Can Evolution Strategies Improve Learning Guidance in XCS? Design and Comparison with Genetic Algorithms based XCS

Sergio MORALES-ORTIGOSA, Albert ORRIOLS-PUIG, and
Ester BERNADÓ-MANSILLA

Grup de Recerca en Sistemes Intel·ligents
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull
Quatre Camins 2, 08022, Barcelona (Spain)

Abstract. XCS is a complex machine learning technique that combines credit apportionment techniques for rule evaluation with genetic algorithms for rule discovery to evolve a distributed set of sub-solutions online. Recent research on XCS has mainly focused on achieving a better understanding of the reinforcement component, yielding several improvements to the architecture. Nonetheless, studies on the rule discovery component of the system are scarce. In this paper, we experimentally study the discovery component of XCS, which is guided by a steady-state genetic algorithm. We design a new procedure based on evolution strategies and adapt it to the system. Then, we compare in detail XCS with both genetic algorithms and evolution strategies on a large collection of real-life problems, analyzing in detail the interaction of the different genetic operators and their contribution in the search for better rules. The overall analysis shows the competitiveness of the new XCS based on evolution strategies and increases our understanding of the behavior of the different genetic operators in XCS.

Keywords. Evolutionary Algorithms, Genetic Algorithms, Evolution Strategies, Learning Classifier Systems, Supervised Learning, Data Mining.

Introduction

Learning Classifier Systems (LCSs) [8] are machine learning techniques that learn *rule sets* on-line through the interaction with an *environment* that represents a *stream of labeled examples*. In the recent years, XCS [13], the most influential LCS, has arisen as a promising technique for classification tasks and data mining, showing its competitiveness with respect to highly-used machine learning techniques such as the decision tree C4.5 and support vector machines [10]. XCS consists of a complex architecture that evolves a rule set by means of the interaction of two main components: (i) a rule evaluation system and (ii) a rule discovery procedure. The rule evaluation system, based on credit apportionment techniques, is responsible for evaluating the quality of the rules on-line with the information provided by the environment. This component has received an increasing amount of attention during the last few years, resulting in several improvements that enabled the system to solve problems that previously eluded solution [3]. The rule discovery procedure, driven by a genetic algorithm (GA), is responsible for providing new promising rules to the system. Whereas the evaluation component has been studied in detail, the discovery component has received little attention, remaining practically unchanged from its initial conception.

In this paper, we aim at analyzing the behavior of the discovery component in XCS. For this purpose, we design a new discovery procedure based on evolution strategies (ESs) [12] to drive the discovery of new rules in the on-line learning architecture. We modify the interval-based rule representation of XCS [14] by introducing a vector of strategy parameters and adapt the selection, recombination, and mutation operators of ESs to let them deal with interval-based rules. Both original XCS and XCS based on evolution strategies are compared on a collection of real-life problems. The interaction of different genetic operators, i.e., the combination of selection and mutation and the combination of selection, crossover, and mutation, is carefully studied for each one of the two systems. This study not only shows the performance improvements due to the new discovery component, but also enables us to increase our comprehension of how the evolutionary genetic process works.

The remainder of this paper is structured as follows. Section 1 gives a brief description of XCS. Section 2 presents the new ES-based discovery component introduced to XCS, detailing the modifications introduced to XCS representation and to the different genetic operators. Section 3 provides the methodology followed in the experimentation and Section 4 analyzes the obtained results, especially highlighting the interaction of different genetic operators. Finally, Section 5 summarizes, concludes, and presents some future work lines that will be followed in light of the results presented in the current study.

1. XCS in a Nutshell

As follows, we provide a brief description of XCS focused on classification tasks. For an algorithmic description the reader is referred to [5].

1.1. Knowledge Representation

XCS evolves a population [P] of classifiers. Each classifier consists of a production rule and a set of parameters. The production rule takes the following form: **if** $x_1 \in [l_1, u_1] \wedge \ldots x_\ell \in [l_\ell, u_\ell]$ **then** *class*. That is, each variable of the condition x is represented by an interval $[\ell_i, u_i]^\ell$ (ℓ is the number of input attributes of the problem). Then, a rule matches an input instance $e = (e_1, e_2, \ldots, e_\ell)$ if $\forall_i \ell_i \leq e_i \leq u_i$.

