Analysis and Improvement of the Genetic Discovery Component of 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)

email: {is09767,aorriols,esterb}@salle.url.edu

Abstract

XCS is a learning classifier system that uses genetic algorithms to evolve a population of classifiers online. When applied to classification problems described by continuous attributes, XCS has demonstrated to be able to evolve classification models—represented as a set of independent interval-based rules—that are, at least, as accurate as those created by some of the most competitive machine learning techniques such as C4.5. Despite these successful results, analyses of how the different genetic operators affect the rule evolution for the interval-based rule representation are lacking. This paper focuses on this issue and conducts a systematic experimental analysis of the effect of the different genetic operators. The observations and conclusions drawn from the analysis are used as a tool for designing new operators that enable the system to extract models that are more accurate than those obtained by the original XCS scheme. More specifically, the system is provided with a new discovery component based on evolution strategies, and a new crossover operator is designed for both the original discovery component and the new one based on evolution strategies. In all these cases, the behavior of the new operators are carefully analyzed and compared with the ones provided by original XCS. The overall analysis enables us to supply important insights into the behavior of different operators and to improve the learning of interval-based rules in real-world domains on average.

1 Introduction

Learning classifier systems (LCSs) [17] are machine learning techniques that combine *genetic algorithms* (GAs) [13, 16] with *apportionment of credit techniques* to learn *rule sets* online through the interaction with an *environment* that represents a *stream of examples*. In the recent years, XCS [28], 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 [20, 21]. When applied to real-world classification problems that are described by continuous attributes, XCS evolves a population of interval-based classification rules [29] by means of the interaction of two main components: (1) a rule evaluation system and (2) a rule discovery procedure. The rule evaluation system, based on reinforcement learning techniques, is responsible for evaluating the quality of the rules online 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 [6]. The rule discovery procedure, driven by a GA, is responsible for providing the system with new promising rules. For the interval-based representation, XCS uses simple crossover and mutation operators that were copied from the ternary-rule representation. Thence, whereas the evaluation component has been studied in detail, the discovery component for the interval-based representation has received little attention, remaining practically unchanged from its initial conception [29].

The purpose of this paper is to experimentally analyze the role of the different genetic operators and to use the observations extracted from this analysis to improve the existing operators to deal with interval-based rules. To achieve this, we first focus on analyzing the role of the mutation operator and design a new discovery procedure based on *evolution strategies* (ESs) [24], since this type of search procedures were specifically ideated to deal with continuous attributes such as those used by the interval-based representation. Both original XCS and XCS based on ESs are systematically compared on a collection of real-world problems, emphasizing the benefits provided by the mutation operator incorporated by the new ES-based discovery component. Subsequently, the role of the crossover operator is studied. We propose a new crossover scheme, addressed as BLX crossover, which is inspired by the BLX-α crossover operator of real-coded GAs [14], but adapted to deal with the interval-based representation of XCS. The performance obtained when using the new crossover operator in XCS with both ESs and GAs is tested on a large collection of real-world problems. The overall study enables us to (1) increment our understanding of the role of different genetic operators, as well as their limitations, and (2) improve the discovery procedure of the original scheme of XCS, being able to evolve models that are, on average, more accurate.

The remainder of this paper is organized as follows. Section 2 briefly describes XCS, and Sect. 3 explains the new discovery component based on ESs. Section 4 first analyzes the role of the mutation operator alone in both GA-based and ES-based XCS, and later, introduces the crossover operator in the analysis. Section 5 starts with the improvement of the crossover operator, describing a new enhanced operator that tries to capture some of the ideas articulated in the analysis of GA-based XCS and ES-based XCS. The benefits of this new operator are experimentally analyzed in Sect. 6. Finally, Section 7 summarizes, concludes, and discusses future work lines.

2 The XCS Classifier System

XCS [28] is the most influential Michigan-style LCS. The system combines reinforcement learning techniques [25] with GAs [13, 16] to learn a distributed set of sub-solutions online. As follows, we briefly describe the knowledge representation used by XCS in domains with continuous variables and then explain the process organization followed by the system to evolve a classification model. Lastly, we exemplify the type of classification models built by XCS and discuss the possible difficulties that the discovery component of XCS may need to face to evolve an accurate classification model in complex domains. For further details of the system, the reader is referred to [8, 28, 29].

2.1 Knowledge Representation

XCS evolves a population [P] of *classifiers* that consist of a *production rule* and a set of *parameters*. In domains with continuous attributes, production rules most often take the following form [29]:

$$\textbf{if } x_1 \in [\ell_1, u_1] \wedge x_2 \in [\ell_2, u_2] \wedge \ldots \wedge x_\ell \in [\ell_\ell, u_\ell] \textbf{ then } c, \tag{1}$$

where the antecedent of the rule contains ℓ input variables that are represented by an interval of possible values $[\ell_i, u_i]^\ell$, and the consequent denotes the predicted class. Note that the condition of the rule defines a hyper rectangle in the solution space. Therefore, the population represents a set of hyper rectangles that together should cover all the solution space. Then, a rule matches an input instance $e = (e_1, e_2, \ldots, e_\ell)$ if $\forall_i\ \ell_i \le e_i \le u_i$.

Each classifier has six main parameters: (1) the payoff prediction p, an estimate of the reward that the system will receive if the class of the rule is selected as output, (2) the prediction error ϵ, which estimates the error of the payoff prediction, (3) the fitness F, which is computed as an inverse function of the prediction error, (4) the action set size as, an estimate of the average size of the action sets in which the classifier has participated, (5) the experience exp, which reckons the number of examples that the classifier has matched during its life, and (6) the numerosity n, the number of copies of the classifier in the population.

To fully understand the learning process of XCS, the next three subsections explain how the different components of the system interact to evaluate the existing classifiers and how the GA is applied to evolve new classifiers.

2.2 Learning Interaction

XCS learns online by interacting with an environment which provides a new training example at each iteration. 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 consists of all the classifiers in [P] whose condition matches e. The next step depends on whether the system is in exploitation (test) mode or in exploration (training) mode. 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. 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 parameters of all the classifiers in [A] are updated according to a generalized version of Q-learning, which is explicated in the following subsection.

2.3 Parameter Update

Once [A] is formed, the selected class is sent to the environment, which returns a reward R that is used by XCS to update the parameters of the classifiers in [A]. First, the prediction p is adjusted as $p = p + \beta(R - p)$, where β is the learning rate ($0 < \beta \leq 1$). Next, the error ϵ is updated as $\epsilon = \epsilon + \beta(|R - p| - \epsilon)$. To update the fitness of the classifiers, XCS first computes the *accuracy* κ of each classifier as follows:

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon < \varepsilon_0, \\ \alpha(\varepsilon/\varepsilon_0)^{-\nu} & \text{otherwise;} \end{cases} \tag{2}$$

where ϵ_0 is the maximum error that a classifier can have to be considered maximally accurate, and α is a discount factor. κ is used to compute the *relative accuracy* κ' of the classifier in [A] as $\kappa' = \kappa / \sum_{[A]} \kappa_i$. Finally, fitness is updated from the relative accuracy as $F = F + \beta(\kappa' - F)$. Note that fitness is shared among the classifiers in the same action set since it is calculated from the relative accuracies.

