Effect of Pure Error-Based Fitness in XCS

Martin V. Butz[1], David E. Goldberg[2], and Pier Luca Lanzi[2,3]

[1] Department of Cognitive Psychology,
University of Würzburg, 97070 Würzburg, Germany
butz@psychologie.uni-wuerzburg.de
[2] Illinois Genetic Algorithms Laboratory,
University of Illinois at Urbana-Champaign, Urbana, Illinois, USA
{deg,lanzi}@illigal.ge.uiuc.edu
[3] Dipartimento di Elettronica e Informazione
Politecnico di Milano
Milano, Italy
lanzi@elet.polimi.it

Abstract. The accuracy-based fitness approach in XCS is one of the most significant changes in comparison with original learning classifier systems. Nonetheless, neither the scaled accuracy function, nor the importance of the relative fitness approach has been investigated in detail. The recent introduction of tournament selection to XCS has shown to make the system more independent from parameter settings and scaling issues. The question remains if relative accuracy itself is actually necessary in XCS or if the evolutionary process could be based directly on error. This study investigates advantages and disadvantages of pure error-based fitness vs. relative accuracy-based fitness in XCS.

1 Introduction

Recent advances in XCS understanding have shown that the accuracy-based fitness approach can guide the evolutionary process to the discovery of accurate, maximally general classifiers [7]. Additionally, with the introduction of tournament selection, XCS gained a more reliable and persistent pressure towards accuracy [9]. However, it did not become clear why accuracy needs to be scaled nor why fitness is derived from the *relative* accuracy.

This study investigates the fitness approach in XCS. The relative accuracy-based fitness approach underlies several peculiar parameter choices which need to be investigated and clarified. Moreover, although XCS's fitness approach was successful in many different investigations (e.g. [2,11,4]), it is not clear if the additional accuracy bias is necessary for a successful evolutionary process in XCS. In fact, it seems possible that XCS selection with tournament selection could be solely based on minimizing error instead of maximizing accuracy. In this way, the additional accuracy bias would become irrelevant and parameter estimations should reach less noisy values faster.

The remainder of this study is structured as follows. The next section gives a short overview over the XCS system with the relevant parameter initialization

X. Llorà et al. (Eds.): IWLCS 2003-2005, LNAI 4399, pp. 104–114, 2007.

method and update methods. Next, we study the effect of basing selection directly on error instead of accuracy-based fitness. Summary and conclusions conclude the study.

2 XCS Overview

XCS is a very general learning mechanism that combines gradient-based optimization of predictions with evolutionary-based space partitioning. The partitions evolve to enable maximally accurate predictions. While XCS was also successfully applied in multi-step problems [22,15,16,1], we restrict this study to classification problems to avoid the additional problem of reward propagation. However, the insights of this study should readily carry over to multi-step problems. This section introduces XCS as a pure classification system providing the necessary details to comprehend the remainder of this work. For a more complete introduction to XCS the interested reader is referred to the original paper [22] and the algorithmic description [10].

We define a classification problem as a problem that consists of problem instances $s \in \mathcal{S}$ that need to be classified by XCS with one of the possible classifications $a \in \mathcal{A}$. The problem then provides scalar payoff $R \in \Re$ with respect to the made classification. The goal for XCS is to choose the classification that results in the highest payoff. To do that, XCS is designed to learn a complete mapping from any possible $s \times a$ combination to an accurate payoff value. To keep things simple, we investigate problems with Boolean input and classification, i.e. $\mathcal{S} \subseteq \{0,1\}^L$ where L denotes the fixed length of the input string and $\mathcal{A} = \{0,1\}$.

XCS evolves a population $[P]$ of rules, or *classifiers*. Each classifier in XCS consists of five main components. The condition $C \in \{0,1,\#\}^L$ specifies the subspace of the problem instances in which the classifier is applicable, or *matches*. The "don't care" symbol # matches in all input cases. The action part $A \in \mathcal{A}$ specifies the advocated action, or classification. The payoff prediction p approaches the average payoff encountered after executing action A in situations in which condition C matches. The prediction error ε estimates the average deviation, or error, of the payoff prediction p. The fitness reflects the average relative accuracy of the classifier with respect to other overlapping classifiers.

