Facetwise Analysis of XCS for Problems with Class Imbalances

Albert Orriols-Puig, Ester Bernadó-Mansilla, David E. Goldberg, Kumara Sastry, and Pier Luca Lanzi

.

Abstract

Michigan-style learning classifier systems (LCSs) are online machine learning techniques that incrementally evolve distributed sub-solutions that solve a portion of the problem space. As in many machine learning systems, extracting accurate models from problems with class imbalances—that is, problems where one of the classes is poorly represented with respect to the other classes—has been identified as a key challenge in LCSs. Empirical studies have shown that Michigan-style LCSs fail to provide accurate sub-solutions that represent the minority class in domains with moderate and large disproportion of examples per class; however, analyses of the causes of this failure are lacking. Therefore, the aim of this paper is to carefully examine the effect of class imbalances on different LCSs components. The analysis is focused on XCS, the most-relevant Michigan-style LCS, although the models could be easily adapted to other LCSs. Design decomposition is used to identify five elements that are critical to guarantee the success of LCSs on imbalanced domains, and facetwise models that explain these different elements for XCS are developed. All theoretical models are validated with artificial problems. The integration of all these models permits us to identify the sweet spot where XCS will be able to scalably and efficiently evolve accurate models of rare classes; besides, the facetwise analysis is used as a tool for designing a set of configuration guidelines that have to be followed to ensure convergence. With a proper configuration, XCS is shown to be able to solve highly imbalanced problems that previously eluded solution.

This work was supported by the Ministerio de Educación y Ciencia under Projects TIN2005-08386-C05-04 and by 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 FA9550-06-1-0370, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center and under grant ISS-02-09199 at the National Center for Supercomputing Applications, UIUC. 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.

Albert Orriols-Puig and Ester Bernadó-Mansilla are with the Grup de Recerca en Sistemes Intel·ligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 08022 Barcelona, (Spain) (e-mail: {aorriols,esterb}@salle.url.edu).

David E. Goldberg and Kumara Sastry are with the Illinois Genetic Algorithm Laboratory (IlliGAL), Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (USA). Kumara Sastry is currently at Intel. (e-mail: deg@uiuc.edu, kumara@kumarasastry.com).

Pier Luca Lanzi is with the Artificial Intelligence Robotics Laboratory (AIRLab), Dip. di Elettronica e Informazione, Politecnico di Milano, P.za L. da Vinci 32, I-20133, Milano (Italy) (e-mail: lanzi@elet.polimi.it).

Manuscript received MONTH XX, XXXX; revised MONTH XX, XXXX.

Index Terms

Genetic algorithms, online learning, learning classifier systems, class imbalance problem, facetwise modeling, patchquilt integration.

I. INTRODUCTION

During the last few years, Michigan-style learning classifier systems (LCSs) [1]—genetic-based machine learning (GBML) methods that combine credit apportionment techniques [2] (now recognized as reinforcement learning techniques [3]) and genetic algorithms [4], [5] to evolve a population of rules online—have been enjoying a renaissance. Ad hoc formulation of first generation systems, there have been crucial advances in (1) systematic design of new competent LCSs [6], [7], (2) applications in important domains [8]–[11], and (3) theoretical analyses for design [12]–[14]. Despite these favorable occurrences, there still remain difficult challenges that need to be addressed to scalably and efficiently solve real-world problems. A particular important challenge, shared by traditional machine learning techniques and GBML systems alike, is learning from domains that contain class imbalances—that is, domains where one of the classes is under-sampled with respect to the other classes. Learning from rare classes is a critical aspect since the key knowledge most often resides in the minority class, and it has been shown that many traditional learning techniques are not able to extract accurate models from rare classes [15]. Therefore, the machine learning community has recently started to give importance to develop new approaches to tackle class imbalanced problems [16]. Nonetheless, this aspect has been largely overlooked in online learning architectures; learning from imbalanced domains poses still more challenges to online learning systems, since the learner receives a stream of examples and has to learn accurate models for the upcoming rare classes on the fly.

The purpose of this paper is to study the behavior of LCSs on imbalanced domains and take the lessons provided by the analysis to improve the modeling of rare classes. Our analysis is centered on XCS [6], [7], the most relevant Michigan-style LCSs. Nonetheless, the study is methodologically developed so that it could entirely—or with few changes—be applied to other Michigan-style LCSs. Due to the complexity of the LCSs architecture, we *decompose* the problem of learning from imbalanced domains in several elements and derive *facetwise models* [17] for each one of them. The integration of these models permits us to detect several crucial conditions that need to be satisfied to ensure that the system would extract the key knowledge from rare classes; besides, we also identify *critical bounds* on the system behavior. The lessons learned from this analysis result in several configuration recommendations that, when followed, enable XCS to solve highly imbalanced classification problems that previously eluded solution.

The remainder of this paper is organized as follows. Section II presents the class imbalance problem, reviews how the machine learning community has faced the problem with offline learning techniques, and places the problem in the context of online learning. Section III briefly explains XCS and intuitively discusses the problems that learning from imbalanced domains may pose to the system. Section IV introduces the *facetwise analysis* methodology and specifies the steps that we follow in the present analysis. In Sections V, VI, VII, VIII, and IX, we derive the different facetwise models. Section X integrates all the facetwise models, highlights the lessons learned along the

analysis, and uses them to solve the 11-bit multiplexer problem [6] with large amounts of class imbalance, which previously eluded solution. Finally, Section XI summarizes and concludes the work.

II. The Challenges of Learning from Imbalanced Domains in Machine Learning

During the last few decades, the increasing research on machine learning has led to the application of several learning techniques to real-world problems with the aim of extracting novel, interesting, and useful knowledge from these domains [18]. One of the main characteristics of real-world problems is that some of the sub-concepts or classes may be poorly represented in the training data set due to either the scarcity of these concepts in nature or the cost—or inadequacy of the used techniques—to extract positive samples that represent these concepts. The purpose of this section is to highlight the importance of extracting accurate models from these rare concepts or classes in machine learning tasks. We begin emphasizing the high number of real-world applications in which we have class imbalances. Then, we briefly review how the machine learning community in general, and the GBML field in particular, has approached this problem, accentuating the differences in the challenges that learning from rare classes poses to offline learners and to online systems.

Examples of problems that contain rare classes abound in literature and include identifying fraudulent credit card transactions [19] and predicting telecommunication equipment failures [20] among others. In these domains, while regular occurring patterns can be modeled easily, learners tend to fail to extract accurate models from the rare classes [15], [21], [22]. Nonetheless, these rare classes are of primary interest, since they most often contain key knowledge. For this reason, recently, a lot of research has been conducted on designing new approaches, or modifying existing machine learning techniques, with the aim of creating models that represent the rare classes more accurately [23], [24]. All these approaches have been designed for offline supervised learning techniques, i.e., methods that learn from static data sets which are provided at the beginning of the learning process. In online learning, knowledge acquisition from imbalanced domains poses more severe challenges, since systems receive instances and rare classes have to be detected on the fly.

The many different approaches that have been designed to enhance the discovery of useful models of rare classes in offline learning can be grouped in methods working at (i) the learner level or at (ii) the sampling level. Learner-level methods usually modify the error calculation of an existing system by either introducing a more appropriate inductive bias [25] or assigning a misclassification cost per class [26]. Also, there are some learning systems expressly developed to learn from rare classes [27]. The main drawback of the methods that work at the learner level is that they are designed for specific learning algorithms, and so, they cannot be adapted to other learning techniques in a straightforward manner. For this reason, sampling-level methods have received much more attention than learner-level approaches. Sampling-level methods, usually known as *re-sampling techniques*, eliminate rare classes by balancing the proportion of examples per class of the training data set. As they are data-preprocessing methods, they can be generally used for any learner. Notwithstanding, note that these methods can only be applied to static data sets. Many works have shown the benefits of re-sampling the training data sets in some specific imbalanced problems [21], [28]–[30]. Nonetheless, as pointed out in [31], these techniques should be cautiously

used since they change the distribution of the training instances, and so, they may also alter the geometry of the domain to be learned.

While the class imbalance problem has been extensively analyzed for classical machine learning techniques that learn from static data sets, analyses and new approximations to overcome the problem in online learning are scarce. The typical approaches designed for offline learning to overcome the class imbalance problem can barely be applied to online learning schemes, since they need to know the distribution of examples in the training data set. That is, in online learning, strategies such as over-sampling the occurrence of instances of the minority class or introducing a higher misclassification cost for minority class instances are impractical solutions since, as we do not have a static data set, we can neither estimate the imbalance ratio to re-balance the training data nor assign a misclassification cost per class to favor the rare classes. Therefore, models for the minority class have to be learned online from a set of samples that come infrequently. This is the case of Michigan-style LCSs. Although some ad hoc strategies have been proposed to alleviate this problem in particular LCSs [32]–[34], these strategies cannot be generalized. In general, these approaches have shown to improve LCSs performance on imbalanced domains. However, it is not clear how they affect the learning processes of LCSs; besides, they do not provide further explanation of which are the real problems that imbalanced domains pose to LCSs. Thence, in this paper, we propose a systematic analysis that explains the capabilities of the online learning architecture of LCS—specifically we focus our analysis on XCS [6], [7]—to extract useful information from instances that come very infrequently. Furthermore, the study shows the limitations of the system when learning from rare classes. Before proceeding with the analysis, the next section provides a brief description of XCS. Then, the analysis methodology is introduced and all the study is developed in the subsequent sections.

III. The XCS Classifier System

XCS is an online, model-free learning classifier system originally introduced by Wilson in 1995 [6] and further improved in [7]. XCS is probably the most popular Michigan-style LCSs; for this reason, we adopt it in our theoretical and experimental studies. The purpose of this section is to provide a brief description of the system for classification tasks. As follows, we first present the knowledge representation used by XCS and then introduce the online learning architecture of the system, which includes the process organization or learning interaction, the rule evaluation system, the rule discovery component, and the reasoning mechanism used to predict the class of test instances. Finally, we give an intuitive idea of the difficulties that XCS may have to learn from rare classes and provide some notation that will be used in the remaining of the present analysis. The user is referred to [6], [7] for further details on XCS and to [35] for an algorithmic description of the system.

A. Knowledge Representation

We first we introduce the rule-based representation of XCS and explain the most relevant parameters that are associated to each rule. XCS evolves a distributed knowledge represented by a *population* [P] of *classifiers*, where each classifier has a rule and a set of parameters that estimate the quality of the rule. A rule consists of a condition,

which specifies when the classifier is applicable, and an action, which determines the predicted action or class. For binary inputs, the condition is usually represented by the ternary alphabet $\{0, 1, \#\}^{\ell}$, where ℓ is the number of input variables. The *don't care* symbol # allows for rule generalization; that is, # indicates that the given variable matches any input value.

