Bounding the Population Size in XCS to Ensure Reproductive Opportunities

Martin V. Butz and David E. Goldberg

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

Abstract. Despite several recent successful comparisons and applications of the accuracy-based learning classifier system XCS, it is hardly understood how crucial parameters should be set in XCS nor how XCS can be expect to scale up in larger problems. Previous research identified a *covering challenge* in XCS that needs to be obeyed to ensure that the genetic learning process takes place. Furthermore, a *schema challenge* was identified that, once obeyed, ensures the existence of accurate classifiers. This paper departs from these challenges deriving a *reproductive opportunity bound*. The bound assures that more accurate classifiers get a chance for reproduction. The relation to the previous bounds as well as to the specificity pressure in XCS are discussed as well. The derived bound shows that XCS scales in a machine learning competitive way.

1 Introduction

The XCS classifier system has recently gained increasing attention. Especially in the realm of classification problems, XCS has been shown to solve many typical machine learning problems with a performance comparable to other traditional machine learning algorithms [15,1,7]

On the theory side, however, XCS is still rather poorly understood. There are few hints for parameter settings and scale-up analyses do not exist. Wilson [16,17] suggested that XCS scales polynomially in the number of concepts that need to be distinguished and in the problem length. However, this hypothesis has not been investigated to date.

This paper derives an important population size bound that needs to be obeyed to ensure learning. The derivation departs from the previously identified *covering challenge*, which requires that the genetic learning mechanism takes place, and the *schema challenge*, which requires that important (sufficiently specialized) classifiers are present in the initial population [3]. The two challenges were shown to result in specificity-dependent population size bounds.

In addition to these requirements, the population must be large enough to ensure that more accurate classifiers have reproductive opportunities. Following this intuition, we derive a *reproductive opportunity bound* that bounds the population size of XCS in $O(l^{k_d})$ where l denotes the problem length and k_d denotes the minimal schema order in the problem. Since k_d is usually small, Wilson's hypothesis holds that XCS scales polynomially in problem length l. In general, our analysis provides further insights on scale-up behavior, necessary parameter settings, and basic XCS functioning.

After a short overview of the XCS system, we review the previous identified problem bounds, identify a time dimension in the schema challenge, and then analyze the additional *reproductive opportunity bound*. Empirical confirmation of the derived bound is provided. Summary and conclusions complete the paper.

2 XCS in Brief

The XCS classifier system investigated herein is based on the work in [16,17, 10] and derived from the algorithmic description in [6]. This XCS overview provides only the details necessary for the rest of the paper. The interested reader is referred to the cited literature. For simplicity, we introduce XCS as a pure classifier. The results, however, should readily carry over to multi-step problems in which reward needs to be propagated.

We define a classification problem as a binary problem that provides problem instances $\sigma \in \{0,1\}^l$ (denoting problem length by l). After an instance is classified, the problem provides feedback in terms of scalar payoff $\rho \in \Re$ reflecting the quality of the classification. The task is to maximize this feedback effectively always choosing the correct classification.

XCS is basically a rule learning system. A population of rules, or *classifiers*, is evolved and adapted. Each rule consists of a condition part and a classification part. The condition part C specifies when the rule is active, or *matches*. It is coded by the ternary alphabet $\{0,1,\#\}$ (i.e. $C \in \{0,1,\#\}^l$) where a #-symbol matches both zero and one. The proportion of non-don't care symbols in the condition part determines the *specificity* of a classifier. In addition to condition and classification parts, classifiers specify a *reward prediction* p that estimates resulting payoff, *prediction error* ϵ that estimates the mean absolute deviation of p, and *fitness* F that measures the average relative scaled accuracy of p with respect to all competing classifiers. Further classifier parameters are *experience* exp, *time stamp* ts, *action set size estimate* as, and *numerosity* num.

XCS continuously evaluates and evolves its population. Rule parameters are updated by the Widrow-Hoff rule [14] and reinforcement learning techniques [13]. Effectively, the parameters approximate the average of all possible cases.

