

**Modeling XCS in Class Imbalances: Population
Size and Parameter Settings**

**Albert Orriols-Puig, David E. Goldberg, Kumara
Sastry and Ester Bernadó-Mansilla
IlliGAL Report No. 2007001
January 2007**

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

Modeling XCS in Class Imbalances: Population Size and Parameter Settings

Albert Orriols-Puig^{†‡}, David E. Goldberg[†], Kumara Sastry[†] and Ester Bernadó-Mansilla[‡]

[†]Illinois Genetic Algorithm Laboratory (IlliGAL)
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

[‡]Group of Research in Intelligent Systems
Computer Engineering Department
Enginyeria i Arquitectura La Salle — Ramon Llull University
Quatre Camins, 2. 08022, Barcelona, Catalonia, Spain.

aorriols@salle.url.edu, deg@uiuc.edu, ksastry@uiuc.edu, esterb@salle.url.edu

Abstract

This paper analyzes the scalability of the population size required in XCS to maintain niches that are infrequently activated. Facetwise models have been developed to predict the effect of the imbalance ratio—ratio between the number of instances of the majority class and the minority class that are sampled to XCS—on population initialization, and on the creation and deletion of classifiers of the minority class. While theoretical models show that, ideally, XCS scales linearly with the imbalance ratio, XCS with standard configuration scales exponentially. The causes that are potentially responsible for this deviation from the ideal scalability are also investigated. Specifically, the inheritance procedure of classifiers’ parameters, mutation, and subsumption are analyzed, and improvements in XCS’s mechanisms are proposed to effectively and efficiently handle imbalanced problems. Once the recommendations are incorporated to XCS, empirical results show that the population size in XCS indeed scales linearly with the imbalance ratio.

1 Introduction

XCS (Wilson, S.W., 1995; Wilson, S.W., 1998), one of best representatives of Michigan-style Learning Classifier Systems (LCSs) (Holland, J.H., 1975), is a robust method that has been successfully applied to solve large boolean problems (Butz, M.V., Sastry, K., & Goldberg, D.E., 2003), multi-step problems (Lanzi, P. L. & Wilson, S. W., 2000), datamining problems (Bernadó-Mansilla, E., Llorà, X., & Garrell, J.M., 2002; Bacardit, J. & Butz, M.V., 2004), and function approximation problems (Wilson, S.W., 2002). LCS literature has usually considered problems with near-equal number of instances per class. Nonetheless, many real-world problems are imbalanced, and only few studies on the effect of class imbalances on LCSs have been conducted (Holmes, J.H., 1998; Orriols-Puig, A. & Bernadó-Mansilla, E., 2006; Orriols-Puig, A. & Bernadó-Mansilla, E., *ress*). While population sizing and scalability in XCS has broadly been studied on balanced domains (Butz, M.V., 2004), similar studies are lacking for imbalanced domains.

The aim of this paper is to model the effect of learning from imbalanced data in XCS, and to analyze the scalability of the population size required as the amount of class imbalance increases. We develop facetwise models that predict the impact of the imbalance ratio on the population initialization, and on the creation and deletion of classifiers of the minority class. Moreover, we derive population size bounds that guarantee that XCS will create and maintain these classifiers of the minority class. We design two test problems of bounded difficulty to validate the model: the *one-bit* problem, and the *parity* problem. Results obtained with the *one-bit* problem show that the population size scales exponentially with the imbalance ratio, violating the model bounds. We investigate the causes of this deviation from the ideal scalability, and propose some approaches to alleviate the early deletion of classifiers of the minority class. Once these recommendations are incorporated to XCS, the empirical results on both the *one-bit* and the *parity* problems agree with the theoretical bounds, showing that population size in XCS scales linearly with the imbalance ratio.

The remainder of the paper is organized as follows. XCS is briefly described in section 2. Next, we develop theoretical models for learning from imbalanced datasets, deriving a population size bound to ensure the growth of minority class niches. Section 4 introduces the test problems used in the experimentation. In section 5, we run XCS with one of the test problems, and empirical results show a deviation from the ideal scaling-up of the population size. Section 6 analyzes the causes of the deviation, resulting in a set of recommendations on how to set the system when learning from imbalanced data. These recommendations are grouped in XCS+PMC. The theoretical model is empirically validated using XCS+PMC in sections 7 and 8. Finally we provide further directions, summarize, and conclude.

2 XCS in a Nutshell

This section provides a brief description of the XCS classifier system. For a detailed description, the reader is referred to (Wilson, S.W., 1995; Wilson, S.W., 1998); also an algorithmic description can be found elsewhere (Butz, M.V. & Wilson, S.W., 2001).

XCS is an accuracy-based learning classifier system introduced in (Wilson, S.W., 1995). The main difference from its ancestors is that XCS computes fitness from the accuracy of the reward prediction instead on the reward itself. The accuracy-based approach makes XCS evolve a complete action map (denoted as [O]) of the environment, evolving not only high-rewarded rules (i.e., consistently correct rules), but also consistently incorrect rules (i.e., rules with zero prediction and low error).

XCS works as an online learner. For each input example, XCS forms the match set [M] consisting of all classifiers with matching condition. If not enough classes are covered in [M], XCS triggers the covering operator, which creates new classifiers with uncovered classes. Under pure exploration, a class is selected randomly, and all classifiers predicting that class form the action set [A]. The class is sent to the environment and the received reward is used to update the parameters of the classifiers in [A]. Eventually, the genetic algorithm is triggered in the action set [A], and subsumption may be applied to favor accurate and general classifiers. Under exploit mode, XCS predicts the most voted class for each input example. The class vote is computed as a fitness weighted average of the predictions of all classifiers advocating that class.

3 Complexity when Learning from Class Imbalances

In this section we investigate how class imbalances influence XCS’s learning mechanisms. We benefit from previous studies that analyze the computational complexity of XCS (Butz, M.V., 2004). Nevertheless, this theory is developed considering a uniform sampling of instances. The aim of this section is to extend the theory to class-imbalanced problems, highlighting the impact of learning from class-imbalanced domains on different mechanisms of XCS.

In order to achieve an optimal population, XCS has to evolve different niches distributed around the problem subspace. Since XCS evolves complete action maps, it will evolve both high rewarded and low rewarded niches. High rewarded niches are niches that contain accurate classifiers, addressed as correct classifiers in the remainder of this paper. Low rewarded niches are niches that contain accurate but inconsistent classifiers, addressed as incorrect classifiers. Both correct and incorrect classifiers have all the bits of the schema that they represent correctly specified; however, correct classifiers predict the correct class, whereas incorrect classifiers predict wrongly all the instances they match.

In imbalanced domains, examples of one of the classes are sampled in a lower frequency than the others; thus, niches activated by these instances are poorly nourished. We are interested in the discovering and maintenance of classifiers belonging to high rewarded niches but poorly nourished, to which we refer as correct classifiers of the minority class. For that purpose, we analyze the following points:

- *Population initialization.* We analyze if the covering operator can supply enough schemas of the minority class in the first stage of the learning process.
- *Generation of correct classifiers of the minority class.* We analyze the probability that the GA generates correct classifiers of the minority class when there are not representatives of the minority class in the population.
- *Time of extinction of correct classifiers of the minority class.* We derive the average life-time of correct classifiers of the minority class.

From the analysis of the three points above, we study the maintenance of niches poorly nourished. We derive a bound on the population size to ensure that XCS will be able to maintain correct classifiers of the minority class and that these classifiers will receive, at least, one genetic event before being removed. In the analysis, we consider problems that consist of n classes in which one of the classes, addressed as the minority class, is sampled in a lower frequency than the others. Specifically, the minority class is sampled with a probability $1/(1+ir)$. Thus, the model derived focuses on the creation, maintenance and deletion of correct classifiers that advocate the minority class. For the discovering and maintenance of niches of other classes we rely on the theory presented elsewhere (Butz, M.V., 2004).