Each classifier has three main parameters: (i) the payoff prediction p, an estimate of the reward that the system will receive if the class of the rule is selected as output, (ii) the prediction error ϵ, which estimates the error of the payoff prediction, and (iii) the fitness F, which is computed as and inverse function of the prediction error.

1.2. Process Organization

XCS learns the data model on-line by means of the interaction with an environment which represents a stream of examples. That is, at each learning iteration, XCS receives a new training example $e = (e_1, e_2, \ldots, e_\ell)$ and the system creates a *match set* [M], which consist of all the classifiers in [P] whose condition matches e. If any of the classes is not represented in [M], the covering operator is triggered, creating a new classifier that predicts the missing class and whose condition is generalized from the input as $\forall_i^\ell : l_i = e_i - rand(0, r_0)$ and $u_i = e_i + rand(0, r_0)$, where r_0 $(0 < r_0 < 1)$ is a configuration parameter that determines the initial generalization degree. The next step depends on whether the system is in exploration (training) mode or in exploitation (test) mode. In exploration mode, the system randomly chooses one of the possible classes and builds the action set [A] with all the classifiers in [M] that advocate the selected class. The action set represents the niche of similar classifiers where both the parameter update procedure and the genetic algorithm take place. The parameters of all the classifiers in [A] are updated according to a generalized version of Q-learning (see [13]). The genetic algorithm is applied as explained in the next subsection. In exploitation mode, the classifiers in [M] vote, according to their fitness, for the class they predict. The most voted class is selected as output.

1.3. Discovery Component

XCS applies a steady-state niche genetic algorithm (GA) [7] to discover new promising rules. The GA is triggered on [A] when the average time since its last application to the classifiers in [A] exceeds a certain threshold θ_{GA}. Then, the system selects two parents from [A]. So far, two selection schemes have been studied: *proportionate selection* [13], in which each classifier in [A] has a probability proportional to its fitness to be chosen; and *tournament selection* [4], in which tournaments are held among a set of randomly selected classifiers and the best classifier of the tournament is chosen as a parent. Then, the parents are crossed and mutated with probabilities χ and μ respectively. Crossover shuffles the condition of the two parents by cutting the chromosomes by two points. Mutation decides whether each variable has to be changed; in this case, it adds a random amount to the lower or to the upper bound of the variable interval. The resulting offspring are introduced into the population, removing potentially poor classifiers if there is not room for them [9].

2. Introducing Evolution Strategies to XCS: Representation and New Operators

Evolution Strategies [12], like GAs, are optimization algorithms that take inspiration from biology to solve complex optimization problems. The main differences between GAs and ESs is that ESs incorporate a vector of strategy parameters that are used by the mutation operator to guide the local search towards the objective. Therefore, mutation is the primary operator of ESs. The vector of strategy parameters is self-adapted during the evolutionary process with the aim of applying more precise mutations to the classifier's conditions. As follows, we explain the new representation, as well as the new genetic operators, proposed to adapt ESs to the interval-based rule representation of XCS.

2.1. Knowledge Representation for Evolution Strategies

Now, the classifier representation is enriched with a vector of strategy parameters $s = (\sigma_1, \sigma_2, \ldots, \sigma_\ell)$, which is used to adapt the intervals of the variables of the rules condition (the rules condition is referred to as *object parameters* in ESs terms). Covering initializes each strategy parameter σ_i as $\sigma_i = rand(0, \mu)$, where μ is initially chosen in the range $[1/N, 1/\ell]$ (N is the population size and ℓ is the number of attributes of the problem). The strategy parameters undergo recombination and mutation during learning; therefore, they self-adapt with the aim of letting mutation perform a more precise local search. All the genetic operators, i.e., selection, mutation, and recombination, are redefined as explained in the following sections.

2.2. Selection

In evolution strategies, the typical selection operator is *truncation selection*, which selects the classifiers of [A] with highest fitness. Since this selection strategy can be quite aggressive, especially in steady state algorithms, we also use proportionate and tournament selection as defined in the previous section.