XCS iteratively updates its knowledge base with respect to each problem instance. Given current input s, XCS forms a *match set* $[M]$ consisting of all classifiers in $[P]$ whose conditions match s. If an action is not represented in $[M]$, a covering classifier is created that matches s (#-symbols are inserted with a probability of $P_\#$ at each position). For each classification, XCS forms a *payoff prediction* $P(a)$, i.e. the fitness-weighted average of all reward prediction estimates of the classifiers in $[M]$ that advocate classification a. The payoff predictions determine the appropriate classification. After the classification is selected and sent to the problem, payoff R is provided according to which XCS updates all classifiers in the current action set $[A]$ which comprises all classifiers in $[M]$ that advocate the chosen classification a. After update and possible GA invocation, the next iteration starts.

Prediction and prediction error parameters are update in $[A]$ by $p \leftarrow p + \beta(R - p)$ and $\varepsilon \leftarrow \varepsilon + \beta(|R - p| - \varepsilon)$ where β ($\beta \in [0, 1]$) denotes the *learning rate*. The fitness value of each classifier in $[A]$ is updated according to its current scaled relative accuracy κ':

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

$$F \leftarrow F + \beta(\kappa' - F) \tag{2}$$

The parameter ε_0 ($\varepsilon_0 > 0$) controls the tolerance for prediction error ε; parameters α ($\alpha \in (0, 1)$) and ν ($\nu > 0$) are constants controlling the rate of decline in accuracy κ when ε_0 is exceeded. The accuracy values κ in the action set $[A]$ are then converted to set-relative accuracies κ'. Finally, classifier fitness F is updated towards the classifier's current set-relative accuracy. All parameters except for fitness F are updated using the *moyenne adaptive modifée* technique [19]. This technique sets parameter values directly to the average of the so far encountered cases as long as the experience of a classifier is still less than $1/\beta$. Each time the parameters of a classifier are updated, the experience counter *exp* of the classifier is increased by one.

A GA is invoked in XCS if the average time since the last GA application on the classifiers in $[A]$ exceeds threshold θ_{ga}. The GA selects two parental classifiers using roulette-wheel selection [22] or the recently introduced tournament selection [9]. Two offspring are generated reproducing the parents and applying crossover and mutation. Parents stay in the population competing with their offspring. We apply free mutation in which each attribute of the offspring condition is mutated to the other two possibilities with equal probability. Parameters of the offspring are inherited from the parents, except for the experience counter *exp* which is set to one, the numerosity *num* which is set to one, and the fitness F which is multiplied by 0.1. In the insertion process, *subsumption deletion* may be applied [23] to stress generalization.

The population of classifiers $[P]$ is of fixed size N. Excess classifiers are deleted from $[P]$ with probability proportional to an estimate of the size of the action sets that the classifiers occur in (stored in the additional parameter as). If the classifier is sufficiently experienced and its fitness F is significantly lower than the average fitness of classifiers in $[P]$, its deletion probability is further increased.

3 Error-Based Selection

Although an error-based selection method still pursues the XCS goal of evolving a complete and accurate reward map of a problem several differences can be identified. This section discusses these differences and experimentally investigates error-based fitness in XCS.

3.1 Major Differences

As mentioned in the XCS overview, selection is usually based on the set-relative accuracy derived fitness estimate of a classifier. In offspring classifiers this fitness is usually derived from the parents (sometimes also from the average fitness in the population) and multiplied by 0.1 to be pessimistic about the offspring quality. Dependent on the learning rate β, the moyenne adaptive modifée (MAM) technique, the experience counter, and the accuracy scaling, more accurate offspring reaches a fitness value higher than the parental value after a certain amount of updates. Only then the more-accurate offspring has the chance to outperform its parents and take-over the specific environmental niche it covers. The number of influences suggest that complex interactions of different factors can occur.

Similar to the fitness approach, though, it seems also possible to base selection directly on the prediction error estimate of a classifier. While accuracy-based fitness needs to be maximized, error-based fitness needs to be minimized. Additional effects are expectable, though, since the error estimate is directly derived from the parental value (without a pessimistic increase) and the error estimate is not set relative, effectively disabling fitness sharing. While the former factor should have the effect that offspring sometimes causes additional disruption, the latter factor might result in weaker niche support pressure. These factors are investigated in our experimental study.