The most important parameters associated to a classifier are: (1) the payoff prediction p, an estimate of the payoff that the classifier will receive if its condition matches and its action is chosen; (2) the prediction error ϵ, an estimate of the average error between the classifier prediction and the received payoff; (3) the fitness F, an estimate of the accuracy of the payoff prediction; (4) the action set size as, an estimate of the size of the action sets in which the classifier has participated; and (5) the *numerosity* n, which indicates the number of copies of the classifier in the population.

To completely understand the knowledge representation, in the following sections we detail how the different components of XCS interact to evaluate the existing classifiers and create new promising classifiers.

B. Learning Interaction

XCS learns online by interacting with an environment which provides a new training example at each iteration. This section discusses how XCS learns from the interaction with this environment.

In *exploration* or training mode, XCS is provided with a new example e at each learning iteration. Then, the system builds a *match set* [M] containing all the classifiers in [P] whose conditions match e. If the number of classes represented in [M] is less than the threshold θ_{mna} (θ_{mna} is usually set to the total number of possible classes of the problem), the *covering* operator is triggered, creating as many new classifiers as required to cover θ_{mna} different classes. The condition of the new classifiers created by covering is generalized from e. That is, each variable is set to # with probability $P_{\#}$ (where $P_{\#}$ is a configuration parameter); otherwise, the variable takes the corresponding value in e. The class of the new classifier is randomly selected among the classes that are not covered in [M].

Next, the system computes the *system prediction* $P(c_i)$ for each possible class, which estimates the payoff that the system will receive if c_i is selected as output. $P(c_i)$ is calculated as the fitness weighted average of the predictions of the classifiers in [M] that advocate class c_i. Then, one of the classes is randomly selected. Thus, XCS explores the consequences of all classes for each possible input. The chosen class determines the action set [A], which consists of all classifiers in [M] advocating that class. The action set works as a *niche* where the parameters update procedure and the genetic algorithm take place. The next subsections explicate these two procedures in detail.

C. Classifier Evaluation

In training mode, after XCS sends the chosen class to the environment, a reward R is returned. R is maximal if the proposed class is the same as the training example (usually 1000), and minimal (usually zero) otherwise. As proceeds, we detail how the system uses this reward to update the parameters of the classifiers in [A].

The prediction of each classifier is first updated according to the Widrow-Hoff rule [36] as $p \leftarrow p + \beta(R - p)$, where β ($0 < \beta \leq 1$) is the learning rate. Next, the prediction error ϵ is computed as $\epsilon \leftarrow \epsilon + \beta(|R - p| - \epsilon)$. Then,

the fitness is updated as follows. First, the *accuracy* κ of each classifier in [A] is calculated as

$$\kappa = \begin{cases} \alpha(\epsilon/\epsilon_0)^{-\nu} & \epsilon \geq \epsilon_0; \\ \\ 1 & \text{otherwise.} \end{cases} \quad (1)$$

κ is used to compute the *relative accuracy* κ' as $\kappa' = \frac{\kappa \cdot n}{\sum_{cl \in [A]} \kappa_{cl} \cdot n_{cl}}$, which is used to update the fitness as $F = F + \beta(\kappa' - F)$. Thus, the fitness is an estimate of the accuracy of the classifier prediction relative to the accuracies of the overlapping classifiers. This provides fitness sharing among the classifiers belonging to the same action set. Finally, the action set size is updated as $as = as + \beta(|[A]| - as)$, where $|[A]|$ is the size of the current action set. Once the parameters of the classifiers in [A] have been evaluated, the GA can be applied to the current niche, which follows the process explained in the next section.

D. Classifier Discovery

XCS uses a steady-state niche-based *genetic algorithm* (GA) [5] to discover new promising rules. The GA is triggered in the current action set if the average time since its last application to the classifiers in [A] is greater than θ_{GA}. Here, we explain the basic mechanisms of the GA.

The GA selects two parents from the current [A] following either proportionate selection [6] or tournament selection [37] and copies them. The copies undergo crossover with probability χ and mutation with probability μ per allele. Crossover randomly selects two cut points and mixes the information of the two classifiers. If crossover, is not applied, the offspring are exact copies of the parents. Mutation randomly decides whether each variable of the rule has to be changed; in this case, it randomly chooses a new value for the variable. The class of the rule also undergoes the same process.

The resulting offspring are introduced into the population via subsumption [7]. That is, if there exists a sufficiently experienced ($exp > \theta_{sub}$) and accurate ($\epsilon < \epsilon_0$) classifier in [A] whose condition is more general than the new offspring, the numerosity of this classifier is increased. Otherwise, the new offspring is introduced into the population. Two classifiers are removed if the population is full. The deletion probability of a classifier is proportional to the action set size estimate of the classifier; moreover, the deletion probability is increased if the classifier is experienced enough ($exp > \theta_{del}$) and its fitness F is smaller than a proportion of the average fitness of the population $\overline{F}$ ($F < \delta\overline{F}$) [38]. This biases the search toward highly fit classifiers and, at the same time, balances the classifiers' allocation in the different action sets.

Once the learning finishes, the evolved population is used to infer the class of new unlabeled instances. The next section provides the reasoning mechanism used by XCS.

E. Class Inference in Test Mode

Once XCS has evolved a population of highly general and accurate rules, this population is used to infer the class of new examples. The reasoning mechanism combines the information that resides in the different matching rules as follows. Firstly, XCS creates [M] with all the matching classifiers; covering is not applied in any case.

Then, the prediction array is formed as explained in Section III-B, and the most voted action is returned as output. No further actions are taken. Note that during test, the population is never modified.

In summary, XCS is an online system which represents individuals as classifiers that contain single rules, uses adapted reinforcement learning techniques to evaluate the quality of these classifiers, employs a GA to discover new promising rules, and uses a fitness-based voting policy to infer the class of test instances. XCS process organization is based on the activation of classifiers to form match sets and action sets. Therefore, rare classes may pose a problem to XCS since the instances that belong to these classes are sampled less frequently. The next section elaborates this aspect in more detail.

F. Intuition of XCS Difficulties to Learn from Imbalanced Domains

After describing how XCS works, here we intuitively analyze the difficulties that learning from imbalanced domains may pose to XCS. For this purpose, we first define the concepts of *problem niche* and *representative* of a niche and then review the online learning architecture identifying the components that may impair the discovery of the minority class.

XCS evolves a distributed set of sub-solutions. In the remaining of this analysis, we use the term *problem niche* (or simply *niche*) to refer to a problem subspace where a maximally general sub-solution applies [1]. Each niche is represented by a *schema* [4], which defines the value of the relevant variables of the given problem niche. The order o of the schema is the number of relevant variables of the niche. Then, we define that a *representative* of a niche is a classifier whose condition specifies the o relevant variables of the niche schema correctly and that predicts the class or action of the sub-solution that the niche represents. Oppositely, classifiers that do not match any problem schema and cover examples of different classes are referred to as *over-general classifiers*.

For instance, let us suppose that we have a niche represented by the schema 01*0** and whose class is 0. That means that all the examples matching 01*0** belong to class 0. Generalizing any of the o specific bits of the schema will cause that the schema matches instances of other classes, thence, do not representing an accurate sub-solution anymore. Any classifier whose condition has the same value for the o specific bits is a representative of the niche. For example, classifiers 01#0##:0, 0110##:0, and 010010:0 are representatives of the niche. The maximally general representative of a niche is the classifier that only fixes de o relevant bits of the schema and predicts the action of the sub-solution, i.e., 01#0##:0. An example of an over-general classifier is #1#0##:0, since it generalizes the first fixed bit of the schema.

Provided these definitions, we now intuitively analyze the possible problems that XCS may need to face when learning from imbalanced data by reviewing how the online architecture works. In the beginning of the learning process, the covering operator initializes the population with a set of classifiers that are generalized from the first sampled input instances. Therefore, the initial population mostly consists of over-general classifiers. Then, the evolutionary pressures tend to drive the system toward evolving *complete* and *consistent* rule sets that are minimal

[1] In XCS terms, an action set represents a niche.

representations of the target concept [7], [13]. This is mainly caused by the interaction of two evolutionary pressures: (i) the *accuracy pressure* and (ii) the *generalization pressure*. The accuracy pressure is due to the selection of the fittest individuals. It moves the search toward accurate rules, thus removing over-general classifiers in favor of accurate representatives of the different niches. Generalization pressure is due to the niche-based reproduction, which favors classifiers that are frequently active, and to the population-based deletion, which deletes from the whole population with the aim of balancing the number of classifiers allocated in the different niches. Therefore, XCS is designed to evolve accurate representatives of niches that occur sufficiently frequently.

Several studies have derived theory that support this intuition (e.g., see [12]) and empirical analyses have shown that XCS is able to evolve accurate solutions for problems in which the number of examples per class is approximately the same [13], [39]. Nevertheless, imbalanced data sets are characterized for having a low proportion of examples of one of the classes. That means that niches that represent the instances of different classes are activated with different frequencies. In the remaining of this analysis, we address the niches that are activated by instances of the majority class as *nourished niches*. Oppositely, niches activated by instances of the minority class are referred to as *starved niches*. We also define the *imbalance ratio* ir as the number of examples of the majority class with respect to the number of instances of the minority class that are sampled to the system, which also indicates the disproportion between the number of genetic opportunities that nourished niches and starved niches receive. Due to the occurrence-based reproduction of XCS, intuition seems to indicate that the system may suffer to discover representatives of starved niches, since these niches will be infrequently activated with respect to the other niches of the system, especially as ir increases. Hence, this low number of genetic opportunities combined with the evolutionary pressures—which promote the classifiers that are more frequently activated—may discourage XCS from evolving representatives of starved niches when competing with over-general classifiers and representatives of nourished niches, which are more frequently activated. In our study, we are interested in systematically analyzing the difficulties that XCS may face as the imbalance ratio increases and providing solutions to let the system learn from highly imbalanced domains. In the next section, we first introduce the facetwise methodology and then detail this systematic analysis that will be conducted in the subsequent sections.

IV. Facetwise Analysis of XCS in Imbalanced Domains

In this section, we follow a design decomposition approach to systematically analyze the different sources of difficulty that XCS may find when learning from imbalanced domains. We first briefly introduce the design decomposition methodology adopted by Goldberg [17] as a design approach to competent genetic algorithms and review how it has been transported to XCS. Then, we follow this approach to decompose the problem of learning from imbalanced domains in XCS.

A. Design Decomposition in Genetic Algorithms

Goldberg [17] emphasizes the relevance of *design decomposition* and *facetwise analysis* for advancing in the design and the understanding of complex systems. The design decomposition methodology decomposes the working

process of complex systems into different elements, and each one of these elements is analyzed separately assuming that all the others are behaving in an ideal manner. All these models provide key insights into the working of the complex system, increasing our understanding of the underlying processes of the system; furthermore, they also can be used as a tool for designing new competent and efficient complex systems that satisfy the requirements identified by the different models. Besides, the individual facetwise models can be combined, identifying the sweet spot where the algorithm actually scales.