2.3. Gaussian Mutation

The mutation operator of XCS is redefined as follows. First, we mutate the intervals of each rule variable x_i as $x_i = x_i + z$, where $z = (\sigma_1 N_1(0,1), \sigma_2 N_2(0,1), \ldots, \sigma_\ell N_\ell(0,1))$ are independent random samples from Gaussian normal distribution.

The strategy parameters are self-adapted along the XCS run. After mutation, the new vector of strategy parameters s' is updated as $s' = e^{\tau_0 N_0(0,1)} \left(\sigma_1 e^{\tau N_1(0,1), \ldots, \tau N_\ell(0,1)}\right)$, where τ indicates the precision of self-adaption, τ_0 weights the global effect of mutation, and $N_i(0,1)$ returns a Gaussian number with $\sigma = 1$. In our experiments, we configured $\tau_0 = 1/\sqrt{2\ell}$ and $\tau = 1/\sqrt{2\sqrt{\ell}}$ as usually done in the ESs literature.

2.4. Recombination

We consider two classes of recombination in ESs: (i) discrete/dominant recombination and (ii) intermediate recombination. Discrete recombination produces a new rule where each variable and strategy parameter is randomly selected from one of the parents. Intermediate recombination calculates the center of mass of the parents; thus, this recombination operator pushes towards the average value per attribute among all classifiers.

3. Experimental Methodology

The main concern of this work is to analyze in detail the effect of the different genetic operators on the genetic search as well as the interaction among themselves. Moreover, the advantages and disadvantages of the new operators will be carefully studied. For this purpose, we start examining XCS's behavior with selection and mutation with the aim of analyzing the search capabilities provided by the combination of the two operators. Then, we add crossover to the study, showing the benefits supplied by this operator. Moreover,

Table 1. Properties of the data sets. The columns describe: the identifier of the data set (Id.) the name of the data set (dataset), the number of instances (#Inst), the total number of features (#Fea), the number of real features (#Re), the number of integer features (#In), the number of nominal features (#No), the number of classes (#Cl), and the proportion of instances with missing values (%MisInst).

Id.	**dataset**	**#Inst**	**#Fea**	**#Re**	**#In**	**#No**	**#Cl**	**%MisInst**
bal	Balance	625	4	4	0	0	3	0
bpa	Bupa	345	6	6	0	0	2	0
gls	Glass	214	9	9	0	0	6	0
h-s	Heart-s	270	13	13	0	0	2	0
irs	Iris	150	4	4	0	0	3	0
pim	Pima	768	8	8	0	0	2	0
tao	Tao	1888	2	2	0	0	2	0
thy	Thyroid	215	5	5	0	0	3	0
veh	Vehicle	846	18	18	0	0	4	0
wbcd	Wisc. breast-cancer	699	9	0	9	0	2	2.3
wdbc	Wisc. diagnose breast-cancer	569	30	30	0	0	2	0
wne	Wine	178	13	13	0	0	3	0

in all the cases, the results of GA-based XCS (referred to as XCS$_{GA}$) are compared with those obtained with ES-based XCS (referred to as XCS$_{ES}$), providing some interesting insights on the differences between evolution strategies and genetic algorithms in the context of on-line learning.

For the study, we used a collection of 12 real-life data sets extracted from the UCI repository [1], whose characteristics are summarized in Table 1. The different configurations of XCS$_{GA}$ and XCS$_{ES}$ were ran on these data sets and the quality of the results was compared in terms of the performance (test accuracy) of the final models. To obtain reliable estimates of these metrics we used a ten-fold cross-validation procedure. XCS was configured as follows (see [5] for notation details): $iter.$ = 100,000, N = 6400, θ_{GA} = 50, χ = 0.8, μ = 0.04, r_0 = 0.6, m_0 = 0.1.

We statistically analyzed the performance of each learner following the procedure pointed out in [6]. We first applied the multi-comparison Friedman test to contrast the null hypothesis that all the learning algorithms performed the same on average. If the Friedman test rejected the null hypothesis, the post-hoc Bonferroni-Dunn test was used. Moreover, when required, we also applied pairwise comparisons by means of the non-parametric Wilcoxon signed-ranks test.