3.1 Population Initialization

In this section we analyze the population initialization when instances of one of the classes are under-sampled. We recall the covering challenge (Butz, M.V., 2004) to justify the need for a high generalization in covering, and quantify the supply of classifiers with correctly specified schemas of the minority class in imbalanced datasets.

The covering procedure in XCS works as follows. At each learning iteration, the environment samples a new input instance, and XCS creates the match set, which consists of all classifiers in the

population that match the current input instance. If some class is not represented in the match set, the covering operator is activated to create a new classifier advocating that class with a condition generalized from the current input. The amount of generalization over the inputs is controlled with the parameter $P_{\#}$.

In (Butz, M.V., 2004) the covering challenge is defined to prevent XCS from getting stuck in an infinite covering-deletion loop. That is, if either the population size is too small or the initial specificity is too high ($P_{\#}$ is set too low), XCS might be continuously covering and removing classifiers, and thus, genetic pressures would never take off. The author points out that this issue can be easily avoided if $P_{\#}$ is set high enough.

Class imbalances do not affect the covering challenge; so, the bounds obtained in (Butz, M.V., 2004) still hold. In the remainder of this analysis, we assume that $P_{\#}$ is set high enough to overcome the covering challenge.

Now, let's focus on the impact that class imbalances cause on the covering operator. Since we assumed that $P_{\#}$ is appropriately set to avoid the covering-deletion loop, covering will be activated only in the first stages of the learning. As ir increases, less instances of the minority class will be sampled during the first iterations. Thus, for high class imbalances, covering will be activated on examples of the majority class, and so, classifiers' conditions will be, mainly, a generalization of majority class instances. Consequently, the covering operator will generate, basically, the following four types of classifiers: a) correct classifiers of any class other than the minority class; b) incorrect classifiers of the minority class; c) overgeneral classifiers¹ of any class other than the minority class; and d) overgeneral classifiers of the minority class. Thus, the higher the imbalance ratio, the lower the probabilities that covering generates correct classifiers of the minority class.

However, we are interested in the probability that covering creates correct classifiers of the minority class. To create a correct classifier of the minority class, covering has to generate the classifier's condition from a minority class instance, and do not generalize any position that corresponds to the schema to be represented. So, first, we derive a lower bound for the probability that the system covers the first instance ever seen of the minority class (i.e., that there exist, at least, an overgeneral classifier in the population that matches the instance and advocates the minority class). According to (Butz, M.V., 2004), the probability that one instance is covered by, at least, one classifier is the following:

$$P(\text{cover}) = 1 - \left[1 - \left(\frac{2 - \sigma[P]}{2} \right)^{\ell} \right]^N \quad (1)$$

where ℓ is the input length, N the population size, and $\sigma[P]$ the specificity of the population. During the first learning stage of XCS, we can approximate that: $\sigma[P] = 1 - P_{\#}$.

When dealing with a certain amount of class imbalance ir , the worst case is that XCS receives ir instances of the other classes before receiving the first instance of the minority class. Let's suppose that the system receives these ir instances, and that covering has generated n new classifiers for each instance (where n is the number of classes). After that, the first instance of the minority class arrives; XCS will cover this instance with the following probability:

$$P(\text{cover}) = 1 - \left[1 - \frac{1}{n} \left(\frac{2 - \sigma[P]}{2} \right)^{\ell} \right]^{n \cdot ir} \quad (2)$$

¹Classifiers covering instances of more than one class.

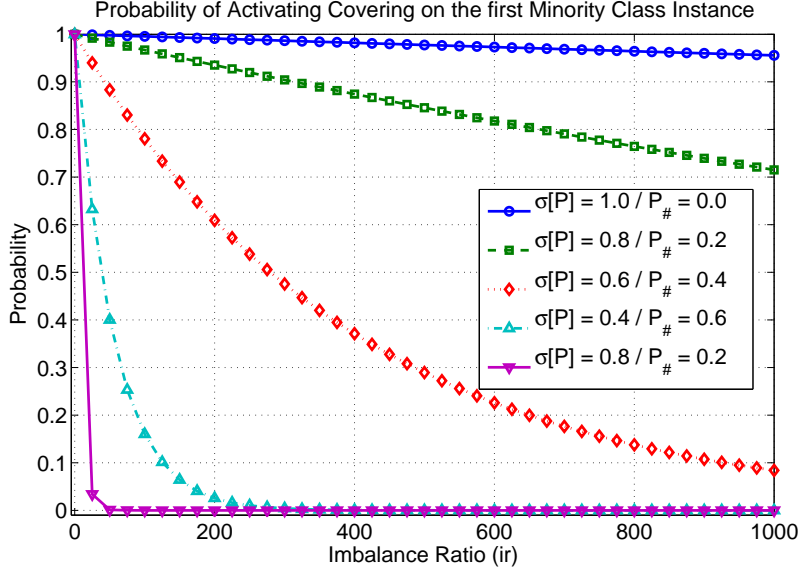


Figure 1: Probability of activating covering on a minority class instance given a certain specificity $\sigma[P]$ and the imbalance ratio ir . The curves have been drawn from formula 4 with different specificities $\sigma[P]$ and setting $\ell=20$. The figure shows that the probability that covering is activated on instances of the minority class decreases exponentially with the imbalance ratio. That is, for lower initial specificities—which are required to overcome the covering challenge—the covering operator fails at providing correct classifiers of the minority class even for low imbalance ratios.

The equation supposes that $N > n \cdot ir$, i.e., that XCS will not delete any classifier during the first ir iterations. Using the approximation $(1 - r/n)^n \approx e^{-r}$ we obtain the following expression:

$$P(\text{cover}) \approx 1 - e^{-ir \cdot \left(\frac{2-\sigma[P]}{2}\right)^\ell} \approx 1 - e^{-ir \cdot e^{-\frac{\ell\sigma[P]}{2}}} \quad (3)$$

That is, the probability of covering an instance of the minority class depends on the length of the input, the initial specificity of the population and the imbalance ratio. Given a fixed ℓ and $\sigma[P]$, the second term in the right hand of the equation decreases exponentially as the imbalance ratio increases; thus, the probability of covering minority class instances tends to one exponentially with the imbalance ratio.

If there is not any classifier that matches the input instance, covering will be activated. Thus, the probability of activating covering having sampled a minority class instance is the following:

$$P(\text{activate cov. on. min. inst.} \mid \text{sampling min. inst.}) = 1 - P(\text{cover}) \approx e^{-ir \cdot e^{-\frac{\ell\sigma[P]}{2}}} \quad (4)$$

which indicates that the probability of activating covering on minority class instances decreases exponentially with the imbalance ratio and, in a higher factor, to the condition length and the initial specificity.

Figure 1 shows the probability of activating the covering operator on the first minority class instance sampled in a problem with $\ell = 20$, $n = 2$. The figure shows that the higher the specificity, the higher the probability that covering is activated on minority class instances. Taking it to the extreme, if $\sigma[P] = 1$, that is, $P_\# = 0$, the system will create classifiers from all the instances

that are sampled. Nevertheless, this configuration would not permit any kind of generalization, requiring a population size equal to the size of the input space (i.e., $N = n2^l$). On the other hand, a high value of $P_{\#}$ implies a low probability of generating classifiers from minority class instances, and so, a low probability of obtaining correct classifiers of the minority class during covering. In the experimentation used in section 5, $P_{\#} = 0.8$; so, for $ir > 50$, the probability that covering is activated on instances of the minority class is practically zero.