The population is evolved by the means of a covering mechanism and a genetic algorithm (GA). Initially, the population of XCS is empty. Given a problem instance, if there is no classifier that matches the instance for a particular classification, a covering classifier is generated that matches in that instance and specifies the missing classification. At each position of the condition, a #-symbol is induced with a probability of $P_\#$. A GA is applied if the average time in the action set $[A]$ since the last GA application, recorded by the time stamp ts, is greater than the threshold θ_{GA}. If a GA is applied, two classifiers are selected in $[A]$ for reproduction using fitness proportionate selection with respect to the fitness of the classifiers in $[A]$. The classifiers are reproduced and the children undergo mutation and crossover. In mutation, each attribute in C of each classifier is changed with a probability μ to either other value of the ternary alphabet (niche mutation is not considered herein). The action is mutated to any other possible action with a probability μ. For crossover, two-point crossover is applied with a probability χ. The parents stay in the population and compete with their offspring. The classifiers are inserted applying *subsumption deletion* in which classifiers are absorbed by experienced, more general, accurate classifiers.

If the number of (micro-)classifiers in the population exceeds the maximal population size N, excess classifiers are deleted. A classifier is chosen for deletion with roulette wheel selection proportional to its action set size estimate as. Further, if a classifier is sufficiently experienced ($exp > \theta_{del}$) and significantly less accurate than the average fitness in the population ($f < \delta * \sum_{cl \in [P]} f_{cl} / \sum_{cl \in [P]} num_{cl}$), the probability of being selected for deletion is further increased. Note that the GA is consequently divided into a reproduction process that takes place inside particular action sets and a deletion process that takes place in the whole population.

3 Specificity Change and Previous Problem Bounds

Previous investigations of XCS have proven the existence of intrinsic generalization in the system. Also, a covering challenge was identified that requires that the GA takes place as well as a schema challenge that requires that important classifiers are present in the initial population. This section gives a short overview of these bounds. Moreover, assuming that none of the important classifiers are present in the initial population, a time estimate for the generation of the necessary classifiers is derived.

3.1 Specificity Pressure

Since XCS reproduces classifiers in the action set but deletes classifiers from the whole population, there is an implicit generalization pressure inherent in XCS's evolutionary mechanism. The change in specificity due to reproduction can be derived from the expected specificity in an action set $s_{[A]}$ given current specificity in the population $s_{[P]}$ which is given by [2]:

$$s_{[A]} = \frac{s_{[P]}}{2 - s_{[P]}} \tag{1}$$

Additionally, mutation pushes towards an equal number of symbols in a classifier so that the expected specificity of the offspring s_o can be derived from parental specificity s_p [4]:

$$\Delta_m = s_o - s_p = 0.5\mu(2 - 3s_p) \tag{2}$$

Taken together, the evolving specificity in the whole population can be asserted considering the reproduction of two offspring in a reproductive event and the impact on the whole population [4] (ignoring any possible fitness influence):

$$s_{[P(t+1)]} = f_{ga}\frac{2(s_{[A]} + \Delta_m - s_{[P(t)]})}{N} \tag{3}$$

Parameter f_{ga} denotes the frequency the GA is applied on average which depends on the GA threshold θ_{ga} but also on the current specificity $s_{[P]}$ as well as the distribution of the encountered problem instances. While the specificity is initially determined by the don't care probability $P_\#$, the specificity changes over time as formulated by Equation 3. Thus, parameter $P_\#$ can influence XCS's behavior only early in a problem run.