4. Experimental Results

As follows, we present the results obtained with the combinations of (i) selection and mutation operators and (ii) selection, crossover, and mutation operators.

4.1. Analysis of the Effect of Selection + Mutation

Our first concern was to analyze the behavior of XCS with both GAs and ESs when only the selection operator and the mutation operator were considered. For this purpose, Table 2 supplies the test accuracies and the ranks obtained by XCS$_{GA}$ and XCS$_{ES}$ with

Table 2. Comparison of the test performance and rank obtained with XCS$_{GA}$ and XCS$_{ES}$ with proportionate selection (ps) and tournament selection (ts). Moreover, the results of XCS$_{ES}$ with truncation selection (tr) and weighted XCS$_{ES}$ are also provided. In all the runs, crossover was switched off. The last three rows provide the average performance, the average rank, and the position of each learner in the ranking.

Dataset	**XCS$_{GA}$-ps**	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**	**weig. XCS$_{ES}$**
bal	82.35 (2)	82.08 (3)	81.55 (5)	81.71 (4)	81.12 (6)	82.93 (1)
bpa	62.51 (5.5)	64.15 (2)	62.80 (3)	65.12 (1)	62.70 (4)	62.51 (5.5)
gls	66.51 (6)	67.13 (4)	66.98 (5)	69.94 (1)	67.29 (3)	67.91 (2)
h-s	41.23 (4)	43.46 (1)	41.60 (3)	42.72 (2)	37.78 (6)	39.63 (5)
irs	94.89 (3)	93.33 (6)	95.33 (1)	94.89 (3)	94.89 (3)	94.44 (5)
pim	70.83 (4)	71.05 (2)	69.99 (6)	72.87 (1)	70.88 (3)	70.53 (5)
tao	89.32 (6)	92.90 (3)	89.79 (5)	93.80 (1)	93.01 (2)	89.90 (4)
thy	94.73 (6)	95.50 (4)	95.66 (2.5)	96.28 (1)	94.88 (5)	95.66 (2.5)
veh	65.52 (3)	66.00 (2)	64.50 (4)	67.26 (1)	63.83 (6)	64.34 (5)
wbcd	80.88 (6)	85.84 (1)	81.26 (5)	85.65 (2)	82.50 (4)	82.93 (3)
wdbc	78.68 (2)	75.28 (3)	80.20 (1)	74.93 (4)	67.60 (6)	69.13 (5)
wne	80.71 (5)	86.70 (1)	82.21 (2)	82.02 (3)	78.09 (6)	81.65 (4)
avg.	75.68	76.95	75.99	77.26	74.55	75.13
rnk.	4.38	2.67	3.54	2.00	4.50	3.92
pos.	**5**	**2**	**3**	**1**	**6**	**4**

proportionate and tournament selection (see from the 2nd to the 5th column). Moreover, we also included truncation selection for XCS_{ES}, since it is a selection operator widely used in the ESs field (6th column). The average rank of each learner shows that two schemes based on ESs are the best ranked methods in the comparison. The Friedman test permitted to reject the hypothesis that all the learners were statistically equivalent at $\alpha = 0.01$. The Bonferroni-Dunn test, at $\alpha = 0.05$, indicated that XCS$_{ES}$ with tournament selection significantly outperformed XCS$_{GA}$ with both proportionate and tournament selection. Besides, XCS$_{ES}$ with proportionate selection was significantly better than XCS$_{G}A$ proportionate selection.

Three important observations can be drawn from these results. Firstly, the XCS_{ES} based on truncation selection resulted in the poorest performance of the comparison. We hypothesize that this behavior is because truncation selection is an excessively elitist operator that makes a strong pressure toward the fittest individuals, which goes in detriment of the population diversity. Secondly, the schemes based on tournament selection yielded better results than those schemes based on proportionate selection for XCS$_{GA}$ and XCS$_{ES}$. These results show the superiority of tournament selection with respect proportionate selection, confirming the theoretical studies presented in [11] and [4]. Therefore, our analysis enables us to extend these conclusions to real-life problems. Thirdly, XCS$_{ES}$ presents brilliant results in the tao, $wbcd$, and wne data sets, significantly outperforming the results obtained by XCS_{GA} according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. To our knowledge, XCS_{ES} obtained, by far, the best ever reported performance in the tao data set, a problem which is especially complicated for XCS since the hyper rectangular representation can barely approximate the decision boundaries of the problem accurately [2].