The analysis made above shows up that the covering operator fails to supply correct schemas of the minority class for moderate and high imbalance ratios. Consequently, XCS will start the search with a high general population that does not contain schemas of the minority class. So, the genetic search will be the main responsible for obtaining the firsts correct classifiers of the minority class. In next section we derive the probabilities of obtaining correct classifiers of the minority class under the assumption that covering has not provided any correct schema of the minority class.

3.2 Generation of Correct Classifiers of the Minority Class

In this section, we analyze the probability of generating correct classifiers of the minority class. In the model, we assume that covering has not provided any schema of the minority class. Since mutation is the primary operator for exploring new regions of the input space, we only consider its effect in our model. Moreover, to simplify the equations, the model assumes low values of μ , that is, $\mu < 0.5$. In fact, μ takes much lower values in practice, since high values of μ can be very disruptive for the genetic search.

Correct classifiers of the minority class can be obtained while exploring any class in the feature space. Thus, they can be generated when sampling either a majority or a minority class instance. In the following, we derive the probabilities of generating a correct classifier of the minority class in either case, and gather them together deriving the probability of generating new correct classifiers of the minority class.

3.2.1 Generating Correct Classifiers of the Minority Class from Minority Class Instances

A minority class instance is sampled with the following probability:

$$P(\text{min. inst.}) = \frac{1}{1 + ir} \quad (5)$$

As XCS chooses the class to be explored randomly, XCS only explores high rewarded niches of the minority class every $1/n$ times that a minority class instance is sampled. The other $(n - 1)/n$ times, XCS will explore low rewarded niches of other classes. A correct classifier of the minority class can be created while exploring any of these n classes.

First of all, we derive the probability of generating a correct classifier of the minority class if the GA is triggered on a high rewarded niche of the minority class. We assume that covering has not provided any correct classifier of the minority class, and so, that the niche being explored contains, mainly, overgeneral classifiers of the minority class. In this case, the GA will generate a correct classifier of the minority class if all the bits of the schema are set to their correct value, and the class of the classifier is not flipped. We consider the worst case, that is, that all the bits of the schema need to be changed. That gives the following lower bound on the probability of generating a new correct classifier of the minority class:

$$P(\text{gen. min. cl} \mid \text{min. inst.} \wedge \text{min. class niche}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu) \quad (6)$$

where μ is the probability of mutation and k_m is the order of the schema. That is, the formula defines a parabola on the values of μ ; for $k_m=1$, $\mu = 0.5$ maximizes the probability. However, that high value of μ may introduce too much disruption in the genetic search. Increasing the order of the schema k_m decreases exponentially the probability of obtaining a correct classifier of the minority class.

Second, XCS can also generate a new representative of the minority class while exploring niches of other classes. In this case, not only all bits of the schema have to be correctly set, but also the class has to be mutated to the minority class. Thus, recalling that the worst case is that all the bits of the schema need to be changed to their correct values, the lower bound on the probability of generating a minority class classifier is:

$$P(\text{gen. min. cl} \mid \text{min. inst.} \wedge \neg \text{min. class niche}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1} \quad (7)$$

As above, the probability of generating a new correct classifier of the minority class depends on the mutation probability μ and the order of the schema k_m . Moreover, it also depends inversely on the number of classes of the problem.

3.2.2 Generating Correct Classifiers of the Minority Class from Other Instances

XCS can also create correct classifiers of the minority class when sampling instances that do not belong to the minority class. The probability of sampling these instances is:

$$P(\neg \text{min. inst.}) = \frac{ir}{1 + ir} \quad (8)$$

Again, to create a correct classifier of the minority class all the bits of the schema have to be correctly specified. As we are exploring niches that match instances of any class other than the minority class, we consider the worst case, that is, all the bits of the schema have to be specified. Moreover, if XCS is exploring a niche of any class other than the minority class, the class also needs to be mutated to the minority class. Thus, the probabilities of creating a correct classifier of the minority class, having sampled an instance of any class other than the minority class, are:

$$P(\text{gen. min. cl} \mid \neg \text{min. inst.} \wedge \text{min. class niche}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu) \quad (9)$$

$$P(\text{gen. min. cl} \mid \neg \text{min. inst.} \wedge \neg \text{min. class niche}) = \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1} \quad (10)$$

The probabilities obtained are equivalent to the ones obtained in formulas 6 and 7 respectively. Thus, the probabilities are mainly guided by the probability of mutation μ and the order of the schema k_m .

3.2.3 Time to Create Correct Classifiers of the Minority Class

Above, we derived the lower bounds on the probabilities of generating correct classifiers of the minority class if the genetic algorithm is triggered on any niche of the system. Now, we use this

probabilities to derive the minimum time required to generate the first correct representant of the minority class.

A niche receives a genetic event if it is activated and the average time since the last application of the GA is greater than a threshold θ_{GA} . For the formulas below, we consider $\theta_{GA} = 0$; that is, we assume that a niche receives a genetic event any time it is activated. Under these circumstances, the probability of generating a correct classifier of the minority class is the sum of the following probabilities:

- The probability p_1 of generating a minority class classifier when sampling a minority class instance and activating a niche of the same class. That is, $p_1 = \frac{1}{1+ir} \cdot \frac{1}{n} \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu)$.
- The probability p_2 of generating a minority class classifier when sampling a minority class instance and activating a niche of any class other than the minority class. That is, $p_2 = \frac{1}{1+ir} \cdot \frac{n-1}{n} \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1}$.
- The probability p_3 of generating a minority class classifier when sampling an instance of any class other than the minority class and activating a niche of the minority class. That is, $p_3 = \frac{ir}{1+ir} \cdot \frac{1}{n} \left(\frac{\mu}{2}\right)^{k_m} \cdot (1 - \mu)$.
- The probability p_4 of generating a minority class classifier when sampling an instance of any class other than the minority class and activating a niche of any class other than the minority class. That is, $p_4 = \frac{ir}{1+ir} \cdot \frac{n-1}{n} \left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1}$.

Therefore, summing the four probabilities up, we obtain that the lower bound on the probability of generating a correct classifier of the minority class is:

$$P(\text{gen. min. cl}) = p_1 + p_2 + p_3 + p_4 = \frac{1}{n} \left(\frac{\mu}{2}\right)^{k_m} \quad (11)$$

From this probability, we derive the expected time until the generation of the first representant of the minority class:

$$t(\text{gen. min. cl}) = \frac{1}{P(\text{gen. min. cl})} = n \left(\frac{2}{\mu}\right)^{k_m} \quad (12)$$

which depends linearly on the number of classes and exponentially on the order of the schema, but it does not depend on the imbalance ratio.

Thus, even though covering fails to provide schemas of the minority class, XCS will be able to generate the firsts correct classifiers of the minority class independently of the imbalance ratio. In the following, we derive the time until the deletion of these classifiers. With both the generation and deletion time, we calculate the minimum population size to maintain these classifiers and to ensure the growth of best representatives of the minority class.

3.3 Deletion Time of Minority Class Classifiers

XCS deletes classifiers depending on their action set size as and their fitness. Having a good estimation of as , deletion would remove classifiers that belong to numerous niches and have low fitness. Thus, under a perfect deletion scheme, XCS would maintain classifiers in the niches poorly nourished once the first individual belonging to those niches is discovered.

Nevertheless, overgeneral classifiers bias the estimated value of as , and so, deletion works under non-perfect conditions. As argued above, there will be overgeneral classifiers in niches of the minority class. These overgeneral classifiers match several niches, and so, their as value would be high. Since as of correct classifiers of the minority class is computed from the average as of the niche, its value would tend to be over-estimated. Under these circumstances, we assume that the probability of deletion is random. As we delete two classifiers every GA application, we obtain that:

$$P(\text{delete min. cl.}) = \frac{2}{N} \quad (13)$$

From this formula, we derive the time until deletion:

$$t(\text{delete min. cl.}) = \frac{N}{2} \quad (14)$$

In the following, we use formulas 12 and 14 to derive the minimum population size that guarantees the discovery, maintenance and growth of poorly nourished niches.