Interestingly, though, due to the lack of fitness sharing, additionally, overlapping classifiers are enabled in this framework. The relative-accuracy-based fitness approach in the original XCS causes the evolution of non-overlapping niches that cover the whole reward map of a learning problem (see e.g. [13,14] for further analyses). Error-based fitness will cause the evolution of a similar complete reward map but allows overlapping classifiers. This might be advantageous in unevenly overlapping niches, but has the drawback that more classifiers need to be sustained to continuously cover the whole problem space. The additional classifiers also undergo additional competition due to the unrestricted population-wide deletion technique.

3.2 Implementation

Error-based selection is realized applying tournament selection. Instead of maximizing the fitness estimate of a classifier, the error estimate is minimized. Thus, the classifier wins in the current tournament in an action set that has the lowest reward prediction error estimate. Parameter updates are not changed.

Additionally, to free XCS completely from the fitness evaluation, the prediction array needs to be formed with respect to a classifier's error estimate and not to its fitness estimate. Since the error estimate in young classifiers is very noisy, the reward prediction estimate is less trusted than in elder classifiers. Widrow & Stearns (1985) formalized how the reward prediction error can be expected to vary with respect to the number of encountered reward prediction updates [20].

Assuming the encountering of a perfect signal P and the initial estimate p_i, then the error of the actual estimate $p(t)$ can be determined as follows [20]:

$$p(t) = P + (1-\beta)^t(p_i - P) \tag{3}$$

Assuming the worst-case initialization error $\epsilon_{wc} = \texttt{max}\{p_i; P_{max} - p_i\}$ as the initial error and assuming furthermore a perfect reward signal from then on, the maximal difference from the actual average encountered reward can be estimated as follows:

$$\Delta p = (1-\beta)^{exp}\epsilon_{wc} \tag{4}$$

Since the prediction error of a classifier can reach on average half the maximal reward prediction $P_{max}/2$ (temporarily it might also lie a little above this value), the maximal error in the reward prediction error can be determined as follows denoting $\epsilon_{\epsilon wc}$ as the maximum possible error of the error and assuming a perfect signal.

$$\Delta\epsilon = (1-\beta)^{exp}\epsilon_{\epsilon wc} \tag{5}$$

The actual error of a classifier can now be estimated somewhat pessimistic as the actual error estimate plus the worst-case differing amount (assuming a perfect signal).

$$\epsilon' = \epsilon + \Delta\epsilon \tag{6}$$

The prediction array may now be weighted according to the estimated error ϵ' in conjunction with the actual reward prediction value p:

$$PA(a) = \frac{\sum_{cl\in[M]\wedge cl.A=a} cl.p \cdot 1/cl.\epsilon' \cdot cl.num}{\sum_{cl\in[M]\wedge cl.A=a} 1/cl.\epsilon' \cdot cl.num} \tag{7}$$

This prediction array determination consequently ignores fitness but weights the reward estimates according to the actual inverse error estimate. Finally, deletion cannot be biased on the fitness estimate of a classifier as originally proposed and investigated in [12]. Consequently, deletion is proportional to the action set size estimate alone as in the original XCS implementation [22]. The next section investigates the impact of these modifications in several typical Boolean function problems.

3.3 Experimental Investigation

Several questions need to be investigated in the new approach. First, the question is if in fact overlapping, accurate classifiers evolve. Next, the speed of evolution will show if the direct error dependence allows a faster or slower detection of the relevant environmental niches and thus, if performance speed increases or decreases. Finally, due to the additional overlapping classifiers, the support of each environmental niche needs to be investigated. Will XCS be able to sustain the representation of the complete problem with the same number of classifiers?

To answer these questions, we apply XCS to the multiplexer function [22,6] and the count ones problem [6,8].