The overall results indicated that the mutation introduced by XCS$_{ES}$, i.e., Gaussian mutation, has a greater freedom of action due to a more disruptive behavior introduced

Table 3. Comparison of the test performance and rank obtained with XCS$_{GA}$ and XCS$_{ES}$ with proportionate selection (ps) and tournament selection (ts). Moreover, the results of XCS$_{ES}$ with truncation selection (tr) and XCS$_{ES}$ are also provided. In all runs, we applied selection, crossover, and mutation. The last three rows provide the average performance, the average rank, and the position of each learner in the ranking.

Dataset	**XCS$_{GA}$-ps**	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**
bal	83.20 (1)	82.77 (2.5)	82.72 (4)	82.77 (2.5)	82.13 (5)
bpa	68.21 (1)	67.05 (2)	65.22 (4)	65.70 (3)	64.06 (5)
gls	72.12 (2)	71.18 (4)	73.21 (1)	71.65 (3)	69.00 (5)
h-s	46.91 (4)	51.23 (1)	47.04 (3)	49.13 (2)	44.32 (5)
irs	95.33 (1.5)	95.33 (1.5)	94.89 (4.5)	95.11 (3)	94.89 (4.5)
pim	72.53 (5)	74.43 (2)	73.39 (4)	73.83 (3)	74.74 (1)
tao	91.22 (4)	93.52 (3)	91.19 (5)	94.35 (1)	94.17 (2)
thy	95.81 (2)	95.50 (5)	96.43 (1)	95.66 (3.5)	95.66 (3.5)
veh	71.79 (2)	71.75 (3)	71.20 (4.5)	72.89 (1)	71.20 (4.5)
wbcd	94.85 (3)	95.47 (1.5)	93.51 (4)	95.47 (1.5)	92.61 (5)
wdbc	91.09 (4)	91.80 (3)	92.44 (2)	92.85 (1)	89.51 (5)
wne	95.50 (4)	96.25 (1.5)	95.69 (3)	96.25 (1.5)	91.38 (5)
avg.	81.55	82.19	81.41	82.14	80.31
rnk.	2.79	2.5	3.33	2.17	4.21
pos.	**4**	**2**	**3**	**1**	**5**

by the random Gaussian variables. This enables Gaussian mutation to assume tasks of innovation, which has been typically performed by the recombination operator in the GA realm. In order to confirm the hypothesis, we designed a new Gaussian mutation operator, which we addressed as weighed Gaussian mutation. This new operator normalized 95% of the values of Gaussian distribution with the aim of softening the behavior presented by Gaussian mutation. The results obtained with the new operator are shown in the last column of Table 2. The results are similar to those showed by XCS_{GA}, which confirms that non-weighed Gaussian mutation has more freedom of action than random mutation, aspect that promotes the global search capabilities of the system.

4.2. Analysis of the Effect of Selection + Crossover + Mutation

After evaluating the behavior of XCS$_{ES}$ with Gaussian mutation with respect to XCS$_{GA}$, we now compare the systems with the complete genetic cycle. That is, we ran the same experiments, but adding the crossover operator to each scheme. More specifically, we used two point crossover for XCS$_{GA}$ and a combination of discrete recombination for object parameters and intermediate recombination for strategy parameters for XCS$_{ES}$. We considered these two crossover methods for each configuration since they were the schemes that maximized the average rank in each case.

