12	`1**1**:1`	19	`0**0**:1`	30	`1**1**:0`
2	`***000:0`	13	`*1*11*:1`	20	`***000:1`	31	`*1*11*:0`
3	`**000*:0`	14	`11**1*:1`	21	`**000*:1`	32	`11**1*:0`
4	`*0*0*0:0`	15	`**1111:1`	22	`*0*0*0:1`	33	`**1111:0`
5	`*0*00*:0`	16	`*111*1:1`	23	`*0*00*:1`	34	`*111*1:0`
6	`*000**:0`	17	`1*1*11:1`	24	`*000**:1`	35	`1*1*11:0`
7	`0***00:0`	18	`111**1:1`	25	`0***00:1`	36	`111**1:0`
8	`0*0*0*:0`			26	`0*0*0*:1`		
9	`00***0:0`			27	`00***0:1`		
10	`00**0*:0`			28	`00**0*:1`		
11	`000***:0`			29	`000***:1`		

Table 6: Niches for the $\texttt{car}_3$; each niche is represented by a schema and an action.

apply XCS to a problem that causes the evolution of overlapping classifiers and we compare the experimental results to the proposed model.

7.1 Boolean Carry

The Boolean carry of size k, briefly $\texttt{car}_k$, takes as argument a string of size $2k$, representing two binary numbers of size k, x and y, and returns the carry obtained by the sum $x + y$. For example, suppose that $k = 3$, $x =$`100`, and $y =$`101`, then $\texttt{car}_3$(`100101`) returns 1; if $x =$`010` and $y =$`001` then $\texttt{car}_3$(`010001`) returns 0. More formally, when $k = 3$, given $x = x_0x_1x_2$ and $y = y_0y_1y_2$, the function $\texttt{car}_3(x_0x_1x_2y_0y_1y_2)$ is defined as:

$$\texttt{car}_3 = x_2y_0y_1y_2 + x_0x_2y_1y_2 + x_1x_2y_0y_2 + x_0x_1x_2y_2 + x_1y_0y_1 + x_0x_1y_1 + x_0y_0$$

Table 6 reports the 36 classifiers that represent the complete optimal solution for the $\texttt{car}_3$. The solution is clearly overlapping. For example, the schemata `000***:0` and `00**0*:0` overlap, the schemata `1**1**:1` and `111**1:1` overlap, etc.

Solution overlap has several effects. First, the sharing of reproductive opportunities decreases the probability of reproduction in each niche dependent on its overlap with other niches. Second, the classifiers' action-set size estimates *as* overestimate niche sizes and thus may lead to an undesired bias towards deleting niches with stronger overlap. Most importantly, highly overlapping classifiers undergo genetic drift and consequently are lost more easily. To investigate these effects with respect to the proposed model, we apply XCS to the Boolean carry of size three, $\texttt{car}_3$, and compare the empirical data with the model.

7.2 Reproduction Probability for Overlapping Niches

To compare the experimental data to the model in the case of overlapping niches we must derive the reproduction probability p. Due to the overlap, this is difficult since p directly depends both (i) on the number of overlapping niches and (ii) on the proportion of representatives of the niche under investigation with respect to the other niches. Nonetheless, it is possible to derive (i) an *overall* reproduction probability p_0 for all representatives of the "*zero niche*" comprising all schemata in Table 6 whose specified bits in the condition part are zero, and (ii) the reproduction probability p_1 for all representatives of the "*one niche*" comprising all schemata in Table 6 whose specified bits in the condition part are one. It is easy to compute that the *zero niche* covers the $9/16^{th}$ of the overall search space, that is, $p_0 = 0.56$ the *one niche* covers the remaining $7/16^{th}$ of the overall search space, that is, $p_1 = 0.44$

fig	N	$\overline{x} \pm s$	$x_{0.5}$	$\mu \pm \sigma$	bdt	tt	wt	gof	K^+	K	K^-
5a	400	183.43 ± 14.52	184.00	225.00 ± 9.92	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	800	397.23 ± 19.61	397.00	450.00 ± 14.03	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	1200	619.78 ± 25.90	620.00	675.00 ± 17.18	0.000	0.000	0.000	0.00	1.000	0.000	0.000
5b	400	167.49 ± 20.52	168.00	225.00 ± 9.92	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	800	425.42 ± 21.09	426.00	450.00 ± 14.03	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	1200	660.77 ± 27.21	661.00	675.00 ± 17.18	0.000	0.000	0.000	0.00	0.158	0.000	0.000
5c	400	154.69 ± 12.61	155.00	175.00 ± 9.92	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	800	351.02 ± 18.63	351.00	350.00 ± 14.03	0.000	0.000	0.000	0.00	0.000	0.000	0.000
	1200	529.44 ± 24.89	529.00	525.00 ± 17.18	0.000	0.000	0.000	0.00	0.000	0.000	0.000
5d	400	140.01 ± 14.42	140.00	175.00 ± 9.92	0.000	0.000	0.000	0.00	1.000	0.000	0.000
	800	357.62 ± 22.47	359.00	350.00 ± 14.03	0.000	0.000	0.000	0.00	0.000	0.000	0.000
	1200	537.80 ± 27.60	538.00	525.00 ± 17.18	0.000	0.000	0.000	0.00	0.000	0.000	0.071

Table 7: XCS in the $\texttt{car}_3$ problem when $\theta_{GA} = 0$, $N = 400$, $N = 800$, and $N = 1200$. Statistics for the distribution of niche size depicted in Figure 5. `fig` indicates the figure to which the statistics refer to; N is the population size; refer to Table 1 for the meaning of the other columns.