From equation 3 we can derive the steady state value of the specificity $s_{[P]}$ as long as no fitness influence is present. Note that the speed of convergence is dependent on f_{ga} and N, the converged value is independent of these values.

$$\begin{gathered} s_{[A]} + \Delta_m - s_{[P]} = 0 \\ \frac{s_{[P]}}{2 - s_{[P]}} + \frac{\mu}{2}\left(2 - 3\frac{s_{[P]}}{2 - s_{[P]}}\right) = s_{[P]} \end{gathered} \tag{4}$$

Solving for $s_{[P]}$:

$$\begin{gathered} {s_{[P]}}^2 - (2.5\mu + 1)s_{[P]} + 2\mu = 0 \\ s_{[P]} = \frac{1 + 2.5\mu - \sqrt{6.25\mu^2 - 3\mu + 1}}{2} \end{gathered} \tag{5}$$

This derivation enables us to determine the converged specificity of a population given a mutation probability μ. Table 1 shows the resulting specificities in theory and empirically determined on a random Boolean function (a Boolean function that returns randomly either zero or one and thus either 1000 or zero reward). The empirical results are slightly higher due to higher noise in more specific classifiers, the biased accuracy function, the fitness biased deletion method, and the parameter initialization method [4]. Although mutation's implicit specialization pressure (as long as $s_{[P]} < 2/3$) can be used to control the specificity of the population, too high mutation most certainly disrupts useful problem substructures. Thus, in the future other ways might be worth exploring to control specificity in $[P]$.

Table 1. Converged Specificities in Theory and in Empirical Results

μ	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
$s_{[P]}$, theory	0.040	0.078	0.116	0.153	0.188	0.223	0.256	0.288	0.318	0.347
$s_{[P]}$, empirical	0.053	0.100	0.160	0.240	0.287	0.310	0.329	0.350	0.367	0.394

3.2 Covering and Schema Bound

Before specificity change and evolutionary pressure in general apply, though, it needs to be assured that the GA takes place in the first place. This is addressed by the *covering challenge* [3]. To meet the challenge, the probability of covering a given random input needs to be sufficiently high (assuming uniformly distributed #-symbols):

$$P(cover) = 1 - \left(1 - \left(\frac{2 - s_{[P]}}{2}\right)^l\right)^N > 0 \tag{6}$$

Once the covering bound is obeyed, the GA takes place and specificity changes according to Equation 3. To evolve accurate classifiers, however, more accurate classifiers need to be present in a population. This leads to the *schema challenge* which determines the probability of the existence of a *schema representative* in a population with given specificity [3].

Order of Problem Difficulty. To understand the schema challenge, we use the traditional schema notation [9,8] of GAs to define a *schema representative.* Hereby, a schema can be characterized by its *order* k, which denotes the number of specified attributes, and its *defining length* δ, which is the distance between the outermost specified attributes. For example, schema `**11***0*000*` has order six and defining length nine whereas schema `**1*0*0******` has order three and defining length four. As proposed in [3], a classifier is said to be a *schema representative* if its condition part has at least all k attributes of the corresponding order k schema specified and its action corresponds to the schema corresponding action. For example, a classifier with condition `C=##1#0#0##000#` would be a representative of the second schema but not of the first one. Consequently, the specificity of a representative is at least equal to the order k of the schema it represents. If the specificity of a representative cl exactly equals k, then it is said to be *maximally general* with respect to the schema.

A problem is now said to be of *order of problem difficulty* k_d if the smallest order schema that supplies information gain is of order k_d. Thus, if a problem is of order of problem difficulty k_d any schema of order less than k_d will have the same (low) accuracy and essentially the same accuracy as the completely general schema (order $k = 0$). Similarly, only classifiers that have at least k_d attributes specified can have higher accuracy, higher fitness, and can thus provide *fitness guidance.* Thus, a problem of order of difficulty k_d can only be solved by XCS if representatives of the order k_d schema exist (or are generated at random) in the population.

Given a particular problem, any schema can be assigned a probability of being correct. It has been shown in [3] that the more consistently a classifier is correct or incorrect, the more accurate the classifier will be. Thus, a problem provides fitness guidance from the over-general side, if classifiers that specify parts of the relevant attributes are more consistently correct/incorrect.