Table 1 shows the typical difference in the classifier lists of relative-accuracy-based fitness and error-based fitness in the six multiplexer. While in the relative case mainly non-overlapping classifiers evolve, that explicitly identify each environmental niche, in the error-based case those classifiers evolve as well as classifiers that overlap two niches. For example, niches `01*0**-1` and `11***0-1` (specifying accurately the incorrect class) would be represented perfectly by the similar classifiers substituting don't care symbols for the star symbols. These evolve in the relative-accuracy-based case. However, in the error-based case, also the overlapping classifiers gain a high numerosity value such as classifier `#1#0#0-1`. Note that in the exemplar runs, the maximal population size was set to $N = 2000$ so that niche support was not a problem in this case. The overlapping classifiers gain a similar numerosity (on average) as the non-overlapping ones do. Hardly any pressure towards the non-overlapping classifiers can be detected.

Table 1. Typical resulting classifier list for relative-accuracy-based fitness and error-based fitness in the 6-multiplexer problem

Relative-Accuracy Based						
C	A	p	ϵ	F	num	exp
11###0	1	0	0	0.836775	85	5134
11###1	1	1000	0	0.792768	73	5478
10##0#	1	0	0	0.702730	67	5847
10##1#	1	1000	0	0.653202	59	5270
01#0##	1	0	0	0.471205	49	5306
01#1##	1	1000	0	0.418793	38	5306
01#00#	1	0	0	0.252941	28	1976
001###	1	1000	0	0.301881	28	5726
#00#0#	1	0	0	0.242931	27	4925
000###	1	0	0	0.328251	27	5529
01#01#	1	0	0	0.234058	26	2557
0010##	1	1000	0	0.272719	25	2095
10##10	1	1000	0	0.256431	24	2269
10##01	1	0	0	0.232770	24	2481
01#10#	1	1000	0	0.242531	22	2570
01#0#1	1	0	0	0.210961	22	2636
01#11#	1	1000	0	0.222898	20	2651
000##1	1	0	0	0.230527	20	2740
001#0#	1	1000	0	0.204827	20	2786
001##0	1	1000	0	0.198300	19	1849
01#1#0	1	1000	0	0.214924	19	2692
000#1#	1	0	0	0.222182	19	2667
001##1	1	1000	0	0.202509	19	2867
#1#0#0	1	0	0	0.170386	18	5351

Error Based						
C	A	p	ϵ	F	num	exp
01#0##	1	0	0	0.900787	106	5994
#1#1#1	1	1000	0	0.637662	91	4973
1###00	1	0	0	0.587410	82	5651
#01#1#	1	1000	0	0.532509	81	2592
0#11##	1	1000	0	0.429461	65	3552
000###	1	0	0	0.712403	63	5175
#00#0#	1	0	0	0.435288	62	4410
#1#0#0	1	0	0	0.369763	46	5853
11###1	1	1000	0	0.504067	41	4982
10##10	1	1000	0	0.412228	38	325
10##0#	1	0	0	0.491715	36	4408
01#1##	1	1000	0	0.409011	32	1174
11###0	1	0	0	0.445064	32	4524
10##1#	1	1000	0	0.288221	28	1054
1###11	1	1000	0	0.270195	27	5872
001##1	1	1000	0	0.239064	25	466
001###	1	1000	0	0.270045	19	1891
0#11#0	1	1000	0	0.139190	13	97
#00#00	1	0	0	0.049742	8	75
0001##	1	0	0	0.053201	5	67
#01#11	1	1000	0	0.043135	5	102
#1#1#0	1	405	501	0.000000	4	1654
1###0#	1	161	302	0.000000	3	191
00#0##	1	519	509	0.000000	3	316

Further experiments in the larger 20 multiplexer problem are displayed in Figure 1 showing the normal multiplexer problem, and the problem with additional Gaussian noise (adding a Gaussian Noise with standard deviation $\sigma = 300$ on the provided reward reflecting noise in the fitness evaluation function).