In [17], the design decomposition approach was followed to design *competent genetic algorithms*. Taking as a point of departure the original Holland's notion of building block (BB) [4], Goldberg proposed to decompose the problem of designing *selectorecombinative GAs* into the following seven subproblems:

1) Know that GAs process BBs.
2) Know the BB challengers.
3) Ensure adequate supply of raw BB.
4) Ensure increased market share for superior BBs.
5) Know BB takeover and convergence times.
6) Make decision well among competing BBs.
7) Mix BBs well.

In brief, the primary idea of this theory is that GAs work through a mechanism of *decomposition*—by identifying the BBs or sub-assemblies of the problem—and *reassembly*—by recombining different sub-assemblies to form very high performance solutions. Therefore, the first subproblem is to identify that GAs process these BBs and to analyze which types of BBs may be challenging to acquire by a GA. Once these key concepts are identified, the theory analyzes how these BBs evolve in a *market economy of ideas*. That is, we need to ensure that the market is provided with enough stock of BBs, that the good BBs grow in the market share, that good decisions are made among them, and that they are exchanged well. Facetwise models that explain each one of these elements were developed in Golberg's work.

As described in the previous section, similarly to GAs, LCSs are complex systems in which several components interact to evolve a set of maximally general and accurate rules. Due to this complexity, recently, efforts have been made to apply the design decomposition and the facetwise analysis methodology to LCSs. In the next section, the existing work on carrying over the design decomposition approach to LCSs domain is briefly reviewed.

B. Carrying over Facetwise Analysis to XCS

The application of facetwise modeling to LCSs, and specifically XCS, has provided key insights into XCS working processes, enabling the solution of more complex problems. As follows, we review these analyses in more detail.

Butz et al. [12] studied the learning pressures in XCS and derived critical bounds on the system convergence. To tackle the whole problem, learning pressures were decomposed in four basic pressures: (1) set pressure, (2) deletion pressure, (3) mutation pressure, and (4) subsumption pressure. Simpler, tractable models were developed for some of these pressures, and qualitative arguments were used to connect the different models. Moreover, the authors also

derived two boundaries beyond which the convergence of XCS could not be guaranteed, which were addressed as the *covering challenge* and the *schema challenge*. Continuing the analysis of the different pressures of XCS, Butz et al. [40] analyzed the fitness pressure and derived the so-called *reproductive opportunity bound*, which sets the population size required to warrant that a classifier, with a certain specificity, will have reproductive opportunities. Later, in [41], the previous study was complemented by deriving models of the learning time in XCS. The models were simplified by not considering crossover and by assuming a domino-convergence model [42]. More recently, Markov chain analysis of niche support in XCS was performed in [14]. The analysis showed that the number of classifiers of a niche followed a binomial distribution, which yielded another population size bound to ensure effective problem sustenance.

However, these facetwise models do not explain all the aspects of XCS. Whereas these models have provided considerable insights and increased our understanding of XCS, they do not fully capture the effect of dealing with problems that contain class imbalances. Hence, in this paper, we follow a design decomposition methodology to study whether XCS can efficiently deal with rare classes. As follows, we first present an artificial problem that will enable us to identify the different difficulties that we may face when learning from imbalanced domains. Later, we present the problem decomposition.

C. A Boundedly-Difficult Problem: The Imbalanced Parity Problem

The first step before proceeding to the facetwise analysis is to design a proper test problem that highlights the difficulties that are to be studied and that serves to validate the developed models. Following this analogy, we designed the imbalanced parity problem, whose description is provided as follows.

The imbalanced parity problem extends the parity problem [39] by introducing a parameter that permits to control de complexity along the imbalance dimension. The problem is defined as follows. Given a binary string of length ℓ, where there are k relevant bits ($0 < k \leq \ell$), the output is the number of one-valued bits in the k relevant bits modulo two. This corresponds to the original definition of the parity problem. We introduce the imbalance complexity to this definition by starving the class labeled as '1'. ir denotes the ratio of examples of the majority class to the number of instances of the minority class. For $ir = 1$, the problem has, approximately, the same number of instances per class. For $ir > 1$, there are ir instances of the majority class for each instance of the minority class. Independent of the imbalance ratio, the optimal population for this problem consists of 2^{k+1} classifiers that have specific values for the k relevant attributes and all the remaining attributes are set to '#' [39].

Note that the complexity of the problem can be moved along two dimensions: the building block size k and the imbalance ratio ir. Larger values of k pose more challenges to XCS, since the system needs to discover larger, more complex building blocks whose bits have to be processed together. Moreover, k also defines the number of irrelevant attributes which XCS must generalize to obtain the optimal population. On the other hand, increasing ir implies a progressive undersampling of instances of the minority class, what may hinder XCS to discover optimal classifiers for this class.

Having defined the parity problem and analyzed the possible sources of complexity of the problem, the next section introduces the particular decomposition proposed for LCSs.

D. Design Decomposition in XCS for Learning from Imbalanced Domains

With the intuitive difficulties of XCS to learn from rare classes discussed in Section III-F and adhering to the design decomposition methodology proposed by Goldberg, we are now in position to articulate the different elements that need to be satisfied to successfully deal with problems that contain class imbalances. Our major concern is to guarantee that representatives of starved niches will win in competition with over-general classifiers as the imbalance ratio increases. Thus, we decompose the problem and consider all the facets that are likely to be affected by the dearth of examples of the minority class. Then, we derive facetwise models that model each one of the subproblems. More specifically, we consider the following five subproblems:

1) Estimate the classifiers parameters correctly—prediction, error, and fitness of classifiers.
2) Analyze whether representatives of starved niches can be provided in initialization.
3) Ensure the generation and growth of representatives of starved niches.
4) Adjust the GA application rate.
5) Ensure that representatives of starved niches will take over their niches.

Estimate the classifiers parameters correctly. The primary factor that needs to be satisfied is that the classifier evaluation procedure of XCS obtains accurate estimates of the parameters of all classifiers, and especially of over-general classifiers. In XCS, classifier parameters are updated online according to the reward received at each time step by means of the Widrow-Hoff rule [36]. This method makes a temporal windowed average of the received rewards, which gives more importance to the last received rewards as opposed to the older rewards. Consequently, the dearth of sampling of examples that represent one of the classes may cause poor estimations in over-general classifiers, since infrequent negative rewards can be forgotten. This aspect is really important since it may completely mislead the genetic search. That is, if the error of over-general classifiers is underestimated, XCS may promote these over-general classifiers when they are competing with accurate classifiers in the same niche. We investigate the accuracy of the parameter update procedure and provide some alternative parameter evaluation methods in Section V.

Analyze whether representatives of starved niches can be provided in initialization. Once ensuring that the parameters of classifiers are properly evaluated, we have to analyze whether XCS initialization process can supply the initial population with classifiers that contain schemas of starved niches in the beginning of the run. XCS starts with an empty population. Then, the covering operator is applied in the first iterations of the learning process, providing classifiers whose conditions are generalized from the first instances that are sampled to the system. Covering should initialize the population with several classifiers that, although not being maximally accurate, contain schemas that represent niches of different classes. Nonetheless, intuition seems to indicate that, for highly imbalanced domains, covering is mainly triggered on instances of the majority class; therefore, covering may fail to provide schemas of

niches that represent the minority class, i.e., starved niches. Furthermore, this problem is even more severe as the schema of starved niches gets larger (in the parity problem, this translates to have larger values of k). In Section VI, we derive models that formally explain this behavior. The remainder of the analysis is derived assuming a covering failure.

Ensure the generation and growth of representatives of starved niches. After the population is initialized, the genetic algorithm drives the search toward more accurate and general classifiers. We already appealed to the intuition that the occurrence-based reproduction of XCS may hamper the discovery of maximally general and accurate classifiers for starved niches. In Section VII, we derive facetwise models that systematically analyze the effect of class imbalances in the (i) generation and (ii) growth of representatives of starved niches. Consequently, we derive bounds on the population size to guarantee that XCS will be able to learn classifiers that belong to starved niches. All the analysis is performed assuming that the GA is applied at each learning iteration.

Adjust the GA application rate. Having ensured the discovery of representatives of starved niches, Section VIII introduces the frequency of application of the GA into the analysis and theoretically shows that decreasing the frequency of application of the GA results in a counter-balancing effect that may help XCS discover representatives of starved niches.

Ensure that representatives of starved niches will take over their niches. Discovering the first representatives of starved niches is not enough. That is, once discovered, the accurate representatives of starved niches should take over their niches, removing competing over-general, less accurate classifiers. Nonetheless, the effect of the occurrence-based reproduction may produce the opposite effect, resulting in situations where over-general classifiers take over starved niches. In Section IX, we study the takeover time of the best representative in starved niches, following the methodology used in the genetic algorithms literature for these types of analyses [43]. Takeover time expressions are derived for the two most-used selection techniques in XCS: proportionate selection [6] and tournament selection [37]. Moreover, conditions for starved niches extinction, i.e., deletion of the best representatives of starved niches in favor of over-general classifiers, are also derived for both selection schemes.

In the remainder of this paper, each of these facets is covered in a different section in which the technical argument will be developed more completely. Moreover, all the models are experimentally validated with the *imbalanced parity problem*. Then, in Section X, we unify all the models, emphasizing the lessons derived from each one. Specifically, the patchquilt integration of the models results in a domain of competence of XCS for imbalanced domains, which (i) determines under which conditions and imbalance ratios the system will be able to successfully represent the minority class in the evolved model and (ii) provides guidelines of how to set the system for different imbalance ratios. The insights supplied by these models are crucial to increase the understanding of how XCS evolves classifiers that represent starved niches, enabling the solution of highly imbalanced problems that previously eluded solution. We show, as example, that all the acquired knowledge enables us to appropriately set XCS so that it is able to solve the multiplexer problem [6] with large amounts of class imbalances.

V. Estimation of Classifier Parameters

In this section, we analyze whether Widrow-Hoff rule provides accurate estimates of the parameters of over-general classifiers as the imbalance ratio increases. We especially focus on the *prediction error* of over-general classifiers, since it determines classifier's fitness. If the prediction error is underestimated, over-general classifiers may be considered as accurate classifiers by the system; hence, they will win in competition with more specific but accurate classifiers. According to [44], the prediction error of over-general classifiers should be

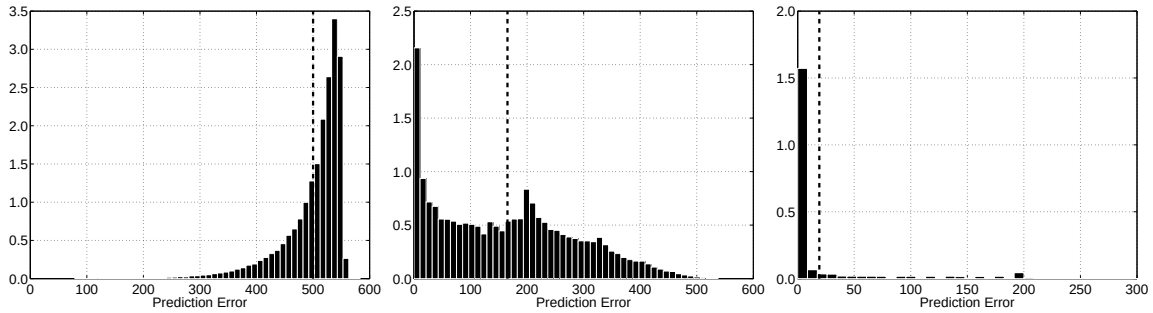
$$\epsilon = \frac{2R_{max} \cdot ir}{(1+ir)^2}, \tag{2}$$

where R_{max} is the maximum reward given by the environment (in our experiments $R_{max} = 1000$).