An extreme case of this is the *parity problem* in which all k relevant bits need to be specified to reach any fitness guidance. The specification of any subset of those k bits will result in a 50/50 chance for correct classification and consequently a base accuracy which is equal to the completely general classifier. Thus, in the parity problem there is no fitness guidance before reaching full accuracy (effectively going from zero to completely accuracy in one step) so that the order of problem difficulty in the parity problem is equal to the size of the parity.

Note that the order of problem difficulty measure also affects many other common machine learning techniques. For example, the inductive decision tree learner `C4.5` [12] totally relies on the notion of information gain. If a problem has an order of difficulty $k_d > 1$, `C4.5` would basically decide at random on which attribute to expand first. Consequently, `C4.5` would generate an inappropriately large decision tree.

Schema Representative. With this notion we can formulate the probability that there exists an order k schema representative in a population that has specificity $s_{[P]}$ [3]:

$$P(\textit{representative in } [P]) = 1 - \left(1 - \frac{1}{n}\left(\frac{s_{[P]}}{2}\right)^k\right)^N > 0 \tag{7}$$

where n denotes the number of possible outcome classes.

Similar to the probability of a representative in a population, the probability that a particular classifier in the population is a representative as well as the expected number of representatives in the population can be assessed:

$$\begin{aligned} P(\textit{representative}) &= \frac{1}{n}(s_{[P]})^k \\ E(\textit{representative}) &= N\frac{1}{n}(s_{[P]})^k \end{aligned} \tag{8}$$

Requiring that at least one representative exists in a particular population with specificity $s_{[P]}$ we derive the following *representative bound* on the population size:

$$\begin{aligned} E(\textit{representative}) &> 1 \\ N &> \frac{n}{s_{[P]}^k} \end{aligned} \tag{9}$$

While this bound is rather easy to obey given a reasonable high specificity, the more interesting result is the constraint on the specificity given an actual population size N:

$$s_{[P]} > \left(\frac{n}{N}\right)^{1/k} \tag{10}$$

This shows that with increasing population size, the required specificity decreases.

3.3 Extension in Time

As mentioned above, initially the population has a specificity of $s_{[P]} = 1 - P_{\#}$. From then on, as long as the population is not over-specific and the GA takes place, the specificity changes according to Equation 3. For example, say we set $P_{\#} = 1.0$ so that the initial specificity will be zero, the specificity will gradually increase towards the equilibrium specificity.

The time dimension is determined from the time it takes that a representative is created due to random mutations. Given a current specificity $s_{[P]}$, we can determine the probability that a representative of a schema with order k is generated in a GA invocation:

$$P(\text{generation of representative}) = (1-\mu)^{s_{[P]}k} \cdot \mu^{(1-s_{[P]})k} \tag{11}$$

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

$$\begin{aligned} &E(t(\text{generation of representative})) = \\ &1/P(\text{generation of representative}) = \\ &\left(\frac{\mu}{1-\mu}\right)^{s_{[P]}k} \mu^{-k} \end{aligned} \tag{12}$$

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

$$\begin{aligned} \mu^{-k} &< \Theta \\ \mu &> \Theta^{\frac{1}{-k}} \end{aligned} \tag{13}$$

This representative bound actually relates to the existence of a representative bound determined above in equation 10. Setting Θ equal to N/n we get the same bound (s can be approximated by $a \cdot \mu$ where $a \approx 2$, see Table 1). Assuming that no representative exists in the population, the order of difficulty of a problem k_d influences speed of learning to take place due to possible delayed supply of representatives. In the extreme case of the parity function, the bound essentially bounds learning speed since a representative is automatically fully accurate.

4 Reproductive Opportunity Bound

Given the presence of representatives and GA application, another crucial requirement remains to be satisfied: The representative needs to get the chance to reproduce. Since GA selection for reproduction takes place in action sets, reproduction can only be assured if the representative will be part of an action set before it is deleted. To derive an approximate population size bound to ensure reproduction, we can require that the probability of deletion of a representative is smaller than the probability that the representative is part of an action set.