All runs show that XCS with error-based fitness is able to solve the problem as well. The evolutionary speed is slightly decreased, that is, perfect performance is reached after a larger number of steps in comparison to relative-accuracy-based fitness. Part of the explanation for this decrease in learning speed can be attributed to the larger number of classifiers that is evolved. Additionally,

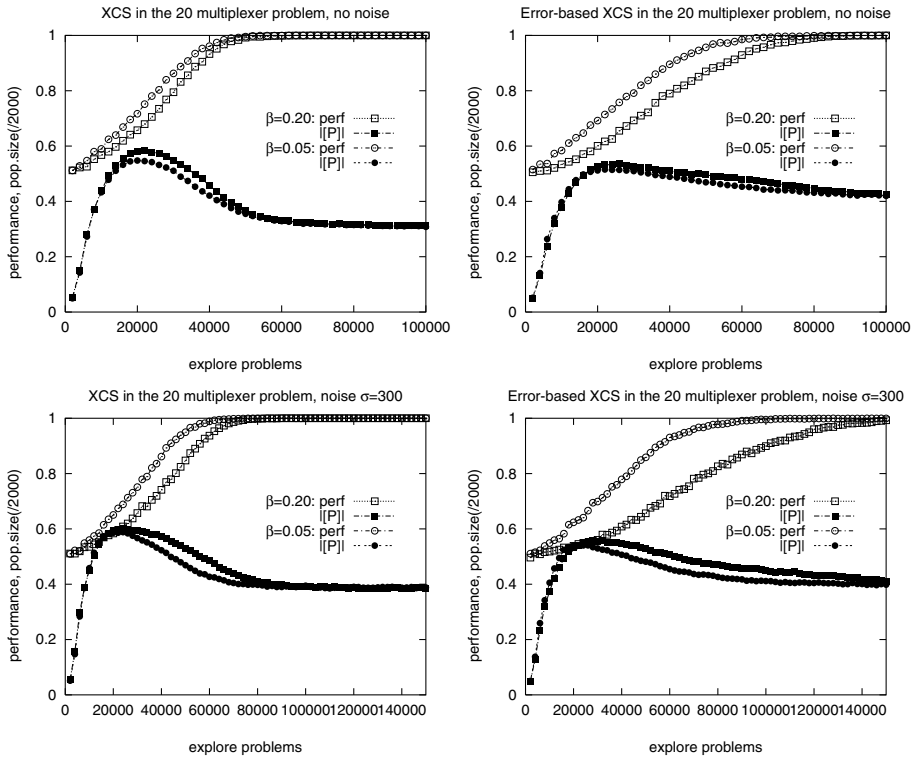

Fig. 1. Slight advantages due to relative-accuracy based selection can be observed in the multiplexer problem. However, the simplicity of error-based selection remains appealing.

parameter initialization issues appear relevant. Since fitness is decreased by 0.1 in offspring classifiers and fitness is updated by the Widrow-Hoff rule from the beginning, disruption by young classifiers appears to be prevented better in the fitness-based selection case than in the error-based selection case.

The relative fitness approach also results in a slightly stronger generalization pressure. The pressure appears to be mainly due the initial decrease in offspring fitness. The decrease in fitness assures that similarly accurate parents win the tournament against their offspring. More specialized classifiers undergo parameter updates less frequently so that the more specialized a classifier the longer it takes for it to exceed its parent's fitness. Thus, more generalized similarly accurate classifiers reach higher fitness values faster.

Besides the multiplexer problem, we experiment with the count ones problem in which overlapping niches need to be sustained to ensure the representation of a complete problem solution. Besides Gaussian noise, we also added alternating noise, in which the incorrect reward is provided with a probability of 0.15, reflecting noise with incorrect classification cases [6]. Similar performance