Here, we empirically analyze if XCS can obtain reliable estimates as ir increases. For this purpose, we ran the imbalanced parity problem with $\ell = 11, k = 4$, and $ir = \{1, 10, 100\}$. We initialized XCS with the optimal population plus the most over-general classifier predicting class 0 (i.e., `###########:0`) and deactivated the GA. We set $\beta = 0.2$, which is a typical value used in the literature. Fig. 1 shows a histogram of the error estimate of the most over-general classifier along a complete run. Results are averages over 10 runs. The vertical line shows the theoretical value for each case.

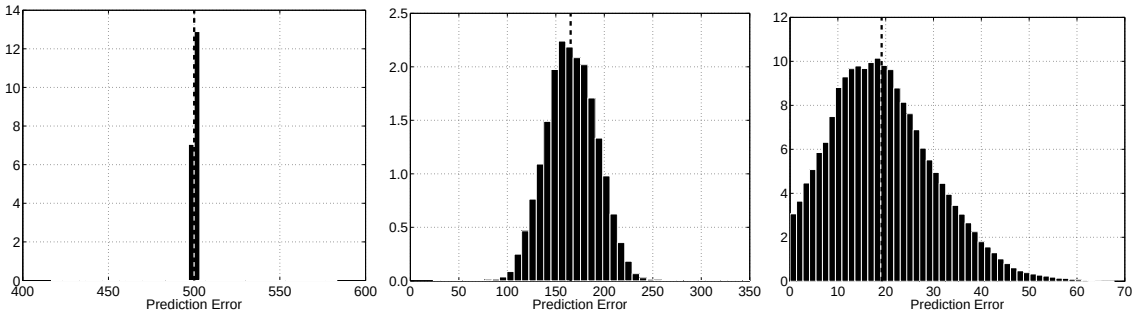
Theoretically, the error of the most over-general classifier should be $\epsilon = \{500, 165.28, 19.60\}$ for imbalance ratios $ir = \{1, 10, 100\}$ respectively. For a completely balanced domain (see Fig. 1(a)), the error oscillates around the theoretical value, i.e., $\epsilon = 500$. For $ir = 10$, the error of the most over-general classifier is around zero with high frequency. For $ir = 100$, the classifier error is zero most of the time. That is, as the classifier receives instances of the majority class, its error keeps on decreasing until becoming approximately zero. At some point, an instance of the minority class is sampled, which cause an increase in the error of the over-general classifier. For $ir = 100$, as 100 instances of the majority class are sampled for each instance of the minority class, the error of the most over-general classifier is underestimated during most of the time. Note that when $\epsilon < \epsilon_0$, XCS considers that the classifier is accurate (a typical value for $\epsilon_0 = 1$). This is the case during large part of the training time for $ir = 10$ and, especially, for $ir = 100$. Besides, as the classifier is the most over-general possible, it will be favored in detriment of highly accurate but more specific classifiers.

The problem of having non-stable estimates for over-general classifiers does not appear exclusively in highly imbalanced domains. This topic has been studied for multi-step problems with large delayed rewards, and several approaches have been designed to propagate the error along the previous action sets effectively when several steps have to be taken before reaching a reward. Herein, we consider two methods to obtain better parameter estimates. First, we show that the Widrow-Hoff rule can obtain better parameter estimates if β is properly tuned. Then, we adapt one of the most relevant methodologies for parameter evaluation designed for multi-step problems to single step tasks: *gradient descent* [9]. The next two sections show that the two methods allow for better estimates in highly imbalanced domains.



(a) $\beta = 0.2$, $ir = 1$ (b) $\beta = 0.2$, $ir = 10$ (c) $\beta = 0.2$, $ir = 100$

Fig. 1. Histogram of the error of the most over-general classifier with Widrow-Hoff delta rule at $\beta = 0.2$ and different imbalance ratios.



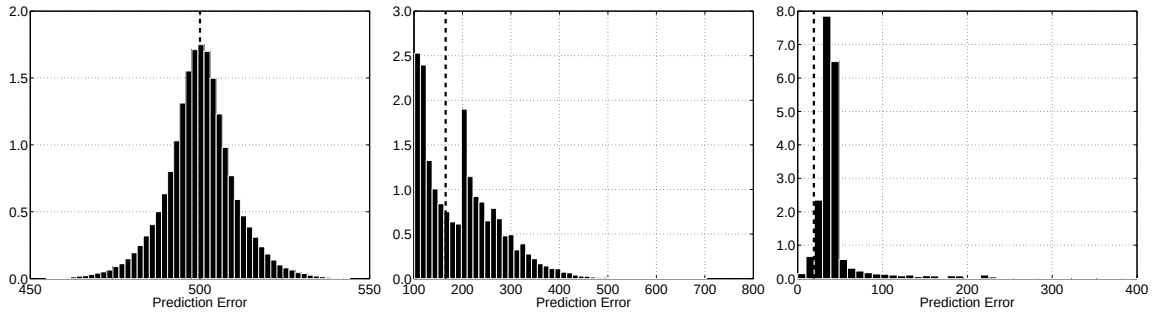
(a) $\beta = 0.01$, $ir = 1$ (b) $\beta = 0.01$, $ir = 10$ (c) $\beta = 0.01$, $ir = 100$

Fig. 2. Histogram of the error of the most over-general classifier with Widrow-Hoff delta rule at $\beta = 0.01$ and different imbalance ratios.

A. Widrow-Hoff Rule

We first illustrate how we can obtain better estimates by adjusting the β parameter of the Widrow-Hoff rule. In the Widrow-Hoff rule, the parameter β determines the proportion of update in the classifier parameters. As Widrow-Hoff rule works as a temporal windowed average, β also fixes the forgetfulness capacity of past rewards. That is, high values of β produce large corrections of classifier parameters every time a new reward is received, forgetting perviously received rewards quickly. Usually, this allows for a faster convergence of the classifier parameters to their real values. However, we have already seen the harmful effect in imbalanced domains. A possible solution would be to decrease β, considering in this way a longer history of rewards. Lower values of β would cause smaller corrections, and so, smaller oscillations. Nonetheless, they would also imply slower convergence. For very small values of β, accurate offspring classifiers may loose against over-general parents at the beginning of the run, since their fitness increases slowly. This may impair XCS to discover new accurate and more specific rules. Thence, β should be the highest value that prevents over-general classifiers from having zero error.

To show the effect of decreasing β, Fig. 2 shows a histogram of the error estimate of the most over-general classifier along a complete run for $\beta = 0.01$. For all the cases, the histograms are centered on the theoretical value



(a) $\beta = 0.2$, $ir = 1$ (b) $\beta = 0.2$, $ir = 0.1$ (c) $\beta = 0.2$, $ir = 0.01$

Fig. 3. Histogram of the error of the most over-general classifier with gradient descent at $\beta = 0.2$ and different imbalance ratios.

of the error. Thus, over-general classifiers have precise estimates of their parameters most of the time. The standard deviation of the histograms increases with the imbalance ratio, since the rewards generated by the minority class examples are less frequent.

B. Gradient Descent Methods

Here, we adopt another approach, originally adapted from a gradient descent methodology, to obtain better parameter estimations. Butz et al. [9] introduced a gradient descent method to improve the parameter estimation of XCS in multi-step problems that involve a large number of state-action pairs. The new approach relies on identifying that the gradient term for a classifier is $\frac{F}{\sum_{[A]} F_i}$, in which F_i is the fitness of classifier i of the action set. Thus, classifier prediction is updated as

$$p = p + \beta(R - p)\frac{F}{\sum_{[A]} F_i}, \tag{3}$$

where R is the received reward. The rest of the parameters are updated as usual (see Section III). Note that this procedure aims at stabilizing classifier prediction. In consequence, the classifier error will be also stabilized since it tracks the prediction error.

The results provided in [9] illustrate that the gradient descent technique enables XCS to solve multi-step problems which eluded solution in XCS with Widrow-Hoff rule. Such improvement is explained by noting that the gradient term in the prediction update is actually an adaptive learning rate that prevents parameters from large corrections when the fitness of the updated classifier is much lower with respect to the fitness of the other classifiers in the same action set. Thereupon, gradient descent automatically tunes β depending on the fitness of each classifier. Then, the difference between gradient descent and decreasing β is that gradient descent automatically uses the aforementioned heuristic to determine the most suitable value for β instead of requiring the user to tune this parameter.

We repeated the same experiments with the imbalanced multiplexer problem but now using gradient descent with $\beta = 0.2$. Fig. 3 plots the histograms of the error estimate of the most over-general classifier along a complete run for gradient descent. XCS with gradient descent maintains fairly accurate estimates of the error of the most

over-general classifier for all the imbalance ratios, even though a high value of β is used. However, note that these estimates are not as accurate as the ones obtained with Widrow-Hoff rule with a proper configuration of β.

In summary, we showed that the Widrow-Hoff rule may provide poor estimates of the error of over-general classifiers for high class imbalances. This effect is undesirable since this may cause XCS to consider over-general classifiers as accurate. Two approaches, i.e., decreasing β for Widrow-Hoff rule and gradient descent provided more reliable parameter estimates. Although these methods would result in a slower convergence of XCS, they are crucial to guarantee that XCS will be able to obtain reliable estimates and converge to an optimal population. In the remainder of the analysis, we consider that classifiers parameters are accurately estimated by the procedures presented above, and thus, that the genetic search is not misled. With this assumption, the next sections study the generation and growth of representatives in starved niches.

VI. Supply of Schemas of Starved Niches in Population Initialization

Here, we study whether the covering operator is able to supply classifiers that represent schemas of starved niches for high imbalance ratios. As explained in Section III, covering is activated in the first stages of the learning process, creating new classifiers from the first sampled instances. To provide representatives of starved niches, the covering operator has to be triggered on minority class instances. However, as ir increases, fewer minority class instances are sampled. Thus, covering will be mainly activated from majority class instances. Then, most of the classifiers will be generalized from majority class instances, and so, classifiers representing schemas of the minority class will be scarce. To analyze this effect, here we derive the probability that covering is triggered on the first minority class examples sampled to the system.