3.4 Bounding the Population Size

Herein, we use the formulas of generation and deletion of minority class classifiers to derive two population size bounds. First, we derive the minimum population size to ensure that XCS will be able to create and maintain correct classifiers of the minority class. Then, we derive the population size bound to ensure the growth of the niches that contain these correct classifiers of the minority class, that is, niches poorly nourished.

3.4.1 Minimum Population Size to Guarantee Representatives

Our first concern is to assure that there would be correct classifiers representing the different niches of the minority class. For that purpose, our sake is to guarantee that, before deleting any classifier of the minority class, another correct classifier of the minority class will be created. Thus, we require that the time until deletion be greater than the time until generation of a correct minority class classifier.

$$t(\text{delete min. cl.}) > t(\text{gen. min. cl.}) \quad (15)$$

Thus:

$$\frac{N}{2} > n \left(\frac{\mu}{2}\right)^{k_m} \quad (16)$$

$$N > 2n \left(\frac{\mu}{2}\right)^{k_m} \quad (17)$$

If we write the same expression in O -notation:

$$N = O \left[2n \left(\frac{\mu}{2}\right)^{k_m} \right] \quad (18)$$

which indicates that, to assure that all the minority class niches of the system have, at least, one correct classifier, the population size have to increase linearly with the number of classes and exponentially with the order of the schema; however, it does not depend on the imbalance ratio.

3.4.2 Population Size Bound to Guarantee Reproductive Opportunities of Minority Class Classifiers

Above, we discussed the minimum time required to generate the first correct classifiers of the minority class; besides, we argued that XCS should be able to maintain representatives in niches of the minority class regardless of the imbalance ratio. Now, we are concerned about the requirements to ensure that classifiers of minority class will evolve to better ones.

To ensure the growth of niches of the minority class, we need to guarantee that the best classifiers in the niche receive, at least, one genetic opportunity. Otherwise, XCS could be continuously creating and removing classifiers from a niche, but not searching toward better classifiers. So, we first calculate the probabilities that a correct classifier of the minority class receives a GA event. As before, we consider $\theta_{GA} = 0$, and so, that any niche receives genetic event every time it is activated. Moreover, we assume that the selection procedure chooses one of the strongest classifier in the niche. Then, the probability that this classifier receive a genetic event is the probability of activation of the niche:

$$P(\text{GA niche min.}) = \frac{1}{1 + ir} \frac{1}{n} \quad (19)$$

which depends inversely on the imbalance ratio and the number of classes. From this formula, we can derive the time required to receive a genetic event:

$$t(\text{GA niche min.}) = n \cdot (1 + ir) \quad (20)$$

To guarantee that these strong classifiers of the minority class will receive a genetic opportunity before being deleted, we require that:

$$t(\text{delete min. cl}) > t(\text{GA niche min.}) \quad (21)$$

From which we derive the population size bound:

$$\frac{N}{2} > n \cdot (1 + ir) \quad (22)$$

$$N > 2n \cdot (1 + ir) \quad (23)$$

Using a O -notation:

$$N = O[2n(1 + ir)] \quad (24)$$

That is, the population size has to increase linearly with the number of classes and the imbalance ratio to warrant that correct classifiers of the minority class will receive, at least, one genetic event before being deleted.

In this section we derived a theoretical model for learning under class imbalances. The analysis revealed a failure of covering to provide schemas of the minority class for high imbalance ratios, giving the responsibility for generating correct classifiers of the minority class to mutation. Under this scenario, we showed that the time until the generation of the firsts correct classifiers of the minority class depended linearly on the number of classes and exponentially on the order of the schema, but did not depend on the imbalance ratio. Moreover, after assuming a random probability

Table 1: Complete action map for the *one-bit* problem. Regardless of the condition length ℓ , it consists of two correct classifiers, i.e., classifiers predicting the correct output (column 1) and two incorrect classifiers, i.e., classifiers predicting the incorrect output (column 2)

0# $\ell-1:0$	0# $\ell-1:1$
1# $\ell-1:1$	1# $\ell-1:0$

of deletion, we developed a population size bound that indicated that XCS should be able to create and maintain correct classifiers of the minority class regardless of the imbalance ratio; furthermore, we derived a second bound indicating that the population size should increase linearly with the imbalance ratio to ensure the growth of poorly nourished niches. In the next section we describe the test problems used to validate the theoretical models developed.

4 Facetwise Design of Test Problems

Goldberg illustrates the big importance of designing types of problems characterized by various *dimensions of problem difficulty* to permit a successful understanding of complex systems (Goldberg, D.E., 2002). We follow this approach closely to design two test problems of *bounded difficulty*, in which we can easily control the complexity introduced by the imbalance ratio. First, we design the *one-bit* problem which completely isolates the complexity introduced by the imbalance ratio from other complexity factors, such as linkages between variables or misleading fitness pressure. Second, we design a more complex problem that requires a stronger pressure toward optimal classifiers and also permit to control the imbalance ratio: the *imbalanced parity* problem.

4.1 The imbalanced One-Bit Problem

The *one-bit* problem is defined as follows. Given a binary input of length ℓ , the output is the value of the left-most bit. The problem contains four niches and only one *building block* or *schema* (Holland, J.H., 1975) per niche; moreover, the order of the schema is one. Table 1 shows the complete action map of the problem, which consists of two maximally general and accurate classifiers predicting class '0' and class '1' respectively (column 1), and two maximally general but incorrect classifiers predicting class '0' and class '1' (column 2). In the following, we will refer to these types of classifiers as correct and incorrect classifiers respectively.

The imbalance complexity is controlled by means of the probability of sampling an instance of the minority class P_{min} . At each learning iteration, the environment chooses an instance randomly. If it is a majority class instance, it is automatically given to XCS. Otherwise, the instance is passed to XCS with probability P_{min} . If it is not accepted, a new instance is randomly sampled, which undergoes the same decision process. In the remainder of this paper, we use the imbalance ratio ir to denote the ratio between the number of majority and minority class instances that are given to XCS. Given an imbalance ratio ir , the probabilities of sampling a majority class instance P_{maj} and a minority class instance P_{min} are the following:

Table 2: Complete action map for the parity problem with $\ell=4$ and $k=2$. It consists of four correct classifiers, i.e., classifiers predicting the correct output (column 1) and four incorrect classifiers, i.e., classifiers predicting the incorrect output (column 2)

00##:0	00##:1
01##:1	01##:0
10##:1	10##:0
11##:0	11##:1

$$P_{min} = \frac{1}{1 + ir} \quad (25)$$

$$P_{maj} = \frac{ir}{1 + ir} \quad (26)$$

4.2 The Imbalanced Parity Problem

The *parity* problem is a two-class binary problem originally introduced in (Kovacs, T. & Kerber, M., 2001). The problem is defined by the length of the input ℓ and the number of relevant bits k , where $\ell \geq k$. Given a binary string of length ℓ , the output is the number of one-valued bits in the first k bits modulo two. The difficulty of the problem resides in that all the k first bits form a single building block, and so, they have to be processed together. Besides, there are $p = \ell - k$ bits that are not relevant for determining the class, and so, they should be generalized. Table 2 shows the complete action map for the parity problem with $\ell = 4$ and $k = 2$.