2.4 Discovery Component

XCS applies a steady-state niche-based GA 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* [28], in which each classifier has a probability proportional to its fitness to be chosen, and *tournament selection* [7], in which tournaments are held among a set of randomly selected classifiers, and the best classifier of the tournament is chosen as a parent.

Next, the parents are crossed and mutated with probabilities χ and μ respectively. For the interval-based representation, the crossover and the mutation operators work as follows [29]. Crossover shuffles the condition of the two parents by cutting the chromosomes by two points. Figure 1 illustrates an example of crossover of two classifiers that are described by two attributes. Mutation decides whether each variable has to be changed; in this case, it adds a random amount, ranging in $[0, m_0]$ to the lower or to the upper bound of the variable interval. An example of mutation is illustrated in Fig. 2.

Finally, each offspring is introduced into the population, removing a classifier with low fitness if the population is full. The deletion probability of a classifier is proportional to the size of the action sets where the classifier has participated and inversely proportional to its fitness [19]. This biases the search toward highly fit classifiers and, at the same time, balances the classifier allocation in the different action sets.

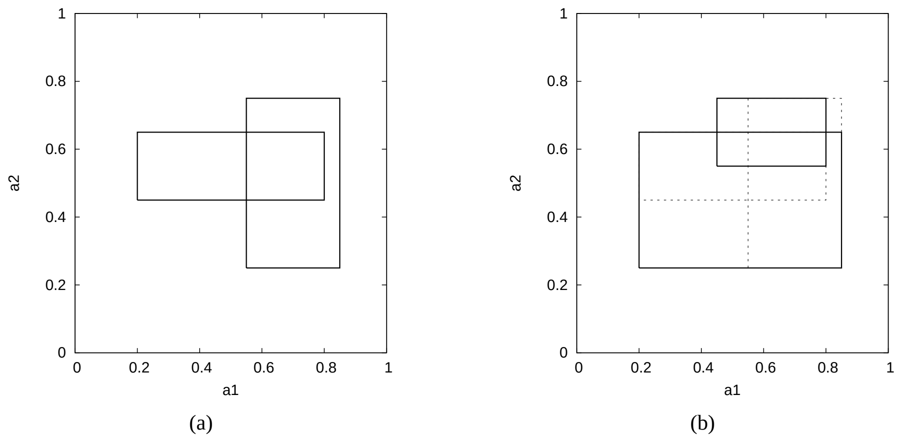

Figure 1: Example of crossover: (a) plots the two parents and (b) shows the offspring resulting from two cut points occurring in the middle of each interval.

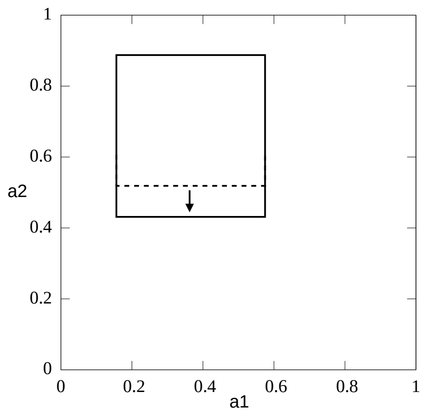

Figure 2: Example of mutation: a random value is added to the lower bound of attribute $a2$.

2.5 How Do Models Evolved by XCS Look Like?

Thus far, we have explained the knowledge representation, which uses interval-based rules, and the process employed to evolve a population of maximally general and accurate rules. In this section, we use the *tao* problem [4] to illustrate how these classification models look like and to intuitively discuss the possible limitations of the genetic operators utilized by XCS.

The *tao* problem is a two-class classification problem described by two attributes. Figure 3(a) depicts the learning examples of the problem, and Fig. 3(b) shows the rules evolved by a single run of XCS on this problem. The hyper rectangles in dashed

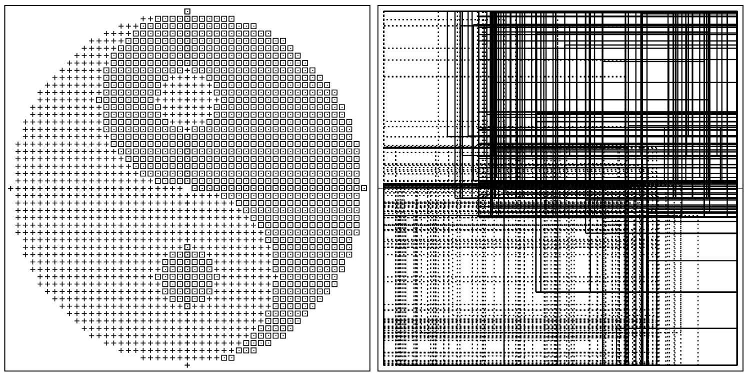
(a) Domain (b) Rule set

Figure 3: Example of (b) the rules evolved by XCS on (a) the *tao* problem.

lines represent the condition of the rules in the final population that predict the first class, while the hyper rectangles in continuous lines represent the rules that predict the second class. Note that the system evolves a large number of overlapping rules to fit the curved boundaries of the *tao* problem. In particular, it is worth noticing that, near the class boundary, there are a lot of overlapping classifiers that predict different classes. These regions of the feature concentrate almost all the error of the classification models evolved by XCS.

Ideally, we would like the discovery component of XCS to incrementally adjust classifiers that are close to the class boundary, resulting in offspring with zero error. However, the genetic operators implemented in the original scheme of XCS may not have the flexibility to provide this adjustment. That is, the mutation operator selects a variable and adds or removes a random number from one of the limits of its interval (see Fig. 2). Therefore, the operator does not consider whether the classifier is near the class boundary or not. On the other hand, 2-point crossover takes two parents and may generate completely different offspring (see Fig. 1(a)). Thus, this operator allows exploring new regions of the feature space, but may be too disruptive when crossing classifiers that lay closely to the class boundary. Consequently, despite the competitive results obtained by XCS when using these two operators, intuition indicates that new operators that consider more information to generate new offspring may be beneficial to the system. With this idea in mind, the next section starts revisiting the mutation operator of XCS by proposing a new discovery component based on ESs.

3 Introducing Evolution Strategies into XCS: Representation and New Operators

ESs [23, 24], like GAs, are optimization algorithms that take inspiration from biology to solve complex optimization problems. One of 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 toward the objective. Therefore, mutation is the primary search 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 population individuals, driving them to the optimal solution.