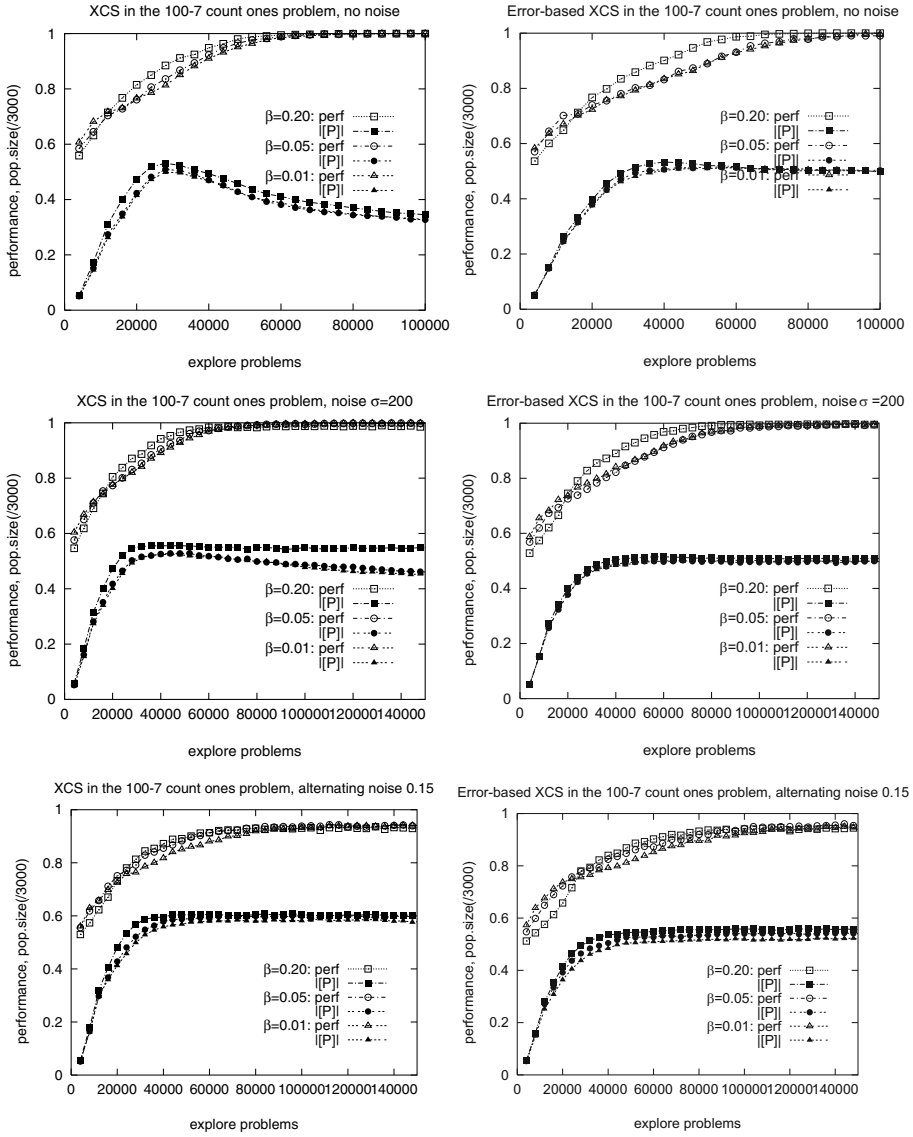

Fig. 2. Differences in the count ones problem with string length 100 and 7 relevant bits are minor. Lower β rates decrease convergence speed but increase accuracy.

observations can be made. The error-based approach again suffers more from more offspring disruption so that a reliable 100% performance is reached slighlty slower. However, particularly in the noisy problem cases, XCS with error-based selection reaches a slightly higher performance level. Thus, fitness sharing may cause disruption in problems in which overlapping niches actually need to be sustained for a complete problem solution.

4 Summary and Conclusions

Summing up, we could show that an XCS in which selection is purely based on error is able to solve similar Boolean function problems as the normal XCS. Hereby, performance was slightly worse than in the original XCS approach but with the gain of having less parameters per classifier and depending less on learning rate β. Moreover, the peculiar accuracy-scaling function is not necessary anymore. Parameter estimates can now be directly inherited from the parents. An additional parameter estimate decrease is not necessary.

Despite the successful application, it became also clear that the approach deserves further investigation. A similar error estimate as done for the prediction array calculation might be useful in the selection mechanism to prevent disruption in the error-based case. Other mechanisms are imaginable to prevent disruption but still ensure detection of better classifiers fast.

In conclusion, the results show that fitness sharing is actually not necessary in the XCS framework. Niching is assured due to the niche-based reproduction in conjunction with population-wide deletion. Thus, while fitness-sharing is very likely to be mandatory in other LCS frameworks, such as the ZCS system [21,3], niching in XCS is accomplished by the niche-based reproduction mechanism. This niching effect and its impact on population sizing in XCS is investigated in detail elsewhere [5,4].