The imbalance complexity is introduced by starving the class labeled as '1'. Besides ℓ and k , the problem is also defined by the imbalance ratio ir , where $ir \geq 1$. For $ir = 1$, the problem is equal to the parity problem. For $ir > 1$, $\frac{ir-1}{ir}2^{\ell-1}$ instances of the class labeled as '1' are taken out uniformly from the minority class niches. Regardless of the imbalance ratio, XCS is expected to evolve the same complete action map as in the parity problem (see the example in table 2).

Once defined the test problems, in the next section we run XCS with the *one-bit* problem, and compare the results with the theoretical models developed. We show that empirical results deviate from the models, and we analyze the causes of the deviations in section 6.

5 XCS on the One-Bit Problem

To validate the model derived in section 3, we first ran XCS with the *one-bit* problem with condition length $\ell = 20$ and imbalance ratios from $ir=1$ to $ir=1024$. The system was configured as follows:

$$\alpha=0.1, \epsilon_0 = 1, \nu=5, \theta_{GA}=25, \\ \chi=0.8, \mu=0.04, \theta_{del}=20, \delta=0.1, \theta_{sub}=200, P_{\#}=0.8$$

We used tournament selection, two point crossover and niched mutation (Butz, M.V. & Wilson, S.W., 2001) for the genetic algorithm. Subsumption was applied on the GA but not in the action set to avoid an excessive pressure toward overgeneral classifiers ($\theta_{sub} = 200$). β was set as suggested in (Orriols-Puig, A. & Bernadó-Mansilla, E., 2006) to assure a good estimation of parameters of overgeneral classifiers in imbalanced domains. In each experimentation, we ran XCS during

Scaling-up of the Population Size with the Imbalance Ratio in the One-Bit Problem

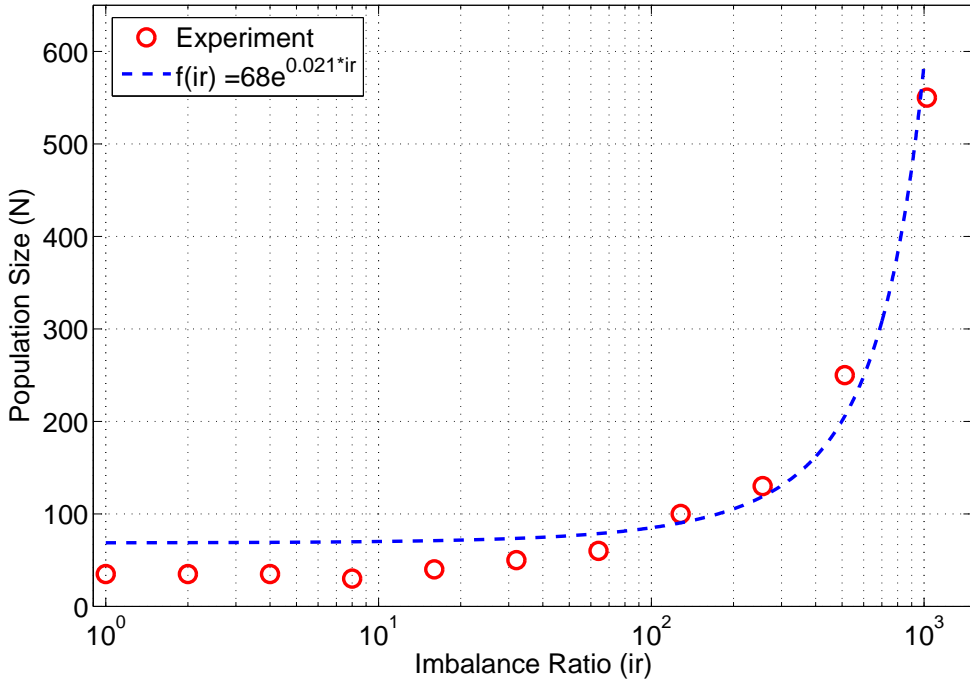


Figure 2: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *one-bit* problem with $\ell = 20$. The points indicate the empirical values for the minimum population size required by XCS to solve the problem at different imbalance ratios. The dashed line shows the curve defined by the the function $f(ir) = 68 \cdot e^{0.021 \cdot ir}$, which grows exponentially and fits the scaling-up of the population size for the highest imbalance ratios. XCS failed to solve imbalance ratios higher than $ir=1024$, even increasing exponentially the population size.

$10,000 \cdot ir$ iterations to ensure that the system received the same number of instances of the minority class for all ir .

The experimentation was made as follows. For each imbalance level, we ran XCS with different population sizes and took the minimum population size required to solve the problem. After training, we tested XCS with all the training instances, and measured the percentage of correct classifications of instances of the majority class (TN rate) and the percentage of correct classifications of instances of the minority class (TP rate). All the results were averaged over 10 different random seeds. We considered that XCS succeeded if the product of TP rate and TN rate was greater than a certain threshold θ (in the results showed herein, we set $\theta = 95\%$).

Figure 2 shows the minimum population size required by XCS in the different imbalance ratios. The points indicate the empirical values for the minimum population size required at different imbalance ratios. The dashed line draws an exponential curve that fits the points for high class imbalances. Two different regimes can be observed in the figures. In the lowest imbalance ratios, i.e., from $ir=1$ to $ir=8$, the population size required to solve the problem is constant. On the other hand, for the highest imbalance levels, i.e., from $ir=256$ to $ir=1024$, the population size increase shows an exponential tendency (fitted by the curve with dashed line). Between $ir=8$ and $ir=256$ there is a a transition region from the constant to the exponential increase.

For $ir > 1024$, XCS failed regardless of the population size, even decreasing ϵ_0 to ensure that overgeneral classifiers were not considered as accurate. Specifically, for $ir=2048$ we tried population

sizes up to $N=6,000$, that is the population size predicted by the exponential curve. XCS was not able to solve the *one-bit* problem in any of the runs.

While facetwise models derived in the previous section revealed that the population size should increase linearly to ensure the growth of poorly nourished niches, empirical results show that population size scales exponentially, and that XCS only can solve the *one-bit* problem up to $ir=1024$. In the next section, we study the causes that are potentially responsible for the deviation of the empirical results from the theoretical models.

6 Analyzing which Factors Hinder the Discovering and Maintenance of Infrequent Niches

Last section evidenced a deviation of the empirical results from the models derived in section 3. Specifically, results showed that the population size requirements increased exponentially for high imbalance ratios, while the theoretical analysis indicated that population size should scale linearly to ensure the growth of poorly nourished niches. This section studies the causes that are potentially responsible for that deviation. We analyze the effect of the mutation scheme in the discovering of new minority class classifiers, the initialization of the parameters of minority class classifiers, and the effect of subsumption on the whole population; we also introduce the need of stabilizing the population before testing. The analysis results in a series of recommendations that aim at protecting minority class classifiers from an early deletion. All these recommendations are grouped together and addressed as XCS+PCM. Afterwards, we validate the models using XCS+PCM.

6.1 Instigating the Discovery of the Minority Class: Niched Mutation versus Free Mutation

In the last section we argued that, as covering failed at providing schemas of the minority class for high imbalances, the main responsible for generating the first minority class representants was mutation. In equation 12 we showed that the time required until the generation of the first individual of the minority class depended linearly on the number of classes and exponentially on the order of the schema, but it was independent of the imbalance ratio.

The model was derived assuming an unrestricted or free mutation. Under free mutation, a minority class classifier can be created while exploring any niche of the system. However, the experiments made in section 5 used a niched mutation, as defined in (Butz, M.V. & Wilson, S.W., 2001). This kind of mutation tries to preserve the niche by forcing new classifiers to match the input instance from which the action set was created. Under niched mutation, first correct classifiers of the minority class can only be created if a minority class instance is sampled. Then, the time until generation of the first representant of the minority class is:

$$t(\text{gen. min. cl} \mid \text{niched mutation}) = n \cdot (1 + ir) \cdot \left(\frac{2}{\mu}\right)^{k_m} \quad (27)$$

which depends linearly on the imbalance ratio and the number of classes, and exponentially on the order of the schema. Thus, with niched mutation, the minimum population size to guarantee that niches of the minority class will have correct representants is the following (see equation 15):

$$N > 2n \cdot (1 + ir) \left(\frac{2}{\mu}\right)^{k_m} \quad (28)$$

which indicates that N should increase linearly with the imbalance ratio to maintain correct classifiers in minority class niches.