The purpose of this section is to design an ES-based search component for XCS with the aim of providing a more guided search toward maximally general and accurate classifiers. For this purpose, we extend the representation of the classifiers and design new genetic operators. The following describes the new ES-based search component in more detail.

3.1 Knowledge Representation for Evolution Strategies

Now, the classifier representation is extended with a vector of strategy parameters $s = (\sigma_1, \sigma_2, \ldots, \sigma_\ell)$, and each strategy parameter σ_i is used to adapt the intervals of the corresponding rule variable. Strategy parameters evolve together with the rule variables (which are referred to as *object parameters* in ESs terms). Thence, the covering and the genetic operators are redefined to let them deal with the new representation. Also, new selection schemes are considered. These modifications are explicated in the next subsection.

3.2 Redefinition of XCS Operators

The covering and genetic operators of XCS with ES-based discovery component are redefined as follows.

Covering

Covering initializes each strategy parameter σ_i as $\sigma_i = rand(0, \mu)$, where μ is initially chosen in the range $[1/N, 1/\ell]$ (where N is the population size, and ℓ is the number of variables of the problem).

Selection

In ESs, the typical selection operator is *truncation selection*, which chooses the individuals that have highest fitness. We consider this strategy in XCS, so, selecting the classifiers with highest fitness in [A]. 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.

Mutation

The mutation operator first mutates the intervals of each rule variable x_i as

$$x_i = x_i + z_i, \tag{3}$$

where $z_i = \sigma_i N_i(0, 1)$, and $N_i(0, 1)$ returns a Gaussian number ranging in [0,1].

The strategy parameters are self-adapted along the XCS run. After mutating the rule condition, the new vector of strategy parameters s' is updated as

$$s' = e^{\tau_0 N_0(0,1)} \left(\sigma_1 e^{\tau N_1(0,1), \ldots, \tau N_\ell(0,1)} \right), \tag{4}$$

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.

Crossover

We consider two different crossover operators for the rule condition and the strategy parameters. For the rule condition, we use discrete recombination. Given two parents, for each variable, this crossover operator randomly selects one of the parents and copies the value of the variable to the first offspring; then, it copies the corresponding variable of the other father to the other offspring. For strategy parameters, we use intermediate recombination. This recombination operator computes each strategy parameter σ_i as the average value of the corresponding parameter in the parent rules.

4 Analysis of XCS$_{GA}$ and XCS$_{ES}$

So far, we have described the original scheme of XCS, which uses a GA as discovery mechanism, and the new proposal of XCS in which the discovery procedure is driven by an ES. In the remainder of the paper, we will address these two versions of XCS as XCS$_{GA}$ and XCS$_{ES}$ respectively. With the description of both XCS$_{GA}$ and XCS$_{ES}$ in mind, we now start a systematic analysis of the effect of mutation and crossover. For this purpose, in what follows, we first present the experimental methodology followed along the study. Then, we proceed to analyze and compare the effect of the different operators of XCS$_{GA}$ and XCS$_{ES}$, as well as the interaction among themselves.

4.1 Experimental Methodology

We first start the experimental study by examining the behavior of XCS with selection and mutation with the aim of analyzing the search capabilities provided by the combination of the two operators in both XCS schemes. Note that the combination of both operators results in a local search around the classifiers of the population. Then, we add crossover to the study, showing the benefits supplied by this operator. Moreover, in all the cases, the results of XCS$_{GA}$ are compared with those obtained by XCS$_{ES}$, providing some interesting insights into the differences between ESs and GAs in the context of online learning.

For the study, we used a collection of 12 real-world data sets whose characteristics are summarized in Table 1. All the data sets were extracted from the UCI repository [1], except for *tao*, which was selected from a local repository [4]. 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.

Table 1: Properties of the data sets. The columns describe: the identifier of the data set (Id.), the name of the data set (data set), the number of instances (#Inst), the total number of features (#Fea), the number of continuous 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.	**data set**	**#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

To obtain reliable estimates of this metric, we used a 10-fold cross-validation procedure [26]. XCS was configured as follows (see [8] for notation details): num_iter = 100 000, N = 6 400, 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 [9]. We first applied the multi-comparison Friedman's test [11, 12] to contrast the null hypothesis that all the learning algorithms performed the same on average. If the Friedman's test rejected the null hypothesis, the post-hoc Bonferroni-Dunn test [10] was used to detect significant differences between a control learner and the remaining methods, and the post-hoc Holm's step-down procedure [15, 18] was employed to identify significant differences between the best ranked method and the other methods of the comparison. Moreover, when required, we also applied pairwise comparisons by means of the non-parametric Wilcoxon signed-ranks test [27].

4.2 Analysis of the Effect of Selection + Mutation

Our first concern was to analyze the behavior of XCS$_{GA}$ and XCS$_{ES}$ when only the selection and the mutation operator were considered. For this purpose, Table 2 supplies the test accuracies and the standard deviations obtained by XCS$_{GA}$ and XCS$_{ES}$ with 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). In all the cases, crossover was switched off. The last three rows of the table supply the average performance and the average rank of each learning technique, and its position in the ranking.

The average rank of each learner shows that two schemes based on ESs were the

Table 2: Comparison of the average test performance and standard deviation obtained by 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$_{ES}$ (w. XCS$_{ES}$) are also provided. In all the runs, crossover was switched off. The last three rows provide the average accuracy and standard deviation, the average rank, and the position of each learner in the ranking.

DS	**XCS$_{GA}$-ps**	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**	**w. XCS$_{ES}$**
bal	82.35 ± 2.54	82.08 ± 2.22	81.55 ± 2.14	81.71 ± 2.27	81.12 ± 2.63	82.93 ± 3.06
bpa	62.51 ± 2.96	64.15 ± 2.74	62.80 ± 2.90	65.12 ± 2.43	62.70 ± 2.47	62.51 ± 2.60
gls	66.51 ± 1.70	67.13 ± 1.66	66.98 ± 1.75	69.94 ± 2.17	67.29 ± 2.18	67.91 ± 1.62
h-s	41.23 ± 2.07	43.46 ± 1.75	41.60 ± 2.42	42.72 ± 2.24	37.78 ± 2.00	39.63 ± 1.99
irs	94.89 ± 0.70	93.33 ± 0.75	95.33 ± 0.67	94.89 ± 0.66	94.89 ± 0.68	94.44 ± 0.71
pim	70.83 ± 3.04	71.05 ± 3.04	69.99 ± 2.96	72.87 ± 4.22	70.88 ± 2.41	70.53 ± 3.18
tao	89.32 ± 4.99	92.90 ± 5.15	89.79 ± 5.23	93.80 ± 3.72	93.01 ± 5.13	89.90 ± 6.26
thy	94.73 ± 1.03	95.50 ± 0.98	95.66 ± 1.21	96.28 ± 1.18	94.88 ± 1.18	95.66 ± 0.97
veh	65.52 ± 3.75	66.00 ± 3.46	64.50 ± 4.22	67.26 ± 3.74	63.83 ± 4.18	64.34 ± 2.85
wbcd	80.88 ± 2.42	85.84 ± 3.18	81.26 ± 3.21	85.65 ± 2.80	82.50 ± 3.12	82.93 ± 2.86
wdbc	78.68 ± 3.82	75.28 ± 3.98	80.20 ± 3.99	74.93 ± 4.07	67.60 ± 3.86	69.13 ± 3.58
wne	80.71 ± 1.35	86.70 ± 1.00	82.21 ± 1.53	82.02 ± 1.17	78.09 ± 1.73	81.65 ± 1.31
Avg	75.68 ± 2.53	76.95 ± 2.49	75.99 ± 2.69	77.27 ± 2.56	74.55 ± 2.63	75.13 ± 2.58
Rnk	4.38	2.67	3.54	2.00	4.50	3.92
Pos	**5**	**2**	**3**	**1**	**6**	**4**