For this purpose, we first consider the probability that one instance is covered by, at least, one classifier $P(\text{cover})$. According to [12], this probability is

$$P(\text{cover}) = 1 - \left[1 - \left(\frac{2-\sigma[P]}{2}\right)^{\ell}\right]^{N}, \tag{4}$$

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

Now, let us relate this probability to the imbalance ratio ir. We consider the worst case where (i) XCS receives ir instances of the other classes before receiving the first instance of the minority class and (ii) the covering operator is triggered for each instance supplying n classifiers per instance (where n is the number of classes; thus $\theta_{mna} = n$). In this case, the probability that the population contains, at least, a matching classifier for each class is

$$P(\text{cover}) = 1 - \left[1 - \frac{1}{n}\left(\frac{2-\sigma[P]}{2}\right)^{\ell}\right]^{n\cdot ir}. \tag{5}$$

In this equation, we assumed that $N > n\cdot ir$, i.e., that XCS will not delete any classifier during the first ir iterations. This assumption is usually satisfied since covering is only applied for the first input examples, when there is room in the population. It is worth noting that, given a fixed ℓ and $\sigma[P]$, the term in brackets in the right hand of the equation decreases exponentially as the imbalance ratio increases. Thus, the probability of having classifiers in the

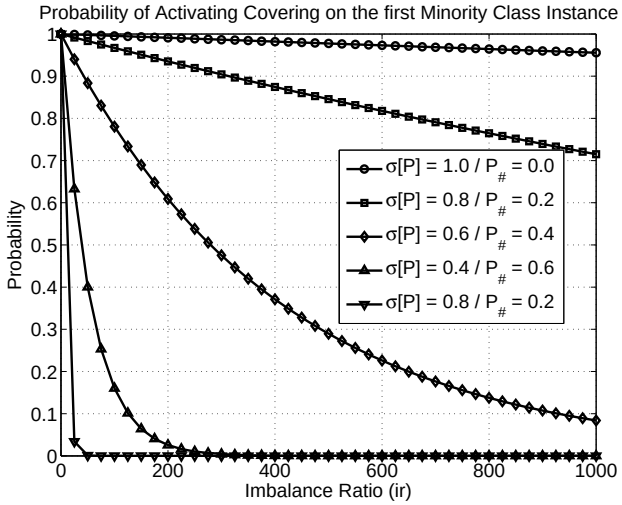


Fig. 4. Probability of activating covering on a minority class instance given a certain specificity $\sigma[P]$ and the imbalance ratio ir. The curves have been drawn from Equation 6 with $\ell = 20$ and different specificities.

population that match the first minority class instances tends to one exponentially with the imbalance ratio. Notice that the matching classifiers would have been generated from majority class instances, and so, would not represent schemas of starved niches.

With Equation 5 we can derive the probability of activating covering having sampled a minority class instance. Provided that the probability of activating covering is $1 - P(cover)$, and recognizing that $(1 - r/n)^n \approx e^{-r}$, we obtain that

$$P(\text{activate cov. on. min.}) = 1 - P(\text{cover}) \approx e^{-ir \cdot e^{-\frac{\ell\sigma[P]}{2}}}, \tag{6}$$

which decreases exponentially with the imbalance ratio and, in a higher degree, with the condition length and the initial specificity. Fig. 4 depicts the equation for $\ell = 20$, $n = 2$, and different initial specificities, showing that the probability of activating covering on the first sampled instance of the minority class decreases exponentially with the imbalance ratio.

The analysis made above manifests that the covering operator fails to supply classifiers representing correct schemas of the minority class for moderate and high imbalance ratios. In classification tasks, dynamic re-sampling techniques could be applied to overcome this problem [31]. Nonetheless, we are interested in deriving the domain of competence of XCS on imbalanced domains; moreover, as XCS learns from a stream of examples, the information required by traditional resampling methodologies, such as the imbalance ratio of the learning data set, is not available, and poor estimations can be done in the beginning of the run. For these two reasons, we do not consider preprocessing techniques such as re-sampling methodologies. In the next section, we assume a covering failure in providing schemas of starved niches and study whether the genetic search can discover representatives of starves niches.

VII. Generation of Classifiers in Starved Niches

Assuming a covering failure to provide schemas of starved niches, this section derives facetwise models that explain how the genetic search can evolve starved niches. As follows, we first enumerate the assumptions of the models and then derive the time to create and delete representatives of starved niches. These models are used to deduce population size bounds to ensure the discovery, maintenance, and growth of starved niches.

A. Assumptions for the Model

Before proceeding with the theoretical derivations, we first enumerate the assumptions made to develop the models. The analysis is focused on the evolution of starved niches. We assume a simplified scenario model where: (i) we do not consider crossover and contemplate mutation as the main operator for discovery, assuming low probabilities of mutation μ ($\mu < 0.5$) as usual in practice; (ii) we assume that the GA is applied at each learning iteration (i.e., $\theta_{GA} = 0$); and (iii) we consider random deletion. Subsequently, we relax all these constraints and experimentally analyze the impact of breaking each one of the assumptions. We experimentally examine the effect of introducing two point crossover. Furthermore, we investigate the biases caused by the enhanced deletion technique used currently in XCS (see Section III-D), and, in the next section, we study the effect of θ_{GA} theoretically and empirically.

B. Genetic Creation of Representatives of Starved Niches

In the first step of the analysis, we derive the time until the creation of the first representatives of starved niches, assuming that covering has not provided any of them. To achieve this, we first derive the probability to obtain the first accurate representative cl_{min} of the starved niche i, which is represented by a schema with order k_m. Thence, we study the probabilities of creating cl_{min} when sampling (i) instances of the minority class and (ii) instances of the majority class. Having that the probability of sampling a minority class instance is $1/(1+ir)$ and the probability of sampling a majority class instance is $ir/(1+ir)$, we can write that

$$P(cl_{min}) = \frac{1}{1+ir} P(cl_{min}|\text{min. inst}) + \frac{ir}{1+ir} P(cl_{min}|\text{maj. inst}). \quad (7)$$

Let us first derive $P(cl_{min}|\text{min. inst})$. As we assumed that there are no representatives of starved niches in the population, when sampling a minority class instance, the match set will only consist of over-general classifiers. Then, the system will choose a class randomly and will explore it, running a genetic event on the selected action set. Regardless of the selected class, and considering that there are only over-general classifiers in [M], to create a representative of a starved niche, all the k_m bits must be correctly set to the values of the niche schema; here, we consider the worst case and assume that all the k_m bits need to be mutated. Thence, the probability of getting the correct schema is $(\frac{\mu}{2})^{k_m}$. Besides, the class of the classifier needs to be set to the class of the niche. If the system selected to explore the minority class (which will be selected with probability $1/n$, where n is the number of classes), we have to ensure that the mutation operator would not change this class (that is, with probability $(1-\mu)$).

Otherwise, we have to require that the mutation operator change this class to the niche class (with probability $\mu/(n-1)$). Therefore,

$$P(cl_{min}|\text{min. inst}) = \frac{1}{n}\left(\frac{\mu}{2}\right)^{k_m} \cdot (1-\mu) + \frac{n-1}{n}\left(\frac{\mu}{2}\right)^{k_m} \cdot \frac{\mu}{n-1} = \frac{1}{n}\left(\frac{\mu}{2}\right)^{k_m}. \quad (8)$$

The same procedure can be followed to derive the probability of creating cl_{min} when sampling an instance of the majority class, i.e., $P(cl_{min}|\text{maj. inst})$. In this case, the match set will consist of both over-general classifiers and representatives of nourished niches. Again, we consider the worst case and assume that, to create a representative of a starved niche, all the k_m bits of the niche schema need to be mutated. Moreover, if the system chooses to explore the minority class, the class of the classifier must be preserved; otherwise, the class has to be changed to the minority class. This results in exactly the same probability as before, i.e.,

$$P(cl_{min}|\text{maj. inst}) = \frac{1}{n}\left(\frac{\mu}{2}\right)^{k_m}. \quad (9)$$

Substituting Equations 8 and 9 into Equation 7 we obtain that

$$P(cl_{min}) = \frac{1}{n}\left(\frac{\mu}{2}\right)^{k_m}. \quad (10)$$

Then, we can derive the time required to discover the first representatives of starved niches $t_{cl_{min}}$ as

$$t_{cl_{min}} = \frac{1}{P_{cl_{min}}} = n\left(\frac{2}{\mu}\right)^{k_m}, \quad (11)$$

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

Thus, even though covering fails to provide classifiers representing schemas of the minority class, XCS will be able to generate the first correct classifiers of the minority class independent of the imbalance ratio. As follows, we derive the time until deletion of these classifiers.

C. Deletion of Representatives of Starved Niches

The time to extinction of classifiers mainly depends on the applied deletion procedure. The current deletion scheme of XCS [38] gives the classifiers a deletion probability proportional to their action set estimate as. This approach permits to balance the allocation of rules in the different niches in problems for which the frequency of the different niches is similar, i.e., balanced problems. Nonetheless, in highly imbalanced problems, the action set size estimate of accurate classifiers of starved niches may be negatively biased by over-general classifiers. That is, as over-general classifiers participate in the same action sets as accurate classifiers of starved niches, the action set size of these accurate classifiers is overestimated.

As our model is developed for highly imbalanced domains, we consider the worst case, i.e., that the deletion procedure gives the same probability to each classifier to be deleted. Since two classifiers are deleted at each GA application, we obtain that the deletion probability is $P(\text{delete cl}) = 2/N$, where N is the population size. From this formula, we derive the time until deletion

$$t(\text{delete cl}) = \frac{N}{2}. \quad (12)$$

In the next section, we use both the creation and the extinction time of representatives of starved niches to derive the minimum population size that guarantees the discovery, maintenance, and growth of starved niches.

D. Bounding the Population Size

The population size is a critical aspect which determines the niches that the system could maintain. In this section, we use the formulas calculated above and derive population size bounds to guarantee (i) that XCS will be able to maintain accurate representatives of starved niches, and (ii) that representatives of starved niches will receive genetic events before being removed.

1) Minimum Population Size to Guarantee Representatives: In the previous section, we theoretically showed that XCS would be able to create representatives of starved niches regardless of the imbalance ratio. Now, we derive the population size conditions that enable XCS to maintain these representatives. For this purpose, we need to guarantee that, before deleting any representative of a starved niche, another one has to be generated. Therefore, we use the formulas derived in the previous section and require that the time until deletion be greater than the time until a new representative of a the starved niche is created. That is, we require that

$$t(\text{delete } cl_{min}\ ) > t(cl_{min}). \tag{13}$$

Using formulas 11 and 12, the expression can be rewritten as

$$N > 2n\left(\frac{\mu}{2}\right)^{k_m}, \tag{14}$$

which indicates that the population size have to increase linearly with the number of classes and exponentially with the order of the schema to guarantee that representatives will be maintained in starved niches. Note that this formula does not depend on the imbalance ratio. This means that XCS will be able to maintain accurate classifiers in starved niches regardless the imbalance ratio.