The study also points out that parameter initialization in offspring classifiers is still in its infancy. Proper mathematical approaches to the parameter estimations are necessary to understand possible disruption and ensure fast detection of more accurate (or lower error) classifiers. Additionally, the MAM technique might be questioned because initial, large updates may be highly disruptive. However, parameter initialization becomes even more crucial once pure Widrow-Hoff updates are applied (since then an incorrect initial value can cause strong disruption). More resent modifications of XCS showed that the prediction part in XCS is generally very flexible enabling the estimation of linear and polynomial predictions approximated with recursive least squares or the pseudo inverse [17,18].

The results herein show that in strongly overlapping problems, the fitness sharing approach may be reconsidered or may actually be obsolete. Future analyses on this matter will be relevant not only for the XCS classifier system but also for LCSs in general since all (Michigan-style) classifier systems rely on iterative parameter updates and thus noisy fitness estimates.

Acknowledgments

We are grateful to the whole IlliGAL lab for their help and the useful discussions.

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129). 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 the Computational Science and Engineering graduate option program (CSE) at the University of Illinois at Urbana-Champaign, the German research foundation (DFG) under grant DFG HO1301/4-3 as well as 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. Alwyn Barry. A hierarchical XCS for long path environments. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 913–920, 2001.
2. Ester Bernadó, Xavier Llorà, and Joseph M. Garrell. XCS and GALE: A comparative study of two learning classifier systems and six other learning algorithms on classification tasks. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems (LNAI 2321)*, pages 115–132. Springer-Verlag, Berlin Heidelberg, 2002.
3. Larry Bull and Jacob Hurst. ZCS redux. *Evolutionary Computation*, 10(2):185–205, 2002.
4. 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, 2005.
5. Martin V. Butz, David E. Goldberg, Pier Luca Lanzi, and Kumara Sastry. Bounding the population size to ensure niche support in XCS. IlliGAL report 2004033, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2004.
6. Martin V. Butz, David E. Goldberg, and K. Tharakunnel. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11:239–277, 2003.
7. Martin V. Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson. How XCS evolves accurate classifiers. *Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 927–934, 2001.
8. Martin V. Butz, M. Pelikan, X. Llorà, and D.E. Goldberg. Extracted global structure makes local building block processing effective in XCS. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 1*, pages 655–662, 2005.
9. Martin V. Butz, Kumara Sastry, and David E. Goldberg. Tournament selection in XCS. *Proceedings of the Fifth Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1857–1869, 2003.
10. Martin V. Butz and Stewart W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Third international workshop, IWLCS 2000 (LNAI 1996)*, pages 253–272. Springer-Verlag, Berlin Heidelberg, 2001.
11. Phillip W. Dixon, David W. Corne, and Martin J. Oates. A preliminary investigation of modified XCS as a generic data mining tool. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in learning classifier systems: Fourth international workshop, IWLCS 2001 (LNAI 2321)*, pages 133–150. Springer-Verlag, Berlin Heidelberg, 2002.

12. Tim Kovacs. Deletion schemes for classifier systems. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 329–336, 1999.
13. Tim Kovacs. Strength or Accuracy? Fitness calculation in learning classifier systems. In Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, editors, *Learning classifier systems: From foundations to applications (LNAI 1813)*, pages 143–160. Springer-Verlag, Berlin Heidelberg, 2000.
14. Tim Kovacs. Towards a theory of strong overgeneral classifiers. *Foundations of Genetic Algorithms 6*, pages 165–184, 2001.
15. Pier Luca Lanzi. An analysis of generalization in the XCS classifier system. *Evolutionary Computation*, 7(2):125–149, 1999.
16. Pier Luca Lanzi. An extension to the XCS classifier system for stochastic environments. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 353–360, 1999.
17. Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, and David E. Goldberg. Extending XCSF beyond linear approximation. *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume 2*, pages 1827–1834, 2005.
18. Pier Luca Lanzi, Daniele Loiacono, Stewart W. Wilson, and David E. Goldberg. Generalization in XCSF for real inputs. IlliGAL report 2005023, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2005.
19. Gilles Venturini. Adaptation in dynamic environments through a minimal probability of exploration. *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 371–381, 1994.
20. Bernard Widrow and Samuel D. Stearns. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
21. Stewart W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2:1–18, 1994.
22. Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
23. Stewart W. Wilson. Generalization in the XCS classifier system. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674, 1998.