best ranked methods in the comparison. The Friedman's test permitted rejecting the hypothesis that all the learners were statistically equivalent, on average, with a p-value of 0.0045. Both, the Bonferroni-Dunn test and the Holm's procedure, at $\alpha = 0.05$, indicated that XCS$_{ES}$ with tournament selection significantly outperformed XCS$_{GA}$ with both proportionate selection. In addition, the Bonferroni-Dunn test also identified that XCS$_{ES}$ with proportionate selection was significantly better than XCS$_{GA}$ with proportionate selection.

Three important observations can be drawn from these results. First, XCS$_{ES}$ based on truncation selection resulted in the poorest performance of the comparison. We hypothesize that this behavior was due to the fact that truncation selection is an excessively elitist operator that makes strong pressure toward the fittest individuals, which goes in detriment of the population diversity. Second, 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 to proportionate selection, confirming the empirical and theoretical studies presented in [7] and [22] for the ternary rule representation. Therefore, our analysis enables us to extend these conclusions to real-world problems. Third, XCS$_{ES}$ presented brilliant results in the *tao*, the *wbcd*, and the *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 performance ever reported for 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 [3]. Therefore, the guided search due to the incremental adaptation of the strategy parameters provided by Gaussian mutation enabled the system to create new classifiers that approximated these complex class boundaries[1] more accurately. These results were aligned with our initial intuition (see Sect. 2.5), in which we already discussed the need of designing operators that permitted this fine tuning in complex regions of the feature space.

Overall, the results indicated that the mutation introduced by XCS$_{ES}$ had a greater search power due to a more guided search toward optimal classifiers introduced by Gaussian mutation. We hypothesized that these better behavior was because Gaussian mutation was assuming 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 weighted Gaussian mutation. This new operator normalized the random values obtained from the Gaussian distribution to the range [0,1]; therefore, it decreased the power of the Gaussian mutation by softening the changes that it produced to the classifier variables. The results obtained with the new operator are shown in the last column of Table 2. Note that the results of XCS$_{ES}$ with weighted Gaussian mutation were similar to those achieved by XCS$_{GA}$. This supported the hypothesis that non-weighted Gaussian mutation provided a more powerful search capability than that of random mutation, promoting the global search capabilities of XCS$_{ES}$.

4.3 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 2-point crossover for XCS$_{GA}$ (see Sect. 2.4) and a combination of discrete recombination for object parameters and intermediate recombination for strategy parameters for XCS$_{ES}$ (see Sect. 3.2).

Table 3 shows the test accuracy and the standard deviation of the different configurations of XCS on the same collection of real-world problems. Several observations can be drawn from the results. First, it is worth noting that the inclusion of crossover led to an improvement of the test accuracy achieved by XCS in most of the data sets not only for XCS$_{GA}$, but also for XCS$_{ES}$. Table 4 shows the p-values resulting of the comparison of each configuration of XCS with and without crossover according to a Wilcoxon signed-ranks test. Note that, in all cases, the inclusion of crossover results in a significant improvement. Therefore, although theoretical studies that show the benefits of crossover in XCS are lacking, these results support the hypothesis that, in general, its use is beneficial to solve complex classification real-world problems with both GAs and ESs. Second, as in the previous section, the XCS's schemes based on ESs were the best ranked in the comparison. The multi-comparison test rejected the null hypothesis that all the learners performed the same, on average, with a p-value of 0.013. Both the

[1]In general, we consider that complex class boundaries are those that are not aligned with the knowledge representation used by the learner. For example, curved boundaries are difficult to approximate with a hyper rectangular representation such as the one used by XCS.

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

DS	**XCS$_{GA}$-ps**	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**
bal	83.20 ± 2.60	82.77 ± 2.44	82.72 ± 2.52	82.77 ± 2.44	82.13 ± 2.78
bpa	68.21 ± 2.88	67.05 ± 2.59	65.22 ± 2.85	65.70 ± 3.02	64.06 ± 2.80
gls	72.12 ± 1.98	71.18 ± 1.50	73.21 ± 1.84	71.65 ± 1.43	69.00 ± 1.90
h-s	46.91 ± 2.04	51.23 ± 2.59	47.04 ± 2.35	49.13 ± 1.91	44.32 ± 2.47
irs	95.33 ± 0.63	95.33 ± 0.52	94.89 ± 0.61	95.11 ± 0.56	94.89 ± 0.76
pim	72.53 ± 2.75	74.43 ± 2.90	73.39 ± 3.44	73.83 ± 3.67	74.74 ± 3.86
tao	91.22 ± 5.06	93.52 ± 4.77	91.19 ± 5.36	94.35 ± 4.86	94.17 ± 4.33
thy	95.81 ± 1.29	95.50 ± 1.33	96.43 ± 1.19	95.66 ± 1.25	95.66 ± 1.42
veh	71.79 ± 4.49	71.75 ± 5.16	71.20 ± 3.22	72.89 ± 3.54	71.20 ± 4.31
wbcd	94.85 ± 1.99	95.47 ± 1.39	93.51 ± 1.99	95.47 ± 1.83	92.61 ± 2.23
wdbc	91.09 ± 1.75	91.80 ± 2.13	92.44 ± 2.19	92.85 ± 2.27	89.51 ± 2.01
wne	95.50 ± 1.34	96.25 ± 1.15	95.69 ± 1.22	96.25 ± 1.06	91.38 ± 1.09
Avg	81.55 ± 2.40	82.19 ± 2.37	81.41 ± 2.40	82.14 ± 2.32	80.31 ± 2.50
Rnk	2.79	2.50	3.33	2.17	4.21
Pos	**3**	**2**	**4**	**1**	**5**

Table 4: P-values of the comparison of the different configurations of XCS without and with crossover. In all cases, XCS with crossover yields significantly better results than XCS without crossover.