Consequently, free mutation incentives the discovery of minority class classifiers by permitting to explore beyond the niche. This is a crucial factor when covering cannot supply correct schemas of the different classes. Nonetheless, this is not the only factor that may hinder XCS from learning minority class niches in highly imbalance datasets. Even with niche mutation, the model indicates that population size should increase linearly with the imbalance ratio; however, the experimentation showed that the scaling-up of the population size was exponential at the highest imbalance levels. In the following we investigate other ways to prevent the early deletion of minority class classifiers.

6.2 Inheritance Error of Classifiers' Parameters

Once covering has been applied at the beginning of the learning process, the responsible for creating new better classifiers is the genetic algorithm. When triggered, the GA selects two parents from the action set, and creates two new classifiers whose conditions are either a crossover of their parents conditions or a exact copy of one of the parents conditions (see (Butz, M.V. & Wilson, S.W., 2001)). The parameters of the new classifiers are also inherited from their parents. If the new classifier is a copy of one of the parents, the prediction P , the error ϵ and the action set size as are directly copied from the parent; moreover, the *fitness* of the new classifier is a discounted value of its parent fitness. If the new classifier is obtained by crossover, P , ϵ and as are averaged over their parents' values, and the *fitness* is set to a discounted value of its parents' average.

Thus, new classifiers start with a rough estimation of their parameters. Let's suppose, for example, that a new correct classifier advocating the action A is created while exploring a low rewarded niche belonging to the action B ($niche_B$). As the new classifier was created from a low rewarded niche, its *prediction* and *error* would be close to zero, its *action set size* would be set to the size of $niche_B$, and its *fitness* would be a discounted value of its parents' ones. From this first rough approximation, XCS will adjust the parameters of new classifiers on-line based on the rewards received when exploring the niche. The first update the classifier goes through, P will be set to the reward received, ϵ to the difference between the inherited reward prediction and the reward received, and as to the size of the niche activated.

The error produced in parameters' inheritance may not affect XCS in balanced or slightly imbalance problems, since all niches are frequently nourished, and so, all classifiers are updated. However, when learning from imbalanced data, some of the instances are sampled in a low frequency, and so, the correspondent niches are updated infrequently. Then, classifiers belonging to that niches, i.e., correct classifiers of the minority class, will have a wrong parameter estimation during a large number of learning iterations.

The model derived in section 3 showed that a high proportion of correct classifiers of the minority class are discovered while exploring low rewarded and numerous niches of other classes. Thus, this classifiers will be created from incorrect or overgeneral classifiers; consequently, their as will be over-estimated, and they will inherit an error from 0 (if they are created from an incorrect classifier) to $R_{max}/2$ (if they are created from an overgeneral classifier). As ir increases, these classifiers will have poor parameter estimations during a larger time, since they are updated every $\frac{1}{n \cdot (1+ir)}$.

To prevent minority class classifiers from an early deletion, we introduced two simple modifications in the procedure that sets the initial values of classifiers' parameters. First of all, we initialized the action set size to 1. In this way, we minimize the deletion probability of a correct classifier of the minority class before being updated for the first time. This modification affects also to majority class classifiers, but, as they are updated frequently, as will tend quickly to its

real value. Note that this approach may be applied cautiously in domains where classifiers that do not match any input can be created. Secondly, we initialize the error of a classifier to 0. So, we encourage new correct classifiers to get a high proportion of fitness when sharing the reward with overgeneral classifiers.

Basically, these few changes try to protect infrequently updated classifiers. While their effect would not be perceptible in balanced or slightly imbalanced datasets, it will be really important when dealing with extremely imbalanced datasets.

6.3 The Effect of Subsumption

Subsumption deletion is a mechanism introduced in (Wilson, S.W., 1998) with the aim of increasing the pressure toward the generalization of the population in order to obtain highly accurate and compact populations. The main idea behind subsumption is to delete accurate but specific classifiers if other accurate but more general classifiers exist in the population. Subsumption has been applied in both the genetic algorithm and the action set.

The subsumption in the genetic algorithm is described as follows. Every time the genetic algorithm creates a new classifier, XCS checks if either parent can subsume the new classifier. A classifier (the subsumer) can subsume another classifier (the subsumed) if the condition of the subsumer is more general than the condition of the subsumed classifier, and the subsumer has an experience higher than θ_{sub} , and error lower than ϵ_0 . If any parent can subsume any offspring, the numerosity of the parent is increased by one instead of adding the new classifier into the population. Otherwise, the new classifier is added.

The subsumption mechanism was also extended to the action set in the following way. After each XCS iteration, the current action set is searched for the most general subsumer. Then, all the other classifiers in the action set are tested against the subsumer. If a classifier is subsumed, it is removed from the population and the numerosity of the subsumer is increased.

Although subsumption is a powerful tool to obtain a highly accurate and maximally general population, if it is not adjusted properly, it may hinder XCS's performance in highly imbalanced datasets. That is, in imbalanced datasets, an overgeneral classifier predicting any class other than the majority may receive, at maximum, ir positive rewards before receiving the first negative reward. Consequently, XCS will consider these classifiers as accurate during ir iterations. If $ir > \theta_{sub}$ some correct but more specific classifiers of the majority class may be subsumed by overgeneral classifiers.

6.4 Stabilizing the Population before Testing

Finally, we are concerned about the composition of the final population. As the GA is continuously applied, new classifiers are generated until the end of the learning. Thus, there may be some classifiers in the final population that have been poorly evaluated, and so, their parameters may not be reliable. In balanced or slightly imbalanced datasets, as all niches are frequently nourished, few classifiers in the final population would have poor estimations of their parameter values. However, in highly imbalanced datasets, instances of the minority class are rarely sampled. Thus, the final population would have:

1. Minority class classifiers poorly evaluated.
2. Overgeneral classifiers predicting any class other than the minority class with over-estimated parameter values.

Classifiers of the minority class are continuously generated regardless of the imbalance ratio (see equation 11). However, these classifiers will be updated every ir learning iterations. From formula 12 we can derive the number of classifiers in the final population that would have not received any parameter update, and so, their parameters can be wrong:

$$\text{Num. not eval. classifiers} = \frac{ir}{n} \left(\frac{\mu}{2}\right)^{k_m} \quad (29)$$

Let's note that the number of non-evaluated classifiers increases linearly with the imbalance ratio. Thus, the higher the imbalance ratio, the bigger the number of classifiers in the final population that have not received any update. Besides, classifiers that have been updated poorly can also have unstable parameters.

On the other hand, overgeneral classifiers would be frequently updated. However, overgeneral classifiers that cover two classes—being one of them the minority class—would only receive a negative reward every ir iterations. Thus, they will be considered as accurate until then. In that way, in the final population there may be overgeneral classifiers considered as highly accurate, and voting consequently in the prediction array.

To avoid having classifiers with wrong parameters in the final population we introduced some extra runs in which the GA is switched off at the end of the learning process. That is, we ran XCS for some extra iterations but turning off the genetic algorithm. In that way, we could stabilize classifiers parameters and get a consistent population.

This section reviewed four different aspects that may hinder XCS to solve extremely imbalanced datasets, that is, a) type of mutation, b) parameters initialization, c) subsumption, and d) condensation of the population. We gather all the recommendations under the name of XCS with protection of the minority class (XCS+PCM). In the following section, we repeat the experimentations with the *one-bit* problem, and compare the empirical results with the theoretical model.