7.3 Experiments

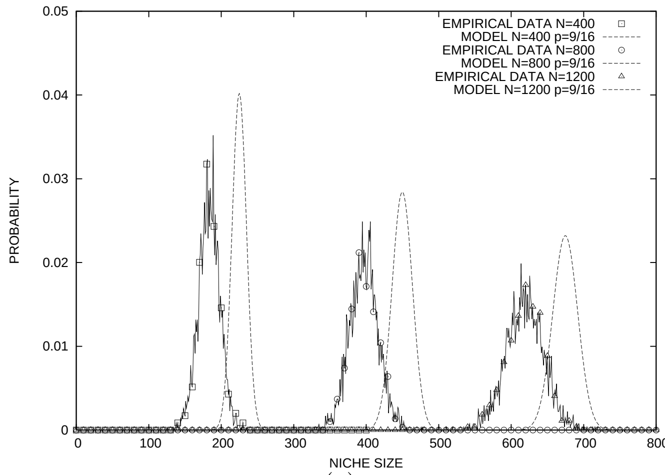
We apply XCS to the $\texttt{car}_3$ problem with different population sizes ($N = 400$, $N = 800$, and $N = 1200$). The GA threshold is set to $\theta_{GA} = 0$. All other parameters were set as in the previous section. For each value of N, we ran 5000 trials, each consisting of 100,000 problem instances. From the final population, we compute (i) the size of the *zero niche*, by summing up the sizes of all the 22 niches whose schema is described by zeros, (ii) the size of the *one niche*, by summing up the sizes of all the 14 niches whose schema is described by ones.

Figure 5a reports (i) with dashed lines, the three theoretical distributions provided by our Markov model (Equation 10) when $p = 9/16$, and $N = 400$, $N = 800$, $N = 1200$; (ii) with solid lines, the size of the *zero niche* estimated through the experiments when $N = 400$, $N = 800$, and $N = 1200$. Figure 5b reports the same results after additional 50,000 condensation steps. Likewise, Figure 5c and Figure 5d compare the theoretical distributions (dashed lines) to the experimental data (solid lines) for the *one niche* and $p = 7/16$. In Table 7, we report the applied statistics collected from the experiments (see Table 1 for reference).

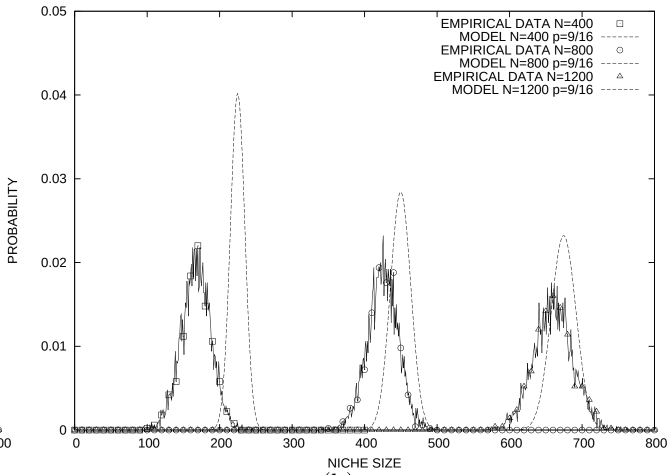
Overall, the reported results show that the actual distribution of zero and one niches is in fact distributed around the expected means. However, if condensation is not applied, both distributions are shifted to the left indicating that the support is smaller than expected. This confirms the disruptive effect of mutation analyzed for the 11-multiplexer in the previous section. When condensation is switched on, the curves do approach their expected mean value. For small population sizes ($N = 400$ and $N = 800$), some niches are lost which explains the (e.g., niche or free; see [16])shift to a smaller support. This niche loss is more visible by analyzing the distribution of each niche separately. For the case of $N = 800$, Figure 6 reports the niche size distributions of each of the 36 niches separately. Note that the smallest and most overlapping niches (`**1111`, `1*1*11`, `*111*1`, and `111**1`) tend to completely lose niche support sometimes (Figure 6c). Since mutation and crossover are switched off during condensation, a recovery is not possible in the event of a niche loss shifting the distribution even further to the left as shown in Figure 6d.

In the case of a larger population size ($N = 1200$), the distributions are slightly shifted to the left for the zero niches and to the right for the one niches. This observation confirms the additional balancing effect due to the action-set size estimate based deletion. Shifting both distributions to a balanced niche support level.