	XCS$_{GA}$-ps	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**
p-value	$4.88 \cdot 10^{-4}$	$9.77 \cdot 10^{-4}$	$9.77 \cdot 10^{-4}$	$3.40 \cdot 10^{-3}$	$9.77 \cdot 10^{-4}$

post-hoc Bonferroni-Dunn test and the Holm's procedure, at $\alpha = 0.05$, identified that XCS$_{ES}$ with tournament selection outperformed XCS$_{ES}$ with truncation selection. In addition, the Bonferroni-Dunn test also detected that XCS$_{ES}$ with proportionate selection outperformed XCS$_{ES}$ with truncation selection. No further significant differences were detected.

Hence, differently from the comparison in the previous section, the results obtained by the XCS schemes based on ESs were not significantly better than those achieved by the XCS schemes based on GAs. That is, the introduction of crossover was more beneficial to XCS$_{GA}$ than to XCS$_{ES}$. This behavior can be explained with the following. In XCS$_{ES}$, the search is basically guided by the mutation operator, and crossover complements this search. Thus, crossover provides a moderate benefit to XCS$_{ES}$, since the principal task is done by mutation. Conversely, in XCS$_{GA}$, the crossover operator plays a key role since it is the main operator to create new promising solutions; this observation is sustained by the large difference between the results obtained by XCS$_{GA}$

with and without crossover.

Despite the improvement observed in XCS$_{GA}$ when crossover is activated, notice that XCS$_{ES}$ continues being the best ranked learner on average. These better results of XCS$_{ES}$ are mainly due to the modifications done on the mutation scheme. In the next section, we focus our attention in the second important genetic operator: crossover. We consider some of the ideas that have appeared in this analysis and design a more flexible crossover operator for both XCS$_{GA}$ and XCS$_{ES}$.

5 New BLX Crossover

One of the important aspects of XCS$_{ES}$ is that its Gaussian mutation operator produces a nice pressure toward the creation of classifiers that approximate the class boundary accurately, which is especially important in problems with complex class boundaries. That is, in XCS$_{ES}$, classifiers that are close to the class boundary will have small strategy parameters, and so, Gaussian mutation is likely to produce new offspring that are slightly different from their parents and that approximate the decision boundary more accurately. In this case, as seen in the last section, crossover, despite being useful, plays a secondary role. On the other hand, the search scheme presented by XCS$_{GA}$ is very different. That is, the mutation operator of XCS$_{GA}$ performs a non-guided, constant local search and has a lower effect with respect to the mutation of the ES scheme. Thus, in XCS$_{GA}$, crossover is mainly responsible for creating new promising solutions; however, 2-point crossover may be very disruptive, especially when crossing classifiers that are close to the class boundary. These observations highlight the need of further studying the classical genetic operators in XCS$_{GA}$ and create some of them that enable the fine-grained tuning of the condition of classifiers that lay closely to the class boundary provided by XCS$_{ES}$.

Taking these observations as inspiration, in this section, we design a new crossover operator that aims at (1) permitting the generation of a larger number of offsprings with respect to 2-point crossover by enabling the modification of the interval values of each variable and (2) being more sensitive in combining the information of the parents, searching a balance between local search and exploration of new regions of the feature space. The new operator is addressed as BLX crossover since it is inspired by the BLX-α crossover operator of *real-coded GAs* [14]. The operator is defined for both XCS-GA and XCS-ES. The following describes the new operator for both schemes of XCS.

5.1 BLX Crossover for XCS$_{GA}$

Given two parent classifiers $p_1 = (x_1^{p_1}, x_2^{p_1}, \ldots, x_\ell^{p_1})$ and $p_2 = (x_1^{p_2}, x_2^{p_2}, \ldots, x_\ell^{p_2})$, where each variable is represented by the interval of feasible values $x_i^{p_i} = [\ell_{x_i}^{p_i}, u_{x_i}^{p_i}]$, the BLX crossover operator generates the interval for each variable i of the two offspring classifiers o_1 and o_2 as follows. First, a random value α ranging in [0, 0.5] is generated. Then, the minimum lower bound c_{min} and the maximum upper bound c_{max} of the two parents is computed, i.e., $c^i_{min} = min(\ell^{p_1}_{x_i}, \ell^{p_2}_{x_i})$ and $c^i_{max} = max(u^{p_1}_{x_i}, u^{p_2}_{x_i})$. The distance between c^i_{max} and c^i_{min} is calculated: $I = c^i_{max} - c^i_{min}$. Finally, I is used to generate the lower bound and the upper bound of the variable i of offspring o_1 and o_2, that is,

$$\ell_{x_i}^{o_1} = c_{min} + \alpha \cdot I \cdot rand(\{-1, 1\}) \tag{5}$$
$$\ell_{x_i}^{o_2} = c_{min} + (1-\alpha) \cdot I \cdot rand(\{-1, 1\}) \tag{6}$$
$$u_{x_i}^{o_1} = c_{max} + \alpha \cdot I \cdot rand(\{-1, 1\}) \tag{7}$$
$$u_{x_i}^{o_2} = c_{max} + (1-\alpha) \cdot I \cdot rand(\{-1, 1\}) \tag{8}$$

where $rand(\{-1, 1\})$ returns either -1 or 1 with the same probability.

There are two key differences between this new crossover scheme and 2-point crossover. First, BLX crossover permits the generation of a larger number of different offspring, since each variable is adjusted individually. On the other hand, 2-point crossover does not modify the value of the variables, so restricting the possible offspring candidates. Second, α enables controlling the disruption in the generation of the offspring. That is, low values of α would imply the generation of offspring that would be very similar to one of the parents, performing a type of local search; conversely, a large value of α would result in offspring whose conditions differ from their parent ones, probably exploring new regions of the feature space. In addition, one of the offspring is generated considering α, while the other uses $(1 - \alpha)$. This combines the generation of offspring which are close to their parents with the creation of solutions that explore different regions of the solution space. For all these reasons, we consider that the BLX crossover balances local search and innovation.