2) Population Size Bound to Guarantee Reproductive Opportunities: To ensure the growth of starved niches, we not only should guarantee that XCS would maintain representatives of starved niches, but that these representatives receive, at least, a genetic opportunity. Otherwise, XCS could be continuously creating and removing classifiers from starved niches, but not searching toward better classifiers. Therefore, here we derive a population size bound to ensure this condition.

In our model, we assume that the selection procedure chooses one of the strongest classifiers in the niche (the effect of different selection schemes will be studied in more detail in the next section). Then, the time required for a classifier of a starved niche to receive a genetic event is inversely proportional to the probability of activation of the niche to which it belongs, i.e.,

$$t(\text{GA } niche_{min}) = n \cdot (1 + ir), \tag{15}$$

which depends on the imbalance ratio and the number of classes.

To guarantee that these accurate classifiers of starved niches receive a genetic opportunity before being deleted, we require that $t(\text{delete } niche_{min}) > t(\text{GA } niche_{min})$, from which we derive the population size bound

$$N > 2n \cdot (1 + ir). \tag{16}$$

That is, the population size has to increase linearly with the number of classes and the imbalance ratio to warrant that accurate classifiers of starved niches will receive, at least, a genetic event before being deleted.

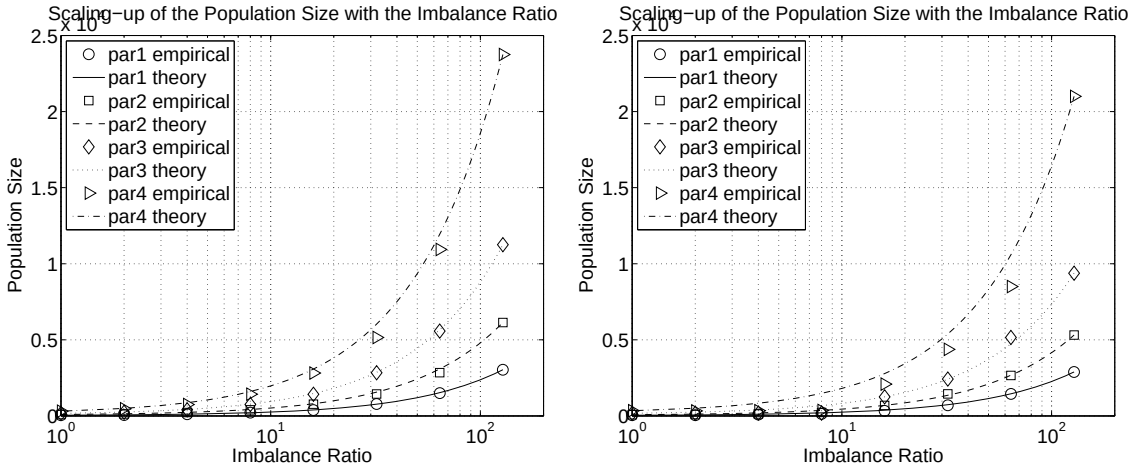
In this section we derived theoretical models that explain the creation, maintenance, and growth of starved niches. The models showed that XCS is able to maintain representatives of starved niches regardless of the imbalance ratio and that the population size has to increase linearly with the imbalance ratio if we want to ensure that the niche will grow. In the next section, we empirically validate the population size bounds with a set of artificial problems.

E. Experimental Validation of the Models

In this section, we experimentally evaluate whether the population size bound increases linearly with ir as predicted by the bound in Equation 16. We first analyze whether the theory fits the experimental results when all the assumptions are made. Later, we study the impact of breaking each one of the assumptions.

1) Experimental Validation when the Assumptions Are Satisfied: To examine whether the theory approximates accurately the empirical results when all the assumptions are met, we performed the following experiments. We ran XCS on the imbalanced parity problem with $k = \{1, 2, 3, 4\}$, $\ell = 10$, and $ir = \{1, 2, 4, 8, 16, 32, 64, 128\}$, and we used the bisection procedure to obtain the minimum population size required to solve the problem. That is, for each parity problem and imbalance ratio, we ran XCS with an initial, randomly selected population size. If the run succeeded, we decreased the population size. Otherwise, we increased the population size. This procedure was repeatedly applied until we obtained the minimum population size with which XCS was able to solve the problem. We employed the following procedure to determine if an XCS run was successful. After training, we tested XCS with all the training instances and measured the proportion of correct classifications of instances of the majority class (TN rate) and the proportion of correct classifications of the minority class (TP rate). All these results were averaged over 50 different random seeds. We considered that XCS succeeded if the product of TP rate and TN rate was greater than a certain threshold θ (we set $\theta = 0.95$).

XCS was configured so that all the assumptions of the model were satisfied. Therefore, crossover was deactivated ($\chi = 0$), random deletion was used, and the GA was applied every time a niche was activated (θ_{GA}=0). The other parameters were set as $\alpha = 0.1$, $\epsilon_0 = 1$, $\nu = 10$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $\theta_{sub} = ir$, $P_{\#} = 0.6$. We used tournament selection for the GA. We ran XCS during $\{10,000 \cdot ir, 20,000 \cdot ir, 40,000 \cdot ir, 80,000 \cdot ir\}$ iterations for the par problem with $k = \{1, 2, 3, 4\}$ respectively; thus, given a problem, we ensured that the system received the same number of genetic opportunities for all imbalance ratios. Finally, to prevent having young over-general classifiers with poorly estimated parameters in the final population, we introduced $5,000 \cdot ir$ iterations with the GA switched off at the end of the learning process. In the remainder of this paper, this configuration is referred to as the *default configuration*.



(a) Default configuration (b) Gradient descent

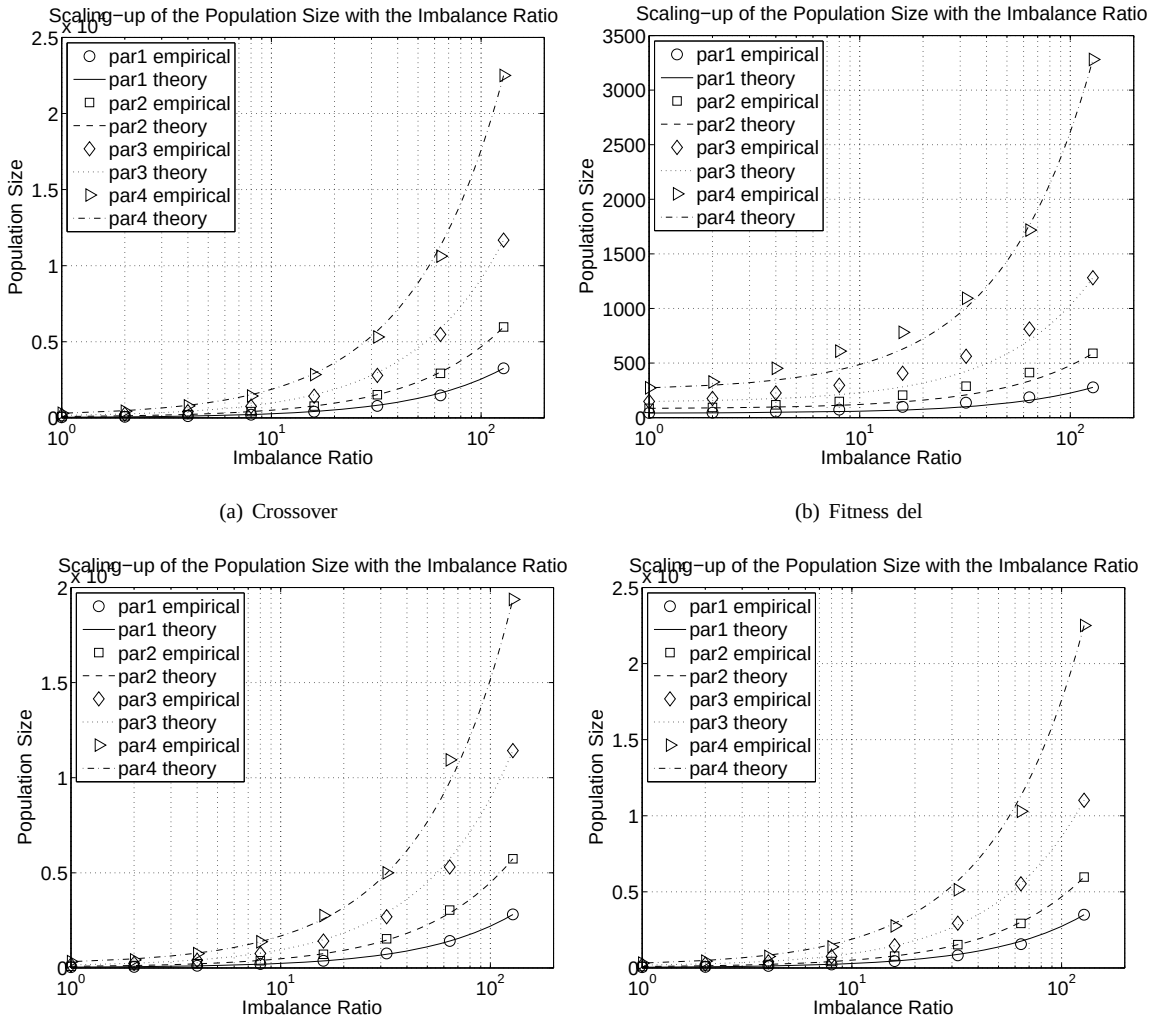
Fig. 5. Scalability of the population size (N) with the imbalance ratio (ir) in the k-parity problem with k={1,2,3,4} and the default configuration with (a) Widrow-Hoff rule update with adjusted β according to ir and (b) Gradient descent parameter's update with $\beta = 0.2$. The dots shows the empirical results and lines plot linear increases with ir (according to the theory).

The two parameter update procedures proposed in Section V were used: (1) Widrow-Hoff rule and (2) gradient descent. For the former method, we used the following heuristic procedure to tune β. For each run, we supposed the worst case and assumed that over-general classifiers received 1 instance of the minority class and then ir instances of the majority class. Thus, we set β so that the error calculated for the most over-general classifier was approximately the same as the theoretical error provided by Equation 2. We followed an iterative approach that incrementally discounted the value of β until a value that yielded error estimates that were close to the theoretical ones was found.

Fig. 5 shows the minimum population size required to solve the parity with different building block sizes ($k = \{1, 2, 3, 4\}$) and imbalance ratios from $ir = 1$ to $ir = 128$ for Widrow-hoff rule and gradient descent. For each plot, the points depict the empirical values and the lines show the theoretical bounds. Note that the theory approximates the empirical results accurately for the two parameter's update procedures and the different configurations and imbalance ratios. These results also permit to establish a comparison among the two parameter update procedures. The pairwise Wilcoxon statistical test [45], at $\alpha = 0.05$, indicated that Gradient descent needed significantly smallest populations to solve the different configurations of the parity problem.

The results provided herein indicated that the theory nicely predicts the experiments when the assumptions of the models are met. In the next section, we investigate whether the population size bound is still valid when the different assumptions are not satisfied.