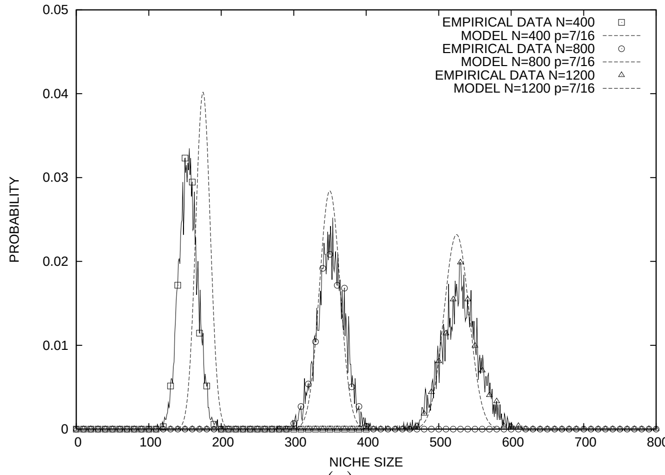
The observations confirm our hypothesis that genetic drift may cause niche loss of highly overlapping, infrequently occurring niches. Although XCS does apply fitness sharing techniques in that fitness reflects the average relative (shared) accuracy of a classifier, due to the multiple overlaps, the sharing technique does not seem to be able to prevent the potential loss of a niche. Further theoretical results for competitive, overlapping niches can be found in [31, 32]. These results shows that overlap becomes relevant only in strongly overlapping problems. Additionally, advanced niching techniques that can detect overlapping niches and assure their sustenance might be desirable. Section 11 discusses such options in further detail.



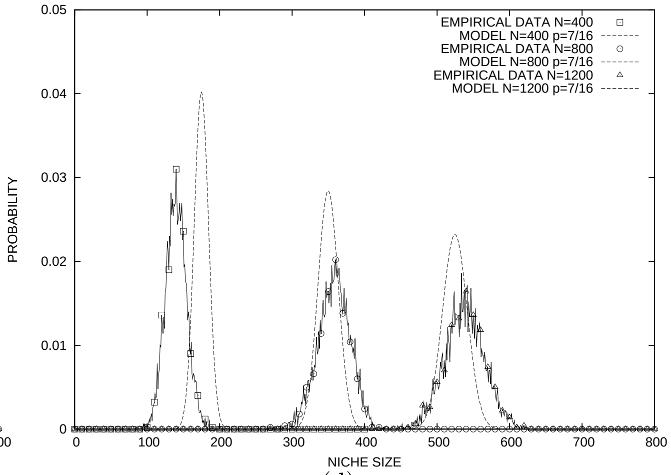
(a)



(b)



(c)



(d)

Figure 5: Distribution of niche sizes for XCS in the $\texttt{car}_3$ (solid line) and for the model provided by Equation 10 (dashed line) when $\theta_{GA} = 0$, deletion is based on the action set size estimate, and the population sizes are 400, 800, and 1200 classifiers. Plots (a) and (b) report the distribution of niches corresponding to the zero niche, without condensation, plot (a), and with condensation, plot (b). Plots (c) and (d) report the distribution of niches corresponding to the one niche, without condensation, plot (c), and with condensation, plot (d). Statistics for XCS were collected over 5000 experiments.