5.2 BLX Crossover for XCS$_{ES}$

The same strategy explained in the previous section is used in XCS$_{ES}$ to cross the classifier condition. In addition, the vector of strategy parameters $s_1 = (\sigma_1^{p_1}, \sigma_2^{p_1}, \ldots, \sigma_\ell^{p_1})$ of parent p_1 and the vector of strategy parameters $s_2 = (\sigma_1^{p_2}, \sigma_2^{p_2}, \ldots, \sigma_\ell^{p_2})$ of parent p_2 are crossed as follows. For each offspring o_i, a new vector of strategy parameters

$$s_i = (\sigma_1^{o_i}, \sigma_2^{o_i}, \ldots, \sigma_\ell^{o_i}) \tag{9}$$

is generated. $\sigma_j^{o_i}$ is a random number ranging in $[c_{min}^i - I_i\alpha, c_{max}^i + I_i\alpha]$, where $c_{min}^i = min(\sigma_i^{p_1}, \sigma_i^{p_2})$, $c_{max}^i = max(\sigma_i^{p_1}, \sigma_i^{p_2})$, and $I_i = (c_{min}^i - c_{max}^i)$.

6 Experimental Results Using BLX Crossover

In this section, we continue the analysis provided in Sect. 4 by introducing the new BLX crossover operator into the comparison of XCS$_{ES}$ and XCS$_{GA}$. For this purpose, we follow the same experimental methodology: we ran experiments on the collection of 12 real-world problems and compare the results by means of statistic tests. In the following subsections, we first statistically compare the overall results and then provide some further observations about the behavior of the new operator on particular data sets.

6.1 Comparison of XCS$_{GA}$ and XCS$_{ES}$ with BLX Crossover

We start our analysis by repeating the same experiments performed in Sect. 4.3, but using BLX crossover in all the configurations. Table 5 provides the test performance and standard deviation of XCS$_{GA}$ and XCS$_{ES}$ with proportionate selection and tournament selection when the new BLX crossover operator is employed (see from the 2nd to the 5th column of the table). In addition, as done in the previous experiments, we also used truncation selection for XCS$_{ES}$, since this selection operator is widely used in the ESs realm. To highlight the improvement provided by the different configurations that use BLX crossover with respect to the original XCS system [29], that is, the scheme that uses proportionate selection, random mutation, and 2-point crossover, the results obtained with the original XCS are included in the last column of the table. The last two rows of the table supply the average rank of each learning technique and its position in the ranking. As proceeds, we report three analyses conducted on the results: (1) the study of the improvement provided by BLX crossover for each particular learning heuristic and configuration, (2) the comparison of the best XCS configuration among the ones that use BLX and the original XCS, and (3) the comparison of the best configuration with BLX crossover and the best configuration with the original crossover operator.

First, we compared the results obtained with BLX crossover (see Table 5) with respect to those achieved by the same learners but with their original crossover operator (see Table 3). Table 6 summarizes the p-values obtained for this comparison according to a Wilcoxon signed-ranks test. The pairwise analysis, at $\alpha = 0.05$, permitted rejecting only the null hypothesis that the results obtained by XCS$_{ES}$ with truncation selection and discrete crossover were, on average, equivalent to those achieved by the same learner but with BLX crossover. In this case, BLX crossover enabled XCS$_{ES}$ to evolve significantly more accurate models. Further differences were not found. Nevertheless, it is worth noting the average improvement with which BLX crossover provided XCS$_{GA}$ with tournament selection. In 8 out of the 12 tested problems, XCS$_{GA}$ with BLX crossover built models that were more accurate than those created by XCS$_{GA}$ with 2-point crossover.

Second, we compared the results obtained with the different schemes that use BLX crossover and the original XCS (see Table 5). The three best ranked methods in the comparison corresponded to configurations that used BLX crossover. More specifically, XCS$_{GA}$ with the both types of selection and XCS$_{ES}$ with tournament selection outperformed the original XCS scheme.This highlighted the suitability of the new crossover operator. Note, moreover, that all the schemes of XCS$_{GA}$ with BLX crossover had a better average rank than the schemes of XCS$_{ES}$ with BLX crossover, which indicated that the GA-based schemes benefited more than the ES-based schemes from the new BLX crossover operator. These better alignment of BLX crossover with GA-based search was expected, since BLX crossover introduced a flexibility in the generation of offspring that was provided neither by the XCS$_{GA}$ original mutation operator nor by the 2-point crossover operator. Conversely, in ES-based schemes, Gaussian mutation already provided the system with enough flexibility to create new classifiers that fit complex boundaries more accurately than their parents, and so, the improvement produced by BLX crossover was moderate.

Table 5: Comparison of the test performance and standard deviation obtained by XCS$_{GA}$ and XCS$_{ES}$ with proportionate selection (ps) and tournament selection (ts). Moreover, the results of XCS$_{ES}$ with truncation selection (tr) are also provided. The last column provides the results of the original definition of XCS, i.e., XCS-GA with proportionate selection, random mutation, and 2-point crossover (tp). The last three rows provide the average accuracy and standard deviation, the average rank, and the position of each learner in the ranking.

DS	**XCS$_{GA}$-ps**	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**	**XCS$_{GA}$-tp**
bal	86.29 $\pm$ 1.85	85.39 $\pm$ 2.06	85.44 $\pm$ 1.91	85.55 $\pm$ 1.82	85.55 $\pm$ 1.61	83.20 $\pm$ 2.60
bpa	69.18 $\pm$ 2.66	65.89 $\pm$ 2.34	68.11 $\pm$ 2.35	68.11 $\pm$ 2.28	66.76 $\pm$ 3.10	68.21 $\pm$ 2.88
gls	69.94 $\pm$ 1.56	70.25 $\pm$ 2.27	71.65 $\pm$ 1.71	71.49 $\pm$ 1.71	69.16 $\pm$ 1.88	72.12 $\pm$ 1.98
h-s	39.51 $\pm$ 2.14	38.64 $\pm$ 2.09	38.76 $\pm$ 2.33	39.88 $\pm$ 2.54	43.21 $\pm$ 2.90	46.91 $\pm$ 2.04
irs	95.11 $\pm$ 0.63	95.11 $\pm$ 0.62	95.33 $\pm$ 0.58	95.11 $\pm$ 0.63	94.89 $\pm$ 0.66	95.33 $\pm$ 0.63
pim	75.22 $\pm$ 2.42	74.43 $\pm$ 2.99	74.65 $\pm$ 3.13	73.83 $\pm$ 3.47	74.65 $\pm$ 2.45	72.53 $\pm$ 2.75
tao	96.80 $\pm$ 2.76	96.77 $\pm$ 3.07	96.68 $\pm$ 3.21	96.61 $\pm$ 3.24	96.57 $\pm$ 3.03	91.22 $\pm$ 5.06
thy	97.36 $\pm$ 1.11	96.90 $\pm$ 1.17	97.05 $\pm$ 1.22	96.43 $\pm$ 1.47	96.74 $\pm$ 1.22	95.81 $\pm$ 1.29
veh	69.34 $\pm$ 3.72	69.23 $\pm$ 3.76	69.74 $\pm$ 3.67	70.21 $\pm$ 3.04	69.31 $\pm$ 3.19	71.79 $\pm$ 4.49
wbcd	95.66 $\pm$ 2.08	96.28 $\pm$ 1.66	96.33 $\pm$ 1.97	95.99 $\pm$ 1.73	96.33 $\pm$ 2.00	94.85 $\pm$ 1.99
wdbc	91.04 $\pm$ 2.16	91.33 $\pm$ 1.86	91.62 $\pm$ 2.23	91.97 $\pm$ 2.29	90.63 $\pm$ 2.06	91.09 $\pm$ 1.75
wne	95.88 $\pm$ 1.14	96.07 $\pm$ 0.94	96.25 $\pm$ 1.25	96.82 $\pm$ 1.28	95.13 $\pm$ 1.20	95.50 $\pm$ 1.34
Avg	81.78 $\pm$ 2.02	81.36 $\pm$ 2.07	81.80 $\pm$ 2.13	81.83 $\pm$ 2.12	81.58 $\pm$ 2.11	81.55 $\pm$ 2.40
Rank	3.00	4.08	2.67	3.17	4.29	3.79
Pos	**2**	**5**	**1**	**3**	**6**	**4**