Table 3 shows the test accuracy of the different configurations of XCS on the same collection of real-life problems. Several observations can be drawn from the results. Firstly, it is worth noting that the inclusion of crossover improves the test accuracy achieved by XCS in most of the data sets not only for XCS$_{GA}$, but also XCS$_{ES}$. Therefore, although theoretical studies that show the benefits of crossover in XCS are lacking, these results support its use to solve complex classification real-life problems. Secondly, as in the previous section, the XCS's schemes based on ESs are the best ranked in the comparison. The multi-comparison test rejected the null hypothesis that all learners performed the same on average at $\alpha = 0.01$. The post-hoc Bonferroni-Dunn test, at $\alpha = 0.05$, identified that XCS_{ES} with proportionate and tournament selection outperformed XCS_{ES} with truncation selection. No further significant differences were detected. That is to say, differently from the comparison in the previous section, the results obtained by the XCS schemes based on ESs were not significantly better than those based on GAs. This confirms our previous hypothesis that the mutation operator introduced by the ESs permitted a more guided search toward highly fit classifiers in our population, which can be simulated in the GA realm by the use of crossover. Notwithstanding, notice that XCS$_{ES}$ continues being the best ranked learner on average. Finally, let us highlight the brilliant performance presented by XCS$_{ES}$ with any configuration in the tao data set, which is even higher than the one obtained in the previous section and than the accuracy reached by XCS$_{GA}$. In this case, the guidance provided by the evolution strategy seems to be crucial to learn the decision boundaries accurately.

5. Summary, Conclusions, and Further Work

In this paper, we introduced an evolution strategy as a mechanism to discover new promising rules in XCS. We extended the XCS's representation by introducing new parameters required by the evolution strategy and compared XCS with genetic algorithms and XCS with evolution strategies in detail. The effect of the different operators was analyzed carefully. The experimental results evidenced that XCS with evolution strategies presented the best results on average. Moreover, different insights on the role of the different operators were provided.

The overall study presented in this paper served to increase our understanding of how genetic operators work in XCS, especially highlighting the role of crossover and mutation. In addition to all the notes provided while discussing the results, the experimentation highlighted two crucial aspects that should be addressed as further work. Firstly, the results clearly showed that XCS could benefit from new genetic operators. Therefore, more research must be conducted on this regard, designing new operators that consider more information that is available during the genetic evolution. Secondly, results also indicated that different problems benefited from different genetic operators. That is, the performance in problems such as tao was impressively increased by the use of an ES. Nonetheless, XCS$_{GA}$ obtained slightly better results than XCS$_{ES}$ in other problems. Therefore, as further work, we will study different strategies to extract characteristics from the training data sets and link these characteristics to the type of operators used during search with the aim of designing hyper-heuristics that enable the system to self-tune its operators depending on the apparent complexity of each particular problem.

Acknowledgements

The authors would like to thank *Enginyeria i Arquitectura La Salle*, Universitat Ramon Llull, as well as the *Ministerio de Educación y Ciencia* for its support under project TIN2005-08386-C05-04 and *Generalitat de Catalunya* for its support under grants 2005FI-00252 and 2005SGR-00302.

References

[1] A. Asuncion and D. J. Newman. *UCI Machine Learning Repository: [http://www.ics.uci.edu/∼mlearn/MLRepository.html]*. University of California, 2007.

[2] E. Bernadó-Mansilla and T.K. Ho. Domain of competence of XCS classifier system in complexity measurement space. *IEEE Transactions on Evolutionary Computation*, 9(1):1–23, 2005.

[3] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems. *IEEE Transactions on Evolutionary Computation*, 9(5):452–473, 2005.

[4] 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(1):53–77, 2005.

[5] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.

[6] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[7] D. E. Goldberg. *Genetic algorithms in search, optimization & machine learning*. Addison Wesley, 1 edition, 1989.

[8] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. New York: Academic Press, 1976.

[9] T. Kovacs. Deletion schemes for classifier systems. In *GECCO'99: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 329–336. Morgan Kaufmann, 1999.

[10] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced datasets. *Soft Computing Journal*, 2008.

[11] A. Orriols-Puig, K. Sastry, P.L. Lanzi, D.E. Goldberg, and E. Bernadó-Mansilla. Modeling selection pressure in XCS for proportionate and tournament selection. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, volume 2, pages 1846–1853. ACM Press, 2007.

[12] I. Rechenberg. *Evolution Strategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, 1973.

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

[14] S. W. Wilson. Get real! XCS with continuous-valued inputs. In *Learning Classifier Systems. From Foundations to Applications*, LNAI, pages 209–219, Berlin, 2000. Springer-Verlag.