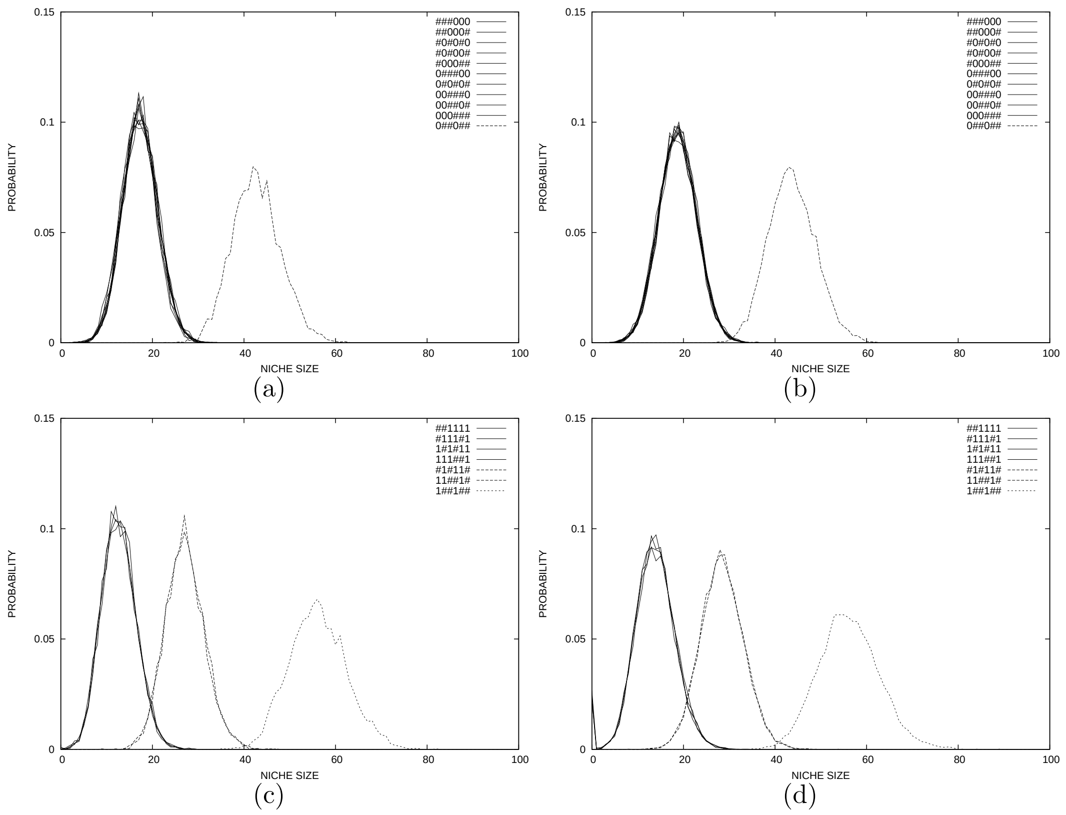


Figure 6: XCS in the $\mathtt{car}_3$ with the usual action-set plus fitness-based deletion, when $N = 800$ and $\theta_{GA} = 0$. Plots (a) and (b) report the distribution of niches belonging to the "*zero niche*", without condensation, plot (a), and after condensation, plot (b). Plots (c) and (d) report the distribution of niches belonging to the "*one niche*", without condensation, plot (a), and after condensation, plot (b).

8 Learning Complexity

The results we reported so far show that our model for niche support in XCS can predict the distribution of niche sizes for problems involving non-overlapping niches. Effectively, our model provides an asymptotic prediction of niche support distribution. The model was able to account for mutation affects as well, where the expected niche support distribution can be derived by the probability of niche occurrence times representative reproduction times mutation maintenance.

Besides the capability of estimating the expected niche support distribution, the model can be used to derive a population size bound to ensure that a complete model is sustained with high probability. In particular, from the probability u_0 of having a niche without any representative classifier we now derive a population-sizing bound that a niche is not lost with an arbitrarily high probability

In essence, our model can approximate the probability that a niche is lost. Using Equation 9, we can derive a bound for the population size N that ensures with high probability that XCS does not forget about any problem niche, that is, loose a relevant subsolution. In fact, in Appendix B we derived that the probability of being in state u_0 (which means, that the respective niche was lost) is $u_0 = (1-p)^N$. Thus, the probability of loosing a niche decreases exponentially with the population size. Given the problem has 2^o problem niches, that is, the perfect solution $[O]$ is represented by 2^o schemata of order o, the probability of loosing a niche equates to $u_0 = \left(1 - \frac{1}{2^o}\right)^N$.

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

$$\frac{\log(1-\theta)}{\log(1-\frac{1}{2^o})} \approx \frac{\log(1-\theta)}{\log(1-p)} \approx -\frac{\log(1-\theta)}{p} < N, \tag{12}$$

showing that population size N grows logarithmically in the confidence value and linearly in the inverse of the occurrence probability p, which can be approximated by 2^o given uniform problem sampling and equally sized niches. 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 polynomially grows in problem complexity and logarithmically in the confidence value.

Alternatively, the result shows that given a certain population size, XCS will maintain a solution that consists of subsolutions with a minimal occurrence probability p as long as the subsolutions are not severely overlapping. In other words, the bound suggests that, given a certain population size N, XCS is able to sustain a complete problem solution with high probability as long as the required problem subsolutions do not fall below an occurrence probability p.

Thus, although we usually do not know how complex the subsolutions to a problem need to be, the bound indicates that we are able to sustain *all* subsolutions that have a minimal occurrence probability p. Given a uniformly sampled, binary problem space, and given any *k-DNF* problem in this space, the occurrence of each subsolution can be bounded by $1/2^k$. Furthermore, given that there is no severe overlap between the problem subsolutions, that is, the conjunctions with k literals, the bound shows that XCS can sustain a problem solution with high probability with a population size that grows logarithmically in the confidence value and linearly in k. This observation confirms that with respect to solution sustenance, XCS is a PAC learning system with respect to (only partially overlapping) $k - DNF$ problems. Since the search for a solution was also shown to be polynomial in confidence, solution complexity, and problem size [16, 15, 13], the result confirms Wilson's original XCS learning hypothesis [50].

It might be argued that the disruption due to mutation is not considered in the complexity analysis and not derivable in the general case. Disruption by mutation may be bounded by the probability that no mutation occurs in the relevant attributes. Since in $k - DNF$ maximally k attributes are relevant, the probability of not disrupting offspring via mutation can be bounded by $(1-\mu)^k$ and the additional factor becomes irrelevant for the complexity bound derived in Equation 12. The additional factor of the probability of selecting a representative given the occurrence of a certain niche is very hard to bound since it depends on the selection method used as well as on the fitness scaling used. Clearly, stronger fitness pressures caused by higher fitness scaling rates, such as a smaller scaling value α and a larger ν (see Equation 3 in Section 2), or by a more focused selection method, such as set-proportionate tournament selection [17], can decrease this additional disruptive influence. In the case of set-proportionate tournament selection, the probability of selecting a representative can be lower bounded by the set-proportion assuming that a representative is guaranteed to have the highest fitness value in an action set [12]. The set-proportion is usually set to $\tau = .4$. Consequently, a constant τ needs to be multiplied to the p value in Equation 12, which again does not influence the overall learning complexity.