2) Impact of Breaking the Assumptions: Here, we repeated the experiments done in the previous section, but breaking each assumption. That is, we used the default configuration specified in the last section and Widrow-Hoff rule was employed for parameter estimation. However, we ran XCS with (i) crossover, setting $\chi = 0.8$, and (ii) the



(a) Crossover (b) Fitness del

(c) RWS (d) $P_{\#}$=0.4

Fig. 6. Scalability of the population size (N) with the imbalance ratio (ir) in the k-parity problem with k={1,2,3,4} and different XCS's configurations. The dots shows the empirical results and lines plot linear increases with ir (according to the theory).

typical deletion scheme of XCS, configuring $\theta_{del} = 20$ and $\delta = 0.1$. Moreover, we also analyzed (iii) the effect of replacing tournament selection with proportionate selection and (iv) the impact of increasing the specificity in the initial population by setting $P\# = 0.4$. Fig. 6 shows the minimum population size required in each configuration.

Several conclusions can be drawn from these results. First of all, it is worth noting that the theory nicely approximates the empirical results for all the experiments, although the initial assumptions are not satisfied. Fig. 6(a) shows the curves obtained by XCS with crossover, which are equivalent to those evolved with the default configuration (see Fig. 5(a)) according to a Wilcoxon signed-ranks test at a significance level of 0.05. This suggests that the models are still valid although crossover is used in the experimental runs. Fig. 6(b) plots the curves resulting of running XCS with the typical deletion scheme of XCS. The results clearly evidence the decrease in the population

size required to solve the different configurations (the Wilcoxon signed-ranks test confirmed this observation). In any case, note that the the minimum population size still increases linearly with the imbalance ratio, as predicted by the theory. The configuration with proportionate selection (see Fig. 6(c)) yielded equivalent results to those obtained with the default configuration according to a Wilcoxon signed-ranks test at a significance level of 0.05. Finally, Fig. 6(d) illustrates the results obtained when there was a higher specificity in the initial populations. The pairwise analysis supports the hypothesis that a higher initial specificity requires larger population sizes to solve the parity with $k = 1$. For the other parity problems, no statistical differences were found.

The overall experimentation conducted in this section showed an agreement between theory and experiments, even when the initial assumptions were not satisfied. Notice that no experiment broke one of the assumptions: that the GA is applied at each learning iteration. The analysis of the frequency of application of the GA is carefully studied in the next section.

VIII. Occurrence-based Reproduction: The Role of θ_{GA}

Throughout all the analysis performed in the last section, we assumed that the different niches of the system receive a genetic opportunity each time they are activated (i.e., θ_{GA}=0). Consequently, nourished niches received more genetic events, and so, generated more offspring. This section takes in consideration the effect of having $\theta_{GA} > 0$, revisits the models derived in the previous section, and shows that we can use θ_{GA} to re-balance the number of genetic opportunities that both starved and nourished niches receive. Finally, the new models are validated with the imbalanced parity problem.

A. Including θ_{GA} in the Generation Models

To analyze the impact of varying θ_{GA}, let us consider again the occurrence-based reproduction of both types of niches and calculate the period of application of the GA to the different niches. The frequency of activation of nourished niches ($F_{occ_{maj}}$) and the frequency of activation of starved niches ($F_{occ_{min}}$) are

$$F_{occ_{maj}} = \frac{1}{n \cdot m}\frac{ir}{1+ir} \quad \text{and} \quad F_{occ_{min}} = \frac{1}{n \cdot m}\frac{1}{1+ir}, \tag{17}$$

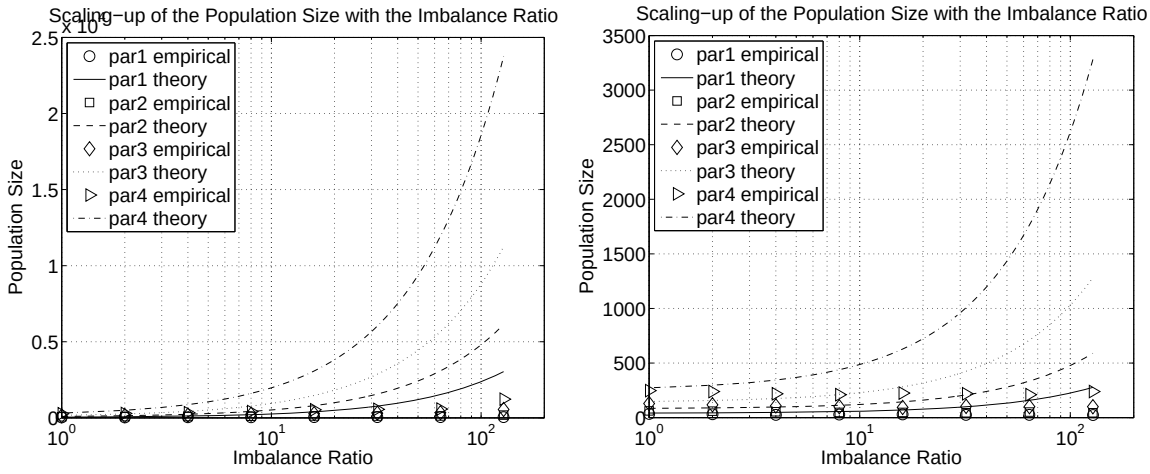
where m is the number of niches[2]. From these frequencies, we can compute the period of activation of each type of niche as

$$T_{occ_{maj}} = n \cdot m\frac{1+ir}{ir} \quad \text{and} \quad T_{occ_{min}} = n \cdot m(1+ir). \tag{18}$$

Once activated, the niche will receive a genetic event if the time since the last application of the GA on the niche exceeds θ_{GA}. Therefore, the period of application of the GA (T_{GA}) on a niche is

$$T_{GA} = \begin{cases} T_{occ} & \text{if } T_{occ} > \theta_{GA} \\ \theta_{GA} & \text{otherwise.} \end{cases} \tag{19}$$

[2]We introduce the number of niches in these equations since we are now modeling the occurrence of a specific niche



(a) Default conf.

(b) Fitness del

Fig. 7. Scalability of the population size (N) with the imbalance ratio (ir) in the k-parity problem with k={1,2,3,4} and different XCS's configurations with $\theta_{GA} = n \cdot m \cdot ir$. The points indicate the empirical values of the minimum population size required by XCS. The lines depict the theoretical increase calculated with the previous models, which assumed $\theta_{GA} = 0$.

That is, if the period of activation of a niche is greater than θ_{GA}, the classifiers that belong to the niche will receive a genetic event every time the action set is formed; thus, the period of application of the GA equals the period of niche activation. This is the case of the theoretical model, in which we assumed $\theta_{GA} = 0$. On the other hand, if $T_{occ} \leq \theta_{GA}$, T_{GA} is approximately θ_{GA}.

To give all niches the same number of genetic events, T_{GA} should be approximately the same for all the niches. Note that this can be easily satisfied by setting $\theta_{GA} = T^*_{occ}$, where T^*_{occ} is the period of the niche that is activated less frequently, i.e., $T^*_{occ} = T_{occ_{min}}$. Therefore, θ_{GA} should be set as follows

$$\theta_{GA} \approx n \cdot m \cdot (1 + ir). \tag{20}$$

Note that if the restriction of Equation 20 is satisfied, all niches will receive approximately the same number of genetic events; moreover, as deletion is only activated after a GA application, the time of deletion of a classifier (see Equation 12) would now increase linearly with the imbalance ratio. Therefore, XCS will be able to maintain starved niches without increasing the population size. The next section experimentally validates this assertion.

B. Experimental Validation

To validate the theory derived in the previous section, we ran the same experiments with the parity problem proposed in Section VII-E. We configured the system with the default configuration but we set $\theta_{GA} = n \cdot m \cdot ir$; Widrow-Hoff rule was used to update classifiers parameters. Fig. 7(a) shows the minimum population size required to solve the parity problem with different building block sizes ($k = \{1, 2, 3, 4\}$) and imbalance ratios from $ir = 1$ to $ir = 128$. The points depict the empirical values. To analyze the differences introduced by adjusting $\theta_{GA} = n \cdot m \cdot ir$,

the lines depict the population size increase predicted by the theoretical model calculated for the same configurations but with $\theta_{GA} = 0$ (see Fig. 5(a)).

The figure shows that, with the default configuration, the population size remained nearly constant for all the tested parity problems and imbalance ratios. This is because the effect of the imbalance ratio was counter-balanced by the increase of the period of application of the GA, as deduced in the previous section. The population size only presented a slight increase for $ir = 128$. This behavior can be easily explained as follows. At such imbalance ratios, the parameter update procedure decreases the value of β to prevent the oscillation of the parameters of over-general classifiers. For very small values of β, accurate offspring classifiers may lose against over-general parents at the beginning of the run, since their fitness increases slowly. As the deletion procedure is random and these offspring receive a low number parameter updates, they may be removed before their parameters are correctly adjusted to the real value. Therefore, lightly larger populations may be set to let new accurate offspring persist in the population until their parameters are sufficiently updated.

To contrast this hypothesis, we ran the same experiments but using the typical deletion scheme of XCS. Fig. 7(a) illustrates the minimum population sizes required for each configuration of the parity problem. The experimental results show that the population size remains constant as the imbalance ratio increases, even for the largest imbalance ratios. That is, the typical deletion scheme of XCS protects the young classifiers by giving more deletion probability to over-general, experienced classifiers.

With the present study, we have theoretically and empirically demonstrated that representatives of starved classifiers will be evolved independent of the population size. In the next section, we analyze the takeover time of these classifiers in more detail.

IX. Takeover Time of Accurate Classifiers in Starved Niches

The study provided until now showed that XCS is able to create accurate classifiers of starved niches, and that these classifiers will receive, at least, a genetic opportunity before being deleted. This facet of the analysis set the population size requirements to guarantee that starved niches are represented. Also, the effect of increasing θ_{GA} was analyzed in detail. However, the conditions required in the previous models are not enough; to ensure full convergence, we have to warrant not only that starved niches will not be extinct but also that accurate classifiers will take over starved niches, removing the majority of over-general classifiers. Therefore, we have to analyze the competition between accurate classifiers of starved niches and over-general classifiers. This analysis is crucial because it permits to extract the upper bound on the admissible imbalance ratio beyond which XCS will not be able to converge.

The purpose of this section is to model the takeover time of the best representatives of starved niches and determine the conditions under which starved niches will be extinguished. We first calculate the takeover time of accurate classifiers, which depends on (i) the initial stock of accurate classifiers in the niche and (ii) the type of selection used by the GA. In LCSs, two selection procedures have mainly been considered: proportionate selection [6] and tournament selection [37]. In this section, we model the takeover time of the best classifier of a niche for

both selection schemes. Although we focus the analysis on the effect of class imbalances, note that the derived models can be used as general models for the two selection schemes. Then, we use the takeover time equations to calculate the extinction conditions of a niche, i.e., the conditions under which all representatives of a given starved niche will be removed from the population due to an overpressure toward generalization. As follows, we present the assumptions made for the analysis, develop the models for each type of selection, and experimentally validate the takeover time and extinction models.