The probability of deletion is

$$P(deletion) = \frac{1}{N} \tag{14}$$

assuming that there are no action set size estimate influences nor any fitness influences in the deletion process. This is a reasonable assumption given that the action set size estimate is inherited from the parents and fitness is initially decreased by 0.1.

Given a particular classifier *cl* that has specificity s_{cl}, we can determine the probability that *cl* will be part of an action set given random inputs.

$$P(\text{in } [A]) = \frac{1}{n} 0.5^{l \cdot s_{cl}} \tag{15}$$

This probability is exact if binary input strings are encountered that are uniformly distributed over the whole problem instance space $\{0,1\}^l$.

Combining equations 14 and 15 we can now derive a constraint for successful evolution in XCS by requiring that the probability of deletion is smaller than the probability of reproduction. Since two classifiers are deleted at random from the population but classifiers are reproduced in only one action set (the current one), the probability of a reproductive opportunity needs to be larger than approximately twice the deletion probability:

$$\begin{aligned} P(\text{in}[A]) &> 2P(\text{deletion}) \\ \frac{1}{n} 0.5^{l \cdot s_{cl}} &> \frac{2}{N} \\ N &> n 2^{l \cdot s_{cl} + 1} \end{aligned} \tag{16}$$

This bounds the population size by $O(n2^{ls})$. Note that the specificity s depends on the order of problem difficulty k_d and the chosen population size N as expressed in Equation 10. Since k_d is usually small, s can often be approximated by $s = 1/l$ so that the bound diminishes. However, in problems in which no fitness guidance can be expected from the over-general side up to order $k_d > 1$, specificity s cannot be approximated as easily.

The expected specificity of such a representative of an order k schema can be estimated given a current specificity in the population $s_{[P]}$. Given that the classifier specifies all k positions its expected average specificity will be approximately:

$$E(s(\text{representative of schema of order } k)) = \frac{k + (l-k)s_{[P]}}{l} \tag{17}$$

Substituting s_{cl} in equation 16 with the expected specificity of a representative of a schema of order $k = k_d$ necessary in a problem of order of difficulty k_d, the population size N can be bounded by

$$\begin{aligned} N &> n2^{l \cdot \frac{k_d + (l-k_d)s_{[P]}}{l}+1} \\ N &> n2^{k_d+(l-k_d)s_{[P]}+1} \end{aligned} \tag{18}$$

This bound ensures that classifiers necessary in a problem of order of difficulty k_d will get reproductive opportunities. Once the bound is satisfied, existing representatives of the necessary order k_d schemata are ensured to reproduce and XCS is enabled to evolve a more accurate population.

Note that this population size bound is actually exponential in schema order k_d and in string length times specificity $l \cdot s_{[P]}$. This would mean that XCS would scale exponential in the problem length which is certainly highly undesirable. However, as seen above the necessary specificity actually decreases with increasing population sizes.

Considering this, we show in the following that out of the specificity constraint a general *reproductive opportunity bound* (ROP-bound) can be derived that shows that population size needs to grow in $O(l^{k_d})$. Equations 10 and 18 can both be denoted in O-notation:

$$\begin{aligned} s_{[P]} &= O((\frac{n}{N})^{1/k}) \\ N &= O(2^{ls_{[P]}}) \end{aligned} \tag{19}$$

Ignoring additional constants, we can derive the following population size bound which solely depends on problem length l and order of difficulty k_d.

$$\begin{aligned} N &> 2^{l(\frac{n}{N})^{1/k_d}} \\ N(\log_2 N)^{k_d} &> nl^{k_d} \end{aligned} \tag{20}$$

This *reproductive opportunity bound* essentially shows that population size N needs to grow approximately exponential in the order of the minimal building block k_d (i.e., order of difficulty) and polynomial in the string length.

$$N = O(l^{k_d}) \tag{21}$$

Note that in the usual case k_d is rather small and can often be set to one (see Sect. 3.2). As shown in the extreme case of the parity problem, though, k_d can be larger than one.