9 Related Work

Several previous articles have approached the problem of modeling the behavior of evolutionary systems with Markov chain analyses including [4, 25, 24, 28, 32, 5, 6]. We now present a brief overview over other approaches in the literature most related to the path taken in this paper.

Harik et. al. [28] used a simple Markov chain model to derive the probability of deciding well among competing building blocks. The Markov chain assumed a constant probability of reproduction and deletion (derived from fitness). Using the result from the random walk literature of how likely it is that a niche is absorbed (reaching the limit of the Markov chain), a population size model was derived that assured the probability of a successful evolutionary process with high probability. Albeit similar in the approach, our Markov chain faces changing deletion probabilities resulting in the observed niche support stabilization.

Horn et. al. [32] developed a Markov chain model for a learning classifier system in which selection and deletion were applied to the whole population. The work emphasizes the importance of fitness sharing for the maintenance of partially overlapping classifiers. Also XCS applies fitness sharing to enforce the sustenance of partially overlapping classifiers. However, the main niching techniques lies in the niche reproduction in combination with global deletion, as seen in the analysis. Thus, the analysis of Horn et. al. matches closer to the ZCS classifier system [48], in which reproduction and deletion are applied globally to the whole population as well.

Bull developed a Markov chain model for single step problems [5] and multistep problems [6] (see also, [8]). The developed models were restricted to simple one-bit problems and two-step multistep problems. Moreover, the models investigated the learning progress rather than the convergence and solution sustenance aspects. In contrast, our niche size distribution model considers solution sustenance and applies to all problems in which the final solution can be represented with potentially partially overlapping classifiers and in which each necessary subsolution has an occurrence probability of at least p. Nonetheless, currently our model is restricted to the single-step, classification domain. In multistep problems, solution sustenance can be expected to be harder due to the expectable skewed problem space sampling. Advanced exploration techniques as well as niche support techniques will be necessary to balance sampling and prevent forgetting of infrequently visited problem subspaces.

10 Summary

In this paper we have investigated the niching capabilities of XCS. We derived a Markov chain model for niche support in XCS and showed that the support of a niche in XCS can be approximated by a Binomial distribution over the niche size. In the case of non-overlapping niches, the model agreed to the experimental evaluations nearly perfectly, even when the disruptive effect of mutation was taken into account.

The experimental evaluations confirmed the focusing effects of action-set size based deletion method and the thresholded GA application. The action-set size based deletion method adds additional pressure towards eliminating useless overlapping and over-general classifiers. The thresholded GA application, controlled by parameter θ_{GA}, decreases variance in the distributions and favors less frequently occurring niches. Both mechanisms are consequently beneficial in maintaining a complete problem solution. However, a large θ_{GA} threshold may actually decrease the initial learning speed. Thus, an adaptive threshold θ_{GA} that is set to a low value early in a run and increases later in the run may be beneficial to speed-up initial learning and to sustain the final problem solution later.

The experimental evaluations also showed that mutation causes disruption in the sustenance of the final population. The estimated disruption caused in the 11-multiplexer was confirmed by experimental evaluations. Clearly, mutation is necessary to evolve a complete problem solution so that, as in the case of the parameter θ_{GA}, an adaptive mutation parameter approach may be advantageous. Bull et. al. have worked on a self-adaptive mutation scheme [7, 9, 33]. However, with our knowledge about an initially necessarily higher mutation rate and a later, smaller mutation rate, it might be advantageous to experiment with a heuristically adaptive mutation parameter that decreases once the solution was found. On the other hand, the complexity derivation showed a negligible influence of mutation as long as the mutation rate is kept small.

In overlapping niches, dependent on the degree of overlap, genetic drift may influence niche sustenance. For this case, the model may need to be enhanced to account for the resulting genetic drift. However, as analyzed elsewhere [32], slight overlaps do not have a significant influence on the required final population size. Nonetheless, the experimental results confirmed that the population sizes need to be set larger when overlap occurs. Thus, future research should investigate the potential incorporation of other niching techniques, such as crowding techniques [20, 44, 27], in which the replacement of classifiers is restricted.

Besides our capability of estimating niche size distributions, we were also able to derive a population size bound that ensures with high probability θ that the complete solution will be maintained, as long as all niches in the final solution have an occurrence probability of more than p and the niches are not severely overlapping. We showed that to satisfy this bound, population size needs to grow logarithmically in $1 - \theta$ and linearly in the inverse of the smallest niche occurrence probability in the problem. If all subsolutions occur approximately equally frequently, the population size needs to grow linearly in the number of required subsolutions (that is, classifiers).

From a more global perspective, we saw that niching in XCS is achieved by a reproduction mechanism that is biased on the niche occurrence frequency in combination with a deletion mechanism that is biased on niche size. Thus, XCS's niching mechanism strongly depends on problem sampling. We assumed a uniform distribution over niche occurrences or an independent probability p of the occurrence of each niche. XCS's mechanism may become less stable, once niche occurrences become skewed or history dependent. In essence, the larger the delay between niche occurrences, the larger the probability that a niche may be lost. Higher population sizes can alleviate this effect with a necessary growth linear in $1/p$ where p might be set to the lowest occurrence probability possible dependent on the history or where $1/p$ may be set directly to the maximum expected delay between two occurrences of the same niche.