Table 6: P-values of the comparison of the different configurations of XCS without 2-point crossover and and with BLX crossover.

	XCS$_{GA}$-ps	**XCS$_{ES}$-ps**	**XCS$_{GA}$-ts**	**XCS$_{ES}$-ts**	**XCS$_{ES}$-tr**
p-value	0.52	0.83	0.47	0.77	0.05

We statistically compared the test performance of the six learning methods. The multi-comparison Friedman's test permitted rejecting the null hypothesis that all the learners were statistically equivalent with *p-value*$= 0.10$. Nonetheless, neither the Bonferroni-Dunn test nor the Holm's procedure could detect any significant difference between the best ranked learner, i.e., XCS$_{GA}$ with tournament selection, and any other learning technique. Therefore, we applied a pairwise analysis by means of the Wilcoxon signed-ranks test at $\alpha = 0.10$ to detect further differences. It is well known that pairwise comparisons increase the risk of rejecting null hypotheses that are actually true. Herein, we assumed this risk with the aim of providing more information about the excellence of the different methods. The pairwise analysis detected (1) that XCS$_{GA}$ with tournament selection generated models that were significantly more accurate than those created by XCS$_{ES}$ with proportionate and truncation selection and (2) that XCS$_{ES}$ with tournament selection outperformed XCS$_{ES}$ with proportionate

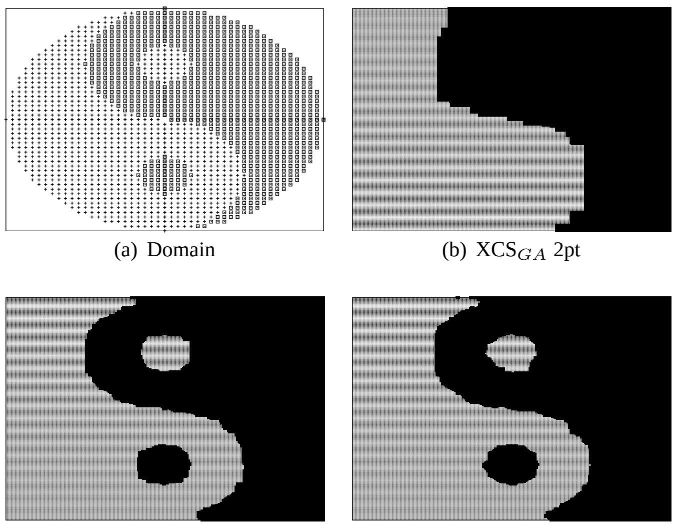
(a) Domain (b) XCS$_{GA}$ 2pt (c) XCS$_{GA}$ BLX (d) XCS$_{ES}$ BLX

Figure 4: Decision boundaries obtained with original scheme of XCS$_{GA}$ with (b) 2-point crossover, (c) XCS$_{GA}$ with BLX crossover, and (d) XCS$_{ES}$ with BLX crossover in the (a) tao problem.

selection. No further significant differences were identified by the statistical analysis.

Third, we compared the best configuration with BLX crossover—i.e., XCS$_{GA}$ with tournament selection—with the best configuration with the original crossover—i.e., XCS$_{ES}$ with tournament selection. The Wilcoxon signed-ranks test did not permit rejecting the null hypothesis that both configuration yielded the same results on average with *p-value*$= 0.69$. Nevertheless, note that XCS$_{GA}$ with BLX crossover and tournament selection outperformed XCS$_{ES}$ with the original crossover operator and tournament selection in 7 data sets, obtained equivalent results in 2 data sets, and provided poorer results in only 3 data sets. This results highlight the robustness of XCS$_{GA}$ with BLX crossover with respect to the other configurations that use the original crossover operators of XCS$_{GA}$ and XCS$_{ES}$.

6.2 Study of the Effect of BLX Crossover on Particular Data sets

We further studied the behavior of the new crossover operator by analyzing the differences in each particular training data set with the aim of providing a glimpse of under which problem characteristics BLX crossover performed the best. In particular, we observed that BLX crossover (1) may help XCS evolve accurate models in dense problems and (2) may prevent the system from over-fitting in complex domains. In what follows, we further elaborate these two hypotheses and show results on particular data sets that support them.

First, we identified that the new operator yielded excellent results in dense problems, i.e., problems with a large number of instances with respect to the number of variables. Notice, for example, the excellent results obtained by all the configurations of XCS that use BLX in the bal and the tao problems with respect to XCS$_{GA}$ with 2-point crossover. To our knowledge, these are, by far, the best results obtained by XCS with these two problems [2, 5]. We hypothesized that this was because the new operator was able to fit complex decision boundaries more accurately since it had a less disruptive behavior than the 2-point crossover operator. To analyze this hypothesis, we did the following experiment. We ran XCS$_{GA}$ and XCS$_{ES}$ on the tao problem (see the domain in figure 4(a)) and plotted the decision boundaries of the evolved rules sets. The decision boundaries were plotted by testing the resulting rule set with a dense test set that contained instances equally distributed around the feature space and depicting with different colors these instances depending on the class predicted by the system. We selected the tao problem for this analysis since it consists of two variables, and so, the class boundaries can be easily illustrated. Besides, tao has curved class boundaries that pose a big challenge for interval-based LCSs [2]. Figure 4(b) shows the class boundaries obtained with XCS$_{GA}$ with 2-point crossover. Figures 4(c) and 4(d) show the same information but for XCS$_{GA}$ and XCS$_{ES}$ with BLX crossover. Note that XCS$_{GA}$ with 2-point crossover evolved models that obviated the two inner concepts of the tao problem; moreover, the class boundary defined by the system presented an abrupt shape, concentring a big amount of the test error. Conversely, when BLX crossover was used, XCS$_{GA}$ and XCS$_{ES}$ were able to define very accurate class boundaries and to discover the two inner concepts of the problem. It is worth noting that this specificity-driven pressure that seems to be present in the BLX crossover operator did not go in detriment of the test accuracy in problems where the training instances were more sparse, as shown in the general results presented in Table 5.