5 Bound Verification

To verify the derived reproductive opportunity bound as well as the time dimension of the representative bound, we apply XCS to a Boolean function problem of order of difficulty k_d where k_d is larger than one. The hidden-parity problem,

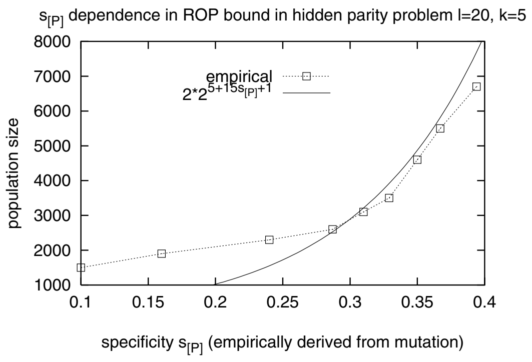

Fig. 1. Minimal population size depends on applied specificity in the population (manipulated by varying mutation rates). When mutation and thus specificity is sufficiently low, the bound becomes obsolete

originally investigated in [11], is very suitable to manipulate k_d. The basic problem is represented by a Boolean function in which k relevant bits determine the outcome. If the k bits have an even number of ones, the outcome will be one and zero otherwise. As discussed above, the order of difficulty k_d is equivalent to the number of relevant attributes k.

Figure 1 shows that the reproductive opportunity bound is well approximated by equation 16.[1] The experimental values are derived by determining the population size needed to reliably reach 100% performance. The average specificity in the population is derived from the empirical specificities in Table 1. XCS is assumed to reach 100% performance reliably if all 50 runs reached 100% performance after 200,000 steps. Although the bound corresponds to the empirical points when specificity is high, in the case of lower specificity the actual population size needed departs from the approximation. The reason for this is the time dimension of the representative bound determined above. The lower the specificity in the population, the longer it takes until a representative is actually generated (by random mutation). Thus, the representative bound extends in time rather than in population size.

To validate the $O(l^k)$ bound of equation 21, we ran further experiments in the hidden parity problem with $k = 4$, varying l and adapting mutation rate μ (and thus specificity) appropriately.[2] Results are shown in Fig. 2. The bound was derived from experimental results averaged over 50 runs determining the population size for which 98% performance is reached after 300000 steps.

[1] XCS with tournament selection was used [5]. If not stated differently, parameters were set as follows: $\beta = 0.2$, $\alpha = 1$, $\epsilon_0 = 1$, $\nu = 5$, $\theta_{ga} = 25$, $\tau = 0.4$, $\chi = 1.0$, uniform crossover, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $P_{\#} = 1.0$, $\theta_{sub} = 20$.

[2] Mutation was set as follows with respect to l: $l = 10, \mu = 0.10; l = 15, \mu = 0.075; l = 20, \mu = 0.070; l = 25, \mu = 0.065; l = 30, \mu = 0.060; l = 35, \mu = 0.057; l = 40, \mu = 0.055; l = 45, \mu = 0.050; l = 50, \mu = 0.050; l = 55, \mu = 0.048; l = 60, \mu = 0.048; l = 65, \mu = 0.048$.

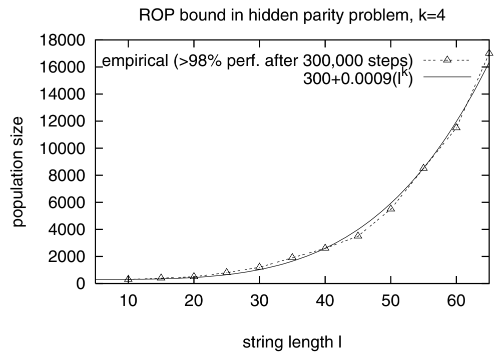

Fig. 2. The *reproductive opportunity bound* can be observed altering problem length l and using optimal mutation μ, given a problem of order of difficulty k_d. The experimental runs confirm that the evolutionary process indeed scales up in $O(l^k)$

With appropriate constants added, the experimentally derived bound is well-approximated by the theory.