11 Conclusions

The related work section showed that a niche support analysis is not available for any other classifier system although the analysis is highly important to ensure complete solution sustenance. Moreover, the analysis is important to show the dependency of the classifier system on problem subsolution sampling. Any classifier system that evolves its population online is destined to lose subsolutions if no proper niching technique is applied. XCS accomplishes proper niching using an occurrence-based selection mechanism in combination with an approximately random (slightly niche occupation-biased) deletion mechanism. As discussed above, the deletion mechanism may be further biased to further improve niching in XCS. Such biasing is expected to be particularly useful in problems in which highly overlapping subsolutions are expected.

Other classifier systems should undergo similar niching analyses. By deriving a reproduction probability estimate as well as a deletion probability estimate, any niche support behavior can be approximated by a Markov chain in which states denote the current niche size. Clearly, the probabilities depend on the particular LCS system at hand. However, if neither the probability of deletion decreases with decreasing niche occupancy nor the probability of reproduction increases with decreasing niche occupancy, then the system will be destined to lose important subsolutions due to genetic drift. In the ZCS system [48, 8], for example, reproduction depends on fitness sharing so that the analysis of Horn et. al. [32] applies. However, the fitness values need to reflect the dynamics in the population as accurately as possible, which may explain the necessary high learning rates observed elsewhere [8]. Niching challenges similar to the ones observed for

XCS are expected in problems in which overlapping subsolutions are required in ZCS.

The modular analysis in this paper revealed that XCS has a very flexible niching mechanism that depends mainly on niche occurrence frequency. Research is on the way to investigate and enhance this analysis to other problem domains and other condition representations. Hereby, it becomes increasingly obvious that the most suitable condition structure as well as the most suitable prediction structure will enable XCS to find the most accurate and most general problem solution [11, 40]. XCS adapts the provided structures in such a way that the conditions partition the problem space to ensure maximally accurate predictions. The derived niche support bound in this paper shows how these partitions are sustained in XCS and it quantifies the population size required to ensure the sustenance of non- or partially overlapping niches with high probability.

Acknowledgments

We are grateful to Xavier Llorà, Kei Onishi, Martin Pelikan, Stewart Wilson, Tian-Li Yu, and the whole IlliGAL lab for their help and the useful discussions.

This work was supported by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, the National Science Foundation under ITR grant DMR-03-25939 (at Materials Computation Center, UIUC), 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 European commission contract no. FP6-511931.

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 any of the organizations mentioned above.

References

[1] E. Bernadó, X. Llorà, and J. M. Garrell. XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In Lanzi et al. [42], pages 115–132.

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

[3] H.-G. Beyer, U.-M. O'Reily, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cant-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 2*, New York, NY, 2005. Association for Computing Machinery, Inc.

[4] C. L. Bridges and D. E. Goldberg. An analysis of reproduction and crossover in a binary-coded genetic algorithm. In Grefenstette [26], pages 9–13.

[5] L. Bull. Simple Markov Models of the Genetic Algorithm in Classifier Systems: Accuracy-Based Fitness. In Lanzi et al. [41], pages 21–28.

[6] L. Bull. Simple Markov Models of the Genetic Algorithm in Classifier Systems: Multi-step Tasks. In Lanzi et al. [41], pages 29–36.

[7] L. Bull and J. Hurst. Self-Adaptive Mutation in ZCS Controllers. In S. Cagnoni, R. Poli, G. Smith, D. Corne, M. Oates, E. Hart, P.-L. Lanzi, E. Willem, Y. Li, B. Paechter, and T. C. Fogarty, editors, *Proceedings of the EvoNet Workshops - EvoRob 2000*, pages 339–346, Berlin Heidelberg, 2000. Springer-Verlag.

[8] L. Bull and J. Hurst. ZCS Redux. *Evolutionary Computation*, 10(2):185–205, 2003.

[9] L. Bull, J. Hurst, and A. Tomlinson. Self-adaptive mutation in classifier system controllers. In J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S. W. Wilson, editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 460–467, Cambridge, MA, 2000. MIT Press.

[10] M. V. Butz. Documentation of XCS+TS C-Code 1.2. Technical Report 2003023, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2003.

[11] M. V. Butz. Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system. In Beyer et al. [3], pages 1835–1842.

[12] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*. Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg, 2006.

[13] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Bounding learning time in XCS. In R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, P. L. Lanzi, L. Spector, A. G. B. Tettamanzi, D. Thierens, and A. M. Tyrrell, editors, *Genetic and Evolutionary Computation Conference - GECCO 2004: Part II*, pages 739–750, Berlin Heidelberg, 2004. Springer-Verlag.

[14] M. V. Butz, D. E. Goldberg, P. L. Lanzi, and K. Sastry. Bounding the population size to ensure niche support in XCS. Technical Report 2004033, Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign, 2004.

[15] 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.

[16] 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(1):28–46, 2004.

[17] M. V. Butz, K. Sastry, and D. E. Goldberg. Strong, stable, and reliable fitness pressure in XCS due to tournament selection. *Genetic Programming and Evolvable Machines*, 6:53–77, 2004.

[18] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. *Journal of Soft Computing*, 6(3–4):144–153, 2002.