7 Results with XCS with Protection of the Minority Class: XCS+PMC

We repeated the experimentation with the *one-bit* problem presented in section 5, but following the recommendations given in section 6. So, niched mutation was replaced by free mutation; parameters of new classifiers were initialized as proposed in section 6.2; subsumption was configured in relation to the imbalance ratio, that is, $\theta_{sub} = ir$; and, after learning, we ran N_{conds} runs without applying the GA, where $N_{conds} = 1000 \cdot ir$.

Figure 3 shows the minimum population size required to solve the *one-bit* problem with imbalance ratios from $ir=1$ to $ir=4096$. Note that in the experiments made in section 5 XCS could only solve the problem up to $ir=1024$. The points show the empirical values for the minimum population size required at different imbalance ratios, while the theoretical bound is shown by the dashed line (the curve increases linearly with slope equal to 1).

The scaling-up of the population size shows two different behaviors. From $ir = 1$ to $ir = 128$, XCS could solve the *one-bit* problem with a constant population size ($N=35$). For higher imbalance levels, i.e., from $ir=128$ to $ir=4096$, the population size had to be slightly increased to solve the problem; for $ir=4096$, XCS succeeded with $N=50$. That is, in this second facet, the population size needs to increase linearly but with a slope close to zero.

Results obtained with the *one-bit* problem indicate that the population size bound to ensure the growth of niches of the minority class is valid but a little over-estimated. Formula 24 indicated

Scaling-up of the Population Size with the Imbalance Ratio in the One-Bit Problem

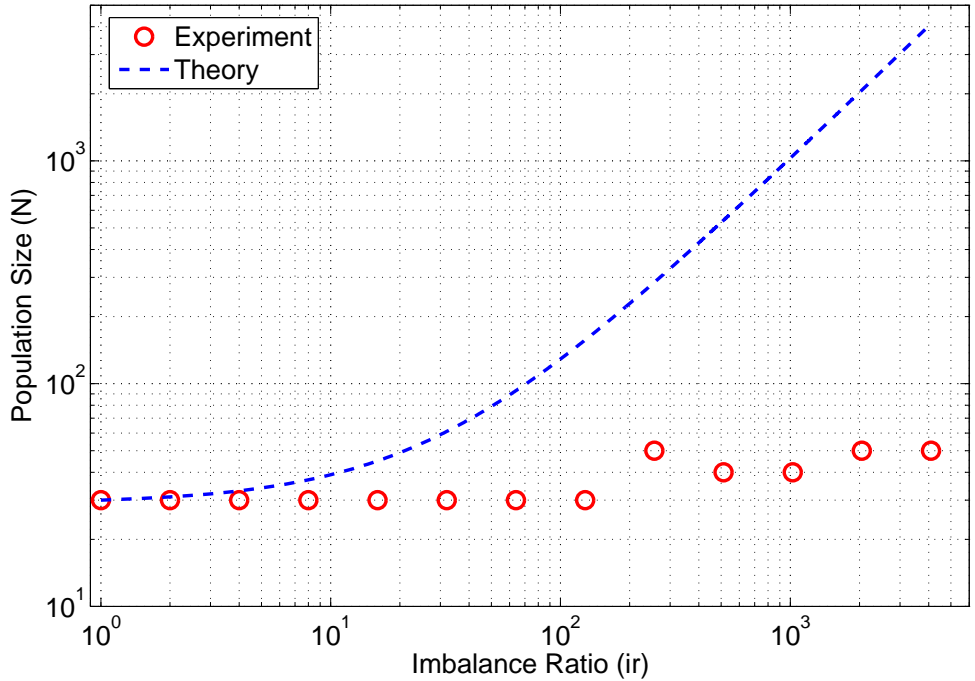


Figure 3: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *one-bit* problem with $\ell = 20$. The points show the empirical values for the minimum population size required at each imbalance level to solve the problem. To compare the empirical results with the theory, we plot a curve (in dashed line) that departs from the minimum population size required at $ir=1$ and increases linearly with slope equal to 1 with the imbalance ratio. The scaling-up of the population size shows two different facets. For the lowest imbalance ratios, the population size remains constant. For the highest imbalance levels, the population size scales linearly with the imbalance ratio, but with a slope close to zero.

that, to ensure the growth of minority class niches, population size should increase linearly with the imbalance ratio. We hypothesize that, as the *one-bit* problem is really simple, XCS can solve it even without a strong genetic pressure; it can be solved by only maintaining some correct classifiers in niches of the minority class. Under this assumption, the results obtained fit formula 18, which indicate that population size do not need to increase with the imbalance ratio to maintain correct classifiers of the minority class. Thus the empirical validate that XCS can maintain representants in niches of the minority class regardless of the imbalance ratio.

8 Increasing the Difficulty: The Imbalanced Parity Problem

In the last section we used a simple problem to demonstrate that XCS could maintain minority class classifiers at high class imbalances with very little increase of the population size. The results validated the population size bound derived in formula 18. Now, our aim is to empirically show how the population size increases in a more complex problem where it is necessary a stronger guide toward optimal classifiers. For this purpose, we chose the *parity* problem.

We ran XCS with the parity problem setting $\ell = 20$ and $k = 3$. The system was configured as

Scaling-up of the Population Size with the Imbalance Ratio in the Par17-3 Problem

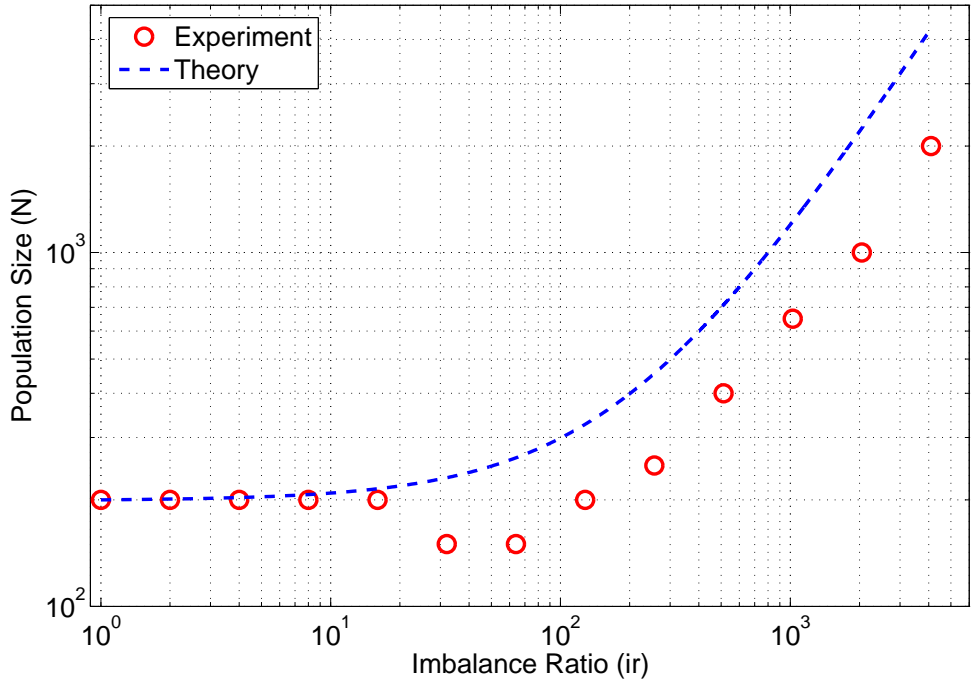


Figure 4: Scalability of the population size (N) with the imbalance ratio (ir) in the imbalanced *parity* problem with $\ell = 20$ and $k = 3$. The points show the minimum population size required at each imbalance level to solve the problem. To compare the empirical results with the theory, we plot a curve (in dashed line) that departs from the minimum population size required at $ir=1$ and increases linearly with slope equal to 1 with the imbalance ratio. The scaling-up of the population size shows two different facets. For the lowest imbalance ratios, the population size remains constant. For the highest imbalance levels, the population size scales linearly, with a slope close to 1.

described in section 7. As we covered the population with highly general classifiers ($P_{\#} = 0.8$), we needed a correct genetic pressure to drive the population from the overgeneral side to the optimum.