We also detected that BLX crossover may help prevent the system from over-fitting in complex problems. That is, in some specific problems of the comparison such as *bal*, *bpa*, and *tao*, we identified that XCS$_{GA}$ with 2-point crossover tended to over-fit the training instances, i.e., to create rules that covered few instances with the aim of maximizing the training accuracy in complex problems. To illustrate this behavior, Figure 5 shows the evolution of the training and the test performance achieved by XCS$_{GA}$ with 2-point crossover and BLX crossover in the *bal* problem. Note that, with 2-point crossover, the training accuracy increased during the 100 000 learning iterations, but the test accuracy started decreasing at about 10 000 iterations. This indicated that XCS$_{GA}$ with 2-point crossover was over-fitting the training instances in this particular problem. On the other hand, the results denoted that BLX crossover operator prevented XCS from over-fitting. Notice that the test accuracy remained oscillating around 86%.

Finally, let us note that the examples provided in this analysis do not enable us to extract general conclusions of the problem characteristics for which BLX crossover is better suited, but to identify some possible problem difficulties with which BLX crossover may deal more efficiently. In further work, these initial observations will be investigated in more detail, trying to identify which types of genetic operators are more suited for different problem characteristics.

The overall results presented in the section resulted in two key conclusions. The first conclusion is that the GA-based search could achieve better results, on average,

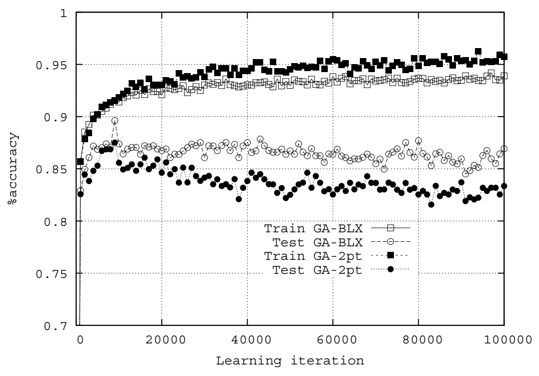

Figure 5: Evolution of the training and test accuracy of XCS $_{GA}$ with 2-point crossover and BLX crossover in the bal problem.

than the ES-based counterpart if the crossover operator is properly designed. Furthermore, as second conclusion, we also showed the suitability of BLX crossover with respect to the original crossover operators of both XCS $_{ES}$ and XCS$_{GA}$. More specifically, we illustrated the advantages of BLX crossover in some particular domains of the comparison which contained a large number of instances per dimension and complex class boundaries. Therefore, the analysis permitted not only increasing our understanding of how the different genetic operators work but also improving them, enabling the system to evolve models that were more accurate than those created by the original XCS.

7 Summary, Conclusions, and Further Work

In this paper, we analyzed and improved the genetic discovery in the XCS classifier system when the system is applied to extract classification models, represented by populations of interval-based rules, from domains that are described by continuous attributes. We proposed to systematically analyze the role of the two main genetic operators, that is, mutation and crossover, and use the observations resulting from this analysis to design new enhanced operators.

First, we started with the analysis of mutation. We proposed to replace the discovery component, driven by a GA, with an ES with the aim of providing a fine tuning of classifiers that are close to the class boundaries. Then, we studied the effect of the mutation operator in XCS$_{GA}$ and XCS$_{ES}$, detecting that the ES permitted the system to create more accurate models. Thereafter, we introduced crossover in the comparison, evidencing a clear improvement of the results. In any case, on average, XCS $_{ES}$ resulted in the most accurate models of the comparison.

Second, we turned our attention to the crossover operator. We designed a new crossover operator, addressed as BLX crossover, with the aim of diminishing the disruption of the typical crossover operators of XCS and balancing the amount of local search and crossbreeding introduced by the operator. We experimentally showed the advantages of BLX crossover with respect to the original crossover operators of XCS$_{ES}$ and XCS$_{GA}$. In particular, the GA-based discovery component benefited more from the new crossover operator than the ES-based discovery component; that is, XCS$_{GA}$ resulted in the most accurate models on average. Moreover, we further analyzed the effect of the new operator on particular data sets and provided some examples in which BLX crossover enabled XCS to fit complex class boundaries more accurately and prevented the system from over-fitting the training instances.

Globally, the study conducted along this paper served to increase our understanding of how genetic operators work in XCS. This better comprehension was used as a tool for designing new genetic operators, which let XCS extract models that were more accurate than those created by the original version of the system. In addition to all the notes provided while discussing the results, the experimentation highlighted two crucial aspects that should be addressed as further work. First, 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. Second, 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-based search scheme and further improved by the introduction of BLX crossover. Nonetheless, XCS$_{GA}$ with the original genetic operators was the best performer in few problems such as *h-s*. These observations indicate that problems with different types of complexities may benefit from different genetic operators. As further work, we will study different strategies to extract characteristics from the training data sets and link these characteristics to the properties of the genetic operators with the aim of designing hyper-heuristics that enable the system to chose the genetic operators that may maximize the system performance according to apparent complexity of each particular problem.

Acknowledgements

The authors would like to thank *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 J.M. Garrell. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[3] 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.

[4] E. Bernadó-Mansilla, X. Llorà, and J.M. Garrell. XCS and GALE: A comparative study of two learning classifier systems on data mining. In *Advances in Learning Classifier Systems*, volume 2321, pages 115–132. Springer, 2002.

[5] M. V. Butz. *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*, volume 109 of *Studies in Fuzziness and Soft Computing*. Springer, 2006.

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

[7] M. V. Butz, K. Sastry, and D. E. Goldberg. Strong, stable, and reliable fitness pressure in XCS due to tournament selection. *GPEM*, 6(1):53–77, 2005.

[8] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In *Proc. IWLCS*, volume 1996, pages 253–272. Springer, 2001.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

[10] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[11] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[12] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

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

[14] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[15] Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800–802, 1988.

[16] J. H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.

[17] 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.

[18] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[19] 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.

[20] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced datasets. *Soft Computing Journal*, doi=10.1007/s00500-008-0319-7, 2008.

[21] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Genetic-based machine learning systems are competitive for pattern recognition. *Evolutionary Intelligence*, doi=10.1007/s12065-008-0013-9, 2008.

[22] 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*, volume 2, pages 1846–1853. ACM Press, 2007.

[23] I. Rechenberg. *Cybernetic solution path of an experimental problem*, volume 1122. 1965.

[24] I. Rechenberg. *Evolution strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998.

[26] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[27] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

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

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