[19] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, 1998. third edition.

[20] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, 1975. University Microfilms No. 76-9381.

[21] P. W. Dixon, D. W. Corne, and M. J. Oates. A preliminary investigation of modified XCS as a generic data mining tool. In Lanzi et al. [42], pages 133–150.

[22] R. A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh, GB, 1925. Available online at `http://psychclassics.yorku.ca/Fisher/Methods/`.

[23] M. Friendly. *Visualizing Categorical Data*. SAS Institute, Cary, NC, 2000.

[24] D. Goldberg and P. Segrest. Finite Markov chain analysis of genetic algorithms. In Grefenstette [26], pages 1–8.

[25] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In L. Davis, editor, *Genetic algorithms and simulated annealing*, Research Notes in Artificial Intelligence, pages 74–88. Pitman, London, 1987.

[26] J. J. Grefenstette, editor. *Proceedings of the Second International Conference on Genetic Algorithms*, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[27] G. Harik. Finding multiple solutions in problems of bounded difficulty. In L. Eschelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.

[28] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7:231–253, 1999.

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

[30] J. H. Holland. *Hidden Order: How Adaptation Builds Complexity.* Perseus Books, Cambridge, MA, USA, 1995.

[31] J. Horn. Finite Markov chain analysis of genetic algorithms with niching. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 110–117, San Francisco, CA, 1993. Morgan Kaufmann.

[32] J. Horn, D. E. Goldberg, and K. Deb. Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1):37–66, 1994.

[33] J. Hurst and L. Bull. A Self-Adaptive XCS. In Lanzi et al. [42], pages 57–73.

[34] K. A. D. Jong and W. M. Spears. Learning Concept Classification Rules using Genetic Algorithms. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence IJCAI-91*, volume 2, San Francisco, CA, 1991. Morgan Kaufmann.

[35] L. Kleinroch. *Queueing Systems: Theory.* John Wiley & Sons, New York, NY, USA, 1975.

[36] T. Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68, New York, NY, USA, 1997. Springer-Verlag.

[37] T. Kovacs. Deletion schemes for classifier systems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 329–336, San Francisco, CA, 1999. Morgan Kaufmann.

[38] T. Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems.* Springer-Verlag, Berlin Heidelberg, 2003.

[39] P. L. Lanzi. The XCS library. `http://xcslib.sourceforge.net`, 2002.

[40] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Extending XCSF beyond linear approximation. In Beyer et al. [3], pages 1827–1834.

[41] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors. *Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000: Paris, France, September 2000: Revised Papers*, volume 1996 of *LNAI*, Berlin Heidelberg, 2001. Springer-Verlag.

[42] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors. *Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001: San Francisco, CA, USA, July 2001: Revised Papers*, volume 2321 of *LNAI*, Berlin Heidelberg, 2002. Springer-Verlag.

[43] D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications.* John Wiley & Sons, New York, NY, USA, May 1979.

[44] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 27–36, Amsterdam, 1992. Elsevier.

[45] R Development Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-00-3.

[46] G. Venturini. Adaptation in dynamic environments through a minimal probability of exploration. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 371–381, Cambridge, MA, 1994. MIT Press.

[47] S. W. Wilson. Classifier systems and the animat problem. *Machine Learning*, 2:199–228, 1987.

[48] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.

[49] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[50] S. W. Wilson. Generalization in the XCS classifier system. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674, San Francisco, CA, 1998. Morgan Kaufmann.

A List of Symbols

XCS Parameters

N	maximal population size
$P_{\#}$	don't care probability
β	learning rate
$\alpha, \varepsilon_0, \nu$	accuracy calculation parameters
θ_{GA}	threshold that controls GA invocation
μ	probability of mutating a condition attribute (or the action)
χ	probability of applying the chosen crossover operator
θ_{del}	threshold that requires min. experience for fitness influence in deletion
δ	fraction of mean fitness below which deletion probability is increased
θ_{sub}	threshold that requires minimal experience for subsumption

Classifier Parameters

C	condition part; in bin. problems, $C \in \{0, 1, \#\}^l$
A	action part
R	reward prediction
ε	mean absolute reward prediction error
κ	current accuracy
κ'	current relative accuracy
F	fitness (in macro classifiers)
as	the mean action set size the classifier is part of
ts	last time stamp the classifier was part of a GA application set
exp	the number of evaluation steps the classifier underwent so far
num	the numerosity, that is, the number of micro-classifiers represented by this (macro-) classifier

Model-Related Parameters

k	number of representatives (state k in Markov chain)
r_k	probability of increasing niche support by one
s_k	probability of leaving niche support unchanged
d_k	probability of decreasing niche support by one
p	probability of reproducing a representative
μ	theoretical mean supply of representatives
σ	theoretical standard deviation of supply of representatives
$\bar{x}$	empirical (sampled) mean supply of representatives
s	empirical (sampled) deviation of supply of representatives
θ	probability of sustaining a niche
$aspec(cl)$	number of additionally specified attributes in a classifier compared to a schema

Other Notations

$[P]$	classifier population
$[M]$	match set
$[A]$	action set
$[O]$	optimal set of classifiers in the sense of [36]
r	feedback (reward) received from the environment
$P(a_i)$	prediction array estimating the value of action a_i