Figure 4 shows the minimum population size required to solve the *parity* problem for imbalance ratios up to $ir = 4096$. The points show the empirical values for the minimum population size required at different imbalance ratios, while the theoretical bound is plot in dashed line (the curve increases linearly with slope equal to 1).

As in the *one-bit* problem, the scaling-up of the population size in the *parity* problem shows to different facets. First, for low imbalance ratios—from $ir=1$ to $ir=128$ —the problem can be solved without increasing the population size. This behavior was already observed in the *one-bit* problem. Our hypothesis is that, for these low imbalance ratios, the estimation of as is accurate, and so, the population size bound is over-sized. Thus, XCS appears to be really efficient when dealing with moderate amounts of imbalance.

On the other hand, for the highest imbalance ratios, i.e., from $ir=128$ to $ir=4096$, the population size requirements increase linearly with the imbalance ratio. In this second facet, the empirical results fit the population size bound derived in formula 24, validating that the theory bounds correctly the population size scalability in imbalanced datasets.

9 Further Work

Apart from modeling the scaling-up of the population size with class imbalances, the theoretical analysis provided in this paper also served to detect different aspects that may be improved to learn from imbalanced data more efficiently. One of the most important ones is the covering operator. As detected in section 6, the probability that covering generates correct classifiers of the minority class decreases exponentially with the imbalance ratio. In this way, mutation is the main responsible for creating correct classifiers of the minority class. We identified that the time to create these classifiers depended exponentially on the order of the schema k_m . However, in some problems, k_m can also depend on the imbalance ratio (e.g., the position problem). In this case, a initial supply of correct classifiers of the minority class should be crucial to speed up the learning and decrease the population size requirements in XCS. In further work, we will investigate mechanisms to provide this initial classifiers of the minority class.

10 Summary and Conclusions

This paper investigated the capabilities of XCS to discover and maintain niches that are nourished infrequently. Firstly, we derived theoretical models that analyzed the impact of learning from datasets that contain class imbalances on different XCS mechanisms. Specifically, the theoretical models evaluated the effects of class imbalances on the covering operator, the creation and deletion of correct classifiers of the minority class, and the number of genetic opportunities that these classifiers receive. Theory indicated that XCS should be able to create minority class classifiers with a time that depended exponentially on the order of the schemas of the minority class, but it was independent of the imbalance ratio. From this, we derived a population size bound indicating that XCS could maintain niches of the minority class regardless of the imbalance ratio, but the population size needed to increase linearly with ir to ensure the growth of better classifiers in the minority class niches.

We used a facetwise approach to design two test problems that permit us to control the difficulty introduced by the imbalance ratio: the *one-bit* problem and the *parity* problem. We empirically showed that the population size in standard XCS scales exponentially with the imbalance ratio in the *one-bit* problem. We analyzed which factors caused this deviation, and detected some little drawbacks that may hinder XCS performance at high class imbalances. They were mainly related to the type of mutation, the error in the initialization of new classifiers' parameters, a strongly pressure toward generalization introduced by subsumption, and the presence of classifiers with inaccurate parameters in the final population. We proposed several methods to alleviate the effect of each problem, resulting in the system that we addressed as XCS+PMC.

XCS+PMC could solve the *one-bit* problem up to $ir=4092$ with a population size nearly constant. Although solving a problem as the *one-bit* could seem to be a trivial task for a learner, the results obtained are really important since they demonstrate that, if genetic pressures lead to the creation of classifiers that represent a niche of the minority class, XCS will be able to maintain this niche. Furthermore, we introduced a more complex problem in the analysis, the *imbalanced parity* problem. The motivation to introduce this problem was to validate the population size bound in a problem where the GA needed to guide strongly toward the optimum. The results obtained showed again two different facets. For low imbalance ratios, the population size remained constant. For high class imbalances, the population size increased linearly, showing that the theory predicts correctly the scaling-up of the population size with the imbalance ratio.

Acknowledgements

The authors thank Pier Luca Lanzi for the feedback he provided throughout this research. We also thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, as well as the support of *Ministerio de Ciencia y Tecnología* under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* under Grants 2005FI-00252 and 2005SGR-00302.

This work was also sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, the National Science Foundation under grant DMR-03-25939 at the Materials Computation Center. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

- Bacardit, J., & Butz, M.V. (2004). *Data Mining in Learning Classifier Systems: Comparing XCS with GAssist*. Springer-Verlag.
- Bernadó-Mansilla, E., Llorà, X., & Garrell, J.M. (2002). XCS and GALE: a Comparative Study of Two Learning Classifier Systems on Data Mining. In Lanzi, P., Stolzmann, W., & Wilson, S. (Eds.), *Advances in Learning Classifier Systems, 4th International Workshop*, Volume 2321 of *Lecture Notes in Artificial Intelligence* (pp. 115–132). Springer.
- Butz, M.V. (2004). *Rule-based Evolutionary Online Learning Systems: Learning Bounds, Classification and Prediction*. Doctoral dissertation, Illinois Genetic Algorithms Laboratory (ILLI-GAL) - University of Illinois at Urbana Champaign.
- Butz, M.V., Sastry, K., & Goldberg, D.E. (2003). Tournament Selection in XCS. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)* (pp. 1857–1869). Springer Verlag, Berlin.
- Butz, M.V., & Wilson, S.W. (2001). An algorithmic description of XCS. In Lanzi, P., Stolzmann, W., & Wilson, S. (Eds.), *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, Volume 1996 of *Lecture Notes in Artificial Intelligence* (pp. 253–272). Springer.
- Goldberg, D.E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms* (1 ed.). Kluwer Academic Publishers.
- Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.
- Holmes, J.H. (1998). Differential Negative Reinforcement Improves Classifier System Learning Rate in two-class Problems with Unequal Base Rates. In *Genetic Programming 1998: Proceedings of the Third Annual Conference* (pp. 635–642). Morgan Kaufmann.
- Kovacs, T., & Kerber, M. (2001). What Makes a Problem Hard for XCS. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds.), *Advances in Learning Classifier Systems: Third International Workshop, IWLCS* (pp. 80–99). Springer-Verlag.
- Lanzi, P. L., & Wilson, S. W. (2000). Toward Optimal Classifier System Performance in Non-Markov Environments. *Evolutionary Computation*, 8(4), 394–418.

- Orriols-Puig, A., & Bernadó-Mansilla, E. (2006). Bounding XCS Parameters for Unbalanced Datasets. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 1561–1568). New York, NY, USA: ACM Press.
- Orriols-Puig, A., & Bernadó-Mansilla, E. (InPress). The Class Imbalance Problem in UCS Classifier System: A Preliminary Study. Springer.
- Wilson, S.W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2), 149–175.
- Wilson, S.W. (1998). Generalization in the XCS Classifier System. In Koza, J., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D., Garzon, M., Goldberg, D., Iba, H., & Riolo, R. (Eds.), *Genetic Programming: Proceedings of the Third Annual Conference* (pp. 665–674). Morgan Kaufmann.
- Wilson, S.W. (2002). Classifiers that Approximate Functions. *Journal of Natural Computing*, 1(2), 211–234.