6 Summary and Conclusions

In accord with Wilson's early scale-up hypothesis [16,17], this paper confirms that XCS scales-up polynomially in problem length l and exponentially in order of problem difficulty and thus in a machine learning competitive way. The empirical analysis in the hidden parity function confirmed the derived bounds. Combined with the first derived scale-up bound in XCS, the analysis provides several hints on mutation and population size settings. All bounds so far identified in XCS assure that the GA takes place (covering bound), that initially more accurate classifiers are supplied (schema bound—in population size and generation time), and that these more accurate classifiers are propagated (reproductive opportunity bound).

Many issues remain to be addressed. First, we assumed that crucial schema representatives provide fitness guidance towards accuracy. Type, strength, and reliability of this guidance require further investigation. Second, classifier parameter values are always only approximations of their assumed average values. Variance in these values needs to be accounted for. Third, although reproductive opportunity events are assured by our derived bound it needs to be assured that recombination effectively processes important substructures and mutation does not destroy these substructures. Forth, the number of to-be represented classifier niches (denoted by $[O]$ in [11]) needs to be taken into account in our model. Fifth, the distribution of classifier niches such as overlapping niches and obliqueness needs to be considered [15]. Integrating these points should allow us to derive a rigorous theory of XCS functioning and enable us to develop more competent and robust XCS learning systems.

Acknowledgment. We are grateful to Kumara Sastry and the whole IlliGAL lab for their help on the equations and for useful discussions. The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. Additional support from the Automated Learning Group (ALG) at the National Center for Supercomputing Applications (NCSA) is acknowledged. Additional funding from the German research foundation (DFG) under grant DFG HO1301/4-3 is acknowledged. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Bernadó, E., Llorà, X., Garrell, J.M.: XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001. Springer-Verlag, Berlin Heidelberg (2002) 115–132
2. Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.W.: Theory of generalization and learning in XCS. IEEE Transaction on Evolutionary Computation (submitted, 2003) also available as IlliGAL technical report 2002011 at http://www-illigal.ge.uiuc.edu/techreps.php3.
3. Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.W.: How XCS evolves accurate classifiers. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 927–934
4. Butz, M.V., Pelikan, M.: Analyzing the evolutionary pressures in XCS. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001) (2001) 935–942
5. Butz, M.V., Sastry, K., Goldberg, D.G.: Tournament selection in XCS. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003) (2003) this volume.
6. Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000. Springer-Verlag, Berlin Heidelberg (2001) 253–272
7. Dixon, P.W., Corne, D.W., Oates, M.J.: A preliminary investigation of modified XCS as a generic data mining tool. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: 4th International Workshop, IWLCS 2001. Springer-Verlag, Berlin Heidelberg (2002) 133–150
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
9. Holland, J.H.: Adaptation in natural and artificial systems. Universtiy of Michigan Press, Ann Arbor, MI (1975) second edition 1992.

10. Kovacs, T.: Deletion schemes for classifier systems. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) (1999) 329–336
11. Kovacs, T., Kerber, M.: What makes a problem hard for XCS? In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000. Springer-Verlag, Berlin Heidelberg (2001) 80–99
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
13. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press, Cambridge, MA (1998)
14. Widrow, B., Hoff, M.: Adaptive switching circuits. Western Electronic Show and Convention **4** (1960) 96–104
15. Wilson, S.W.: Mining oblique data with XCS. In Lanzi, P.L., Stolzmann, W., Wilson, S.W., eds.: Advances in Learning Classifier Systems: Third International Workshop, IWLCS 2000. Springer-Verlag, Berlin Heidelberg (2001) 158–174
16. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
17. Wilson, S.W.: Generalization in the XCS classifier system. Genetic Programming 1998: Proceedings of the Third Annual Conference (1998) 665–674