B Closed-Form Equation

In this section, we develop the closed form equation for the Markov chain in Figure 1. We start from the following equation, introduced in Section 4:

$$u_k = r_{k-1}u_{k-1} + s_k u_k + d_{k+1}u_{k+1} \tag{13}$$

by replacing the probabilities r_k, d_k, and s_k, with their actual values (see Section 4 for details):

$$\begin{aligned} r_k &= p\left(1-\frac{k}{N}\right) \\ s_k &= (1-p)\left(1-\frac{k}{N}\right)+p\frac{k}{N} \\ d_k &= (1-p)\frac{k}{N} \end{aligned}$$

we obtain,

$$\left[p\left(1-\frac{k}{N}\right)+\frac{k}{N}(1-p)\right]u_k = (1-p)\left(\frac{k+1}{N}\right)u_{k+1} + p\left(1-\frac{k-1}{N}\right)u_{k-1}$$

finally, by dividing by $(1-p)u_{k-1}$, we derive the following equation:

$$\left[\frac{p}{1-p}(N-k)+k\right]\frac{u_k}{u_{k-1}} = (k+1)\frac{u_{k+1}}{u_{k-1}} + \frac{p}{1-p}(N-k+1) \tag{14}$$

To derive a closed-form solution for probability u_k we first use Equation 14 to derive a closed-form equation for the ratio $\frac{u_k}{u_0}$. Next, we use the equation for $\frac{u_k}{u_0}$ and the condition $\sum_{k=0}^{N} u_k = 1$, to derive the closed-form solution for u_k.

B.1 Closed-form Equation for u_k/u_0

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

$$u_0 = s_0 u_0 + d_1 u_1. \tag{15}$$

By substituting the values of s_0 and d_1 we obtain:

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

from which we derive equation:

$$\frac{u_1}{u_0} = \frac{p}{1-p}N. \tag{16}$$

To derive the equation for u_2/u_0 we start from Equation 14 and set $k = 1$:

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

so that

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

We now replace u_1/u_0 with Equation 16:

$$\begin{aligned}
\frac{u_2}{u_0} &= \frac{1}{2}\left[\left(\frac{p}{1-p}(N-1)+1\right)\frac{u_1}{u_0} - \frac{p}{1-p}N\right] \\
&= \frac{1}{2}\left[\left(\frac{p}{1-p}(N-1)+1\right)\frac{p}{1-p}N - \frac{p}{1-p}N\right] \\
&= \frac{1}{2}\left[\frac{p}{1-p}(N-1)\frac{p}{1-p}N\right] \\
&= \frac{N(N-1)}{2}\left(\frac{p}{1-p}\right)^2 \\
&= \binom{N}{2}\left(\frac{p}{1-p}\right)^2 \qquad (17)
\end{aligned}$$

This leads us to the hypothesis that

$$\frac{u_k}{u_0} = \binom{N}{k}\left(\frac{p}{1-p}\right)^k, \qquad (18)$$

which we prove by induction. Using Equation 18, we can first derive that:

$$u_{k+1} = \frac{N-k}{k+1}\frac{p}{1-p}u_k \qquad (19)$$

$$u_k = \frac{N-k+1}{k}\frac{p}{1-p}u_{k-1} \qquad (20)$$

$$u_{k-1} = \frac{1-p}{p}\frac{k}{N-k+1}u_k \qquad (21)$$

With Equation 14 substituting Equation 21 as the inductive step, we now derive

$$\begin{aligned}
u_{k+1} &= \frac{\left(\left(\frac{p}{1-p}(N-k)+k\right)\frac{u_k}{u_{k-1}} - \frac{p}{1-p}(N-k+1)\right)u_{k-1}}{k+1} \qquad (22) \\
&= \frac{\left(\frac{p}{1-p}\right)^2\frac{(N-k)(N-k+1)}{k}u_{k-1}}{k+1} \qquad (23) \\
&= \frac{N-k}{k+1}\frac{p}{1-p}u_k, \qquad (24)
\end{aligned}$$

which proves the hypothesis.

B.2 Derivation of u_0

To derive a closed-form for u_k from the equation for u_k/u_0, we use the subsidiary condition:

$$\sum_{k=0}^{N} u_k = 1 \qquad (25)$$

We divide both terms by u_0

$$\sum_{k=0}^{N} \frac{u_k}{u_0} = \frac{1}{u_0}, \tag{26}$$

where

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

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

$$\sum_{k=0}^{N} \frac{u_k}{u_0} = \frac{1}{(1-p)^N} \tag{27}$$

and accordingly,

$$u_0 = (1-p)^N. \tag{28}$$

B.3 Closed-Form Equation for u_k

By combining Equation 18 and Equation 28, we derive the closed-form equation for u_k as follows:

$$\begin{aligned}u_k &= \binom{N}{k} \left(\frac{p}{1-p}\right)^k u_0 \\ &= \binom{N}{k} \left(\frac{p}{1-p}\right)^k (1-p)^N \\ &= \binom{N}{k} p^k (1-p)^{N-k}\end{aligned}$$

Note that the same derivation is possible noting that the proposed Markov chain results in an *Engset* distribution [35].