A. Model Assumptions

In [46], models for takeover time in XCS for proportionate and tournament selection were derived. The models were restricted to problems with a single niche. This assumption permitted to relax the initial conditions, and so, make the model derivation simpler. Nonetheless, this restriction makes these models not suitable for studying the effect of class imbalances, since now we are interested in modeling the interaction among representatives of starved niches and over-general classifiers. Therefore, here we derive takeover time models for problems with an arbitrary number of niches m. The models consider that all the m niches appear with the same frequency. In fact, in Section VIII, we showed that this could be easily achieved by setting θ_{GA} according to ir. Then, we incorporate the effect of the imbalance ratio in the error of the over-general classifier. That is to say, imagine that we have a two-class problem. For $ir = 1$, the error ϵ_o of the most over-general classifier will be $\epsilon_o \approx 500$, since this classifier will predict correctly half of the instances. As the imbalance ratio increases, ϵ_o decreases as shown in Section V. Thence, the imbalance ratio is intrinsically included in the difference between the errors of the over-general and the accurate classifiers. Thus, the takeover time models derived herein compute whether high imbalance ratios may discourage XCS to promote representatives of starved niches in favor of over-general classifiers.

To model the takeover time, we make the following assumptions. We consider that XCS has evolved a maximally general and accurate classifier cl_b, with error ϵ_b and numerosity n_b, for each niche of the system (this is ensured by the models provided in the last sections). Moreover, we assume that there is a single over-general classifier n_o, with error ϵ_o and numerosity n_o, which matches all the niches. As cl_b is maximally accurate, $\epsilon_o > \epsilon_b$. The same expression can be written in function of the classifiers accuracy (a inverse function of the error) as $\kappa_b > \kappa_o$. Therefore, our aim is to model the competition between accurate representatives of the different niches and the over-general classifier. For the analysis, we assume random deletion. We also consider that the GA is applied at each learning iteration and that both crossover and mutation are switched off. Therefore, the GA only selects two parents, copies and introduces them into the population, removing two other classifiers. The subsequent sections model the takeover time for proportionate and tournament selection under these assumptions.

B. Takeover Time for Proportionate Selection

In this section, we first derive the probability of selecting the best representative of a niche under proportionate selection, and then, we use this information to develop equations that model the evolution of the numerosity of this classifier in the niche. Under proportionate or roulette wheel selection (RWS), the selection probability of a

classifier i depends on the ratio of its fitness F_i over the fitness of all classifiers in the action set. Without loss of generality, we assume that classifier's fitness is a simple average of classifier's relative accuracies. Thus, focusing on a single niche, we compute the fitness of classifiers cl_b and cl_o as

$$F_b = \frac{\kappa_b n_b}{\kappa_b n_b + \kappa_o n_o} = \frac{1}{1+\rho n_r}$$

and

$$F_o = \frac{\kappa_o n_o}{\kappa_b n_b + \kappa_o n_o} = \frac{\rho n_r}{1+\rho n_r},$$

where $n_r = n_o/n_b$ and ρ is the ratio between the accuracy of cl_o and the accuracy of cl_b ($\rho = \kappa_o/\kappa_b$). ρ can also be viewed as the fitness separation between cl_o and cl_b. The probability P_s of selecting the best classifier cl_b in the niche is computed as

$$P_{s_{RWS}} = \frac{F_b}{F_b + F_o} = \frac{1}{1+\rho n_r}.$$

Once selected, each classifier is copied and inserted into the population while one classifier is randomly deleted from the niche with probability $P_{del}(cl_j) = n_j/N$, where N is the population size. With this information, as proceeds we model the evolution of the best classifier in a niche.

1) Evolution of the Best Classifier: The numerosity of the best classifier cl_b at time $t+1$, $n_{b,t+1}$, given the numerosity of the classifier at time t, $n_{b,t}$, will

- increase in the next generation if the GA is applied to the niche, cl_b is selected by the GA, and another classifier is selected for deletion;
- decrease if (a) the GA is applied to the niche, cl_b is not selected by the GA, but cl_b is selected for deletion or if (b) the GA is not applied to the niche and cl_b is selected for deletion;
- remain the same, in all the other cases.

More formally,

$$n_{b,t+1} = \begin{cases} n_{b,t}+1 & \frac{1}{m}\frac{1}{1+\rho n_{r,t}}\left(1-\frac{n_{b,t}}{N}\right), \\ n_{b,t}-1 & \frac{1}{m}\left(1-\frac{1}{1+\rho n_{r,t}}\right)\frac{n_{b,t}}{N} + \frac{m-1}{m}\frac{n_{b,t}}{N}, \\ n_{b,t} & \text{otherwise.} \end{cases}$$

where m is the number of niches in the problem. Grouping the above equations, we obtain

$$n_{b,t+1} = n_{b,t} + \frac{1}{m}\cdot\frac{1}{1+\rho n_{r,t}}\left(1-\frac{n_{b,t}}{N}\right) - \frac{1}{m}\left(1-\frac{1}{1+\rho n_{r,t}}\right)\frac{n_{b,t}}{N} - \frac{m-1}{m}\frac{n_{b,t}}{N}, \tag{21}$$

which can be expressed as

$$n_{b,t+1} = n_{b,t} + \frac{1}{m}\frac{1}{1+\rho n_{r,t}} - \frac{n_{b,t}}{N}. \tag{22}$$

This expression can be rewritten in terms of the proportion P_t of classifiers cl_b in the whole population, i.e.,

$$P_t = \frac{n_{b,t}}{N}. \tag{23}$$

Considering that the numerosity of the best classifier in each niche is $n_{b,t}$, we write that the numerosity of the over-general classifier $n_{o,t}$ is $n_{o,t} = N - m \cdot n_{b,t}$, and thus, $n_{r,t}$ can be expressed as

$$n_{r,t} = \frac{n_{o,t}}{n_{b,t}} = \frac{1 - m \cdot P_t}{P_t}. \tag{24}$$

Replacing equations 24 and 23 into equation 22, we obtain

$$P_{t+1} = P_t + \frac{1}{Nm} \frac{P_t}{P_t + \rho(1 - mP_t)} - \frac{1}{N} P_t. \tag{25}$$

Assuming $P_{t+1} - P_t \approx dp/dt$, we have

$$\frac{dp}{dt} \approx P_{t+1} - P_t = \frac{1}{Nm} \frac{P_t}{P_t + \rho(1 - mP_t)} - \frac{1}{N} P_t = \frac{P_t - mP_t^2 - \rho m P_t(1 - mP_t)}{Nm\left[P_t + \rho(1 - mP_t)\right]}. \tag{26}$$

That is,

$$\frac{P_t(1 - m\rho) + \rho}{P_t\left[(1 - \rho m) - mP_t(1 - \rho m)\right]} dp = \frac{1}{Nm} dt, \tag{27}$$

which can be solved by integrating each side of the equation between the initial proportion P_0 of cl_b and the final proportion P_F of cl_b up to which cl_b has taken over the population. Note that, assuming m balanced niches, cl_b will take over, at most, a proportion $1/m$ of the population. From this expression, we derive the following closed-form equation for takeover time of cl_b in roulette wheel selection (see Appendix A for details):

$$t^*_{RWS} = Nm \left[\frac{1}{m} \ln \left(\frac{1 - mP_0}{1 - mP_F} \right) + \frac{\rho}{1 - m\rho} \ln \left(\frac{P_F(1 - mP_0)}{P_0(1 - mP_F)} \right) \right]. \tag{28}$$

The takeover time formula depends on (i) the fitness separability ρ and (ii) the number of niches m. If ρ increases, the influence of the second logarithm also increases; therefore, as the accuracies of cl_b and cl_o get closer, the takeover time increases.

In this subsection we provided a closed-form solution of the takeover time for proportionate selection. The next subsection uses this information to extract the conditions under which the best classifier will not take over its niche.

2) Conditions for Starved Niches Extinction under Proportionate Selection: With the formulas derived above, we analyze under which circumstances the best classifier will not be able to take over the population, and then, we relate it to the imbalance ratio. For this purpose, we take Equation 26 and analyze under which conditions the increment of the numerosity of the best classifier will be negative; in this case, the best classifier will lose copies in favor of over-general classifiers. We can write this expression as

$$\frac{P_t - mP_t^2 - \rho m P_t(1 - mP_t)}{Nm\left[P_t + \rho(1 - mP_t)\right]} < 0, \tag{29}$$

that is,

$$\frac{P_t\left(1 - m\rho\right)(1 - mP_t)}{Nm\left[P_t(1 - m\rho) + \rho\right]} < 0. \tag{30}$$

This condition holds when the numerator is positive and the denominator is negative and viceversa. Thus, we search for values of ρ and m that result in combinations of positive numerator and negative denominator and viceversa. Assuming that $P_t < \frac{1}{m}$, we have the following cases. For $\rho < \frac{1}{m}$, both numerator and denominator take positive values. Therefore, for $\rho < \frac{1}{m}$, the best classifier will always take over the population. For $\rho = \frac{1}{m}$, the expression

is 0, indicating that numerosity of the best classifier would remain constant. For $\rho > \frac{1}{m}$, the numerator is always negative. Then, the whole expression will be negative if the denominator is positive, i.e.,

$$Nm\left[P_t(1-m\rho)+\rho\right] > 0, \tag{31}$$

which can be written as

$$P_t < \frac{\rho}{m\rho - 1}. \tag{32}$$

Having that $0 < P_t < \frac{1}{m}$ and $\frac{1}{m} < \rho \leq 1$, this expression is always satisfied for $m \geq 2$. Thence, this theoretically demonstrates that for $m \geq 2$ and a low separation between the accuracy of the most over-general classifier and the best representative of the niche, i.e., $\rho > \frac{1}{m}$, the best representative will not be able to take over the niche. Note that this expression can be easily related to the imbalance ratio by identifying that $\rho = \frac{\kappa_o}{\kappa_b}$, where the accuracy of the over-general classifier can be computed from its error, which is expressed in Equation 2. Oppositely, in all the other cases, the best classifier will take over its niche.

The overall analysis conducted above provided both a takeover time for proportionate selection and the conditions of extinction of starved niches. Before proceeding with the experimental validation of the theory, the next section follows a similar procedure to derive the takeover time for tournament selection.

C. Takeover Time for Tournament Selection

To develop takeover time models for tournament selection, we assume that the tournament size s is fixed during all the learning process. That is, in each GA event, tournament selection randomly chooses s classifiers in the action set and selects the one with the highest fitness. As before, we assume that cl_b is the best classifier in the niche and cl_o is the over-general classifier; in terms of tournament selection, this translates into requiring that $f_b > f_o$, where f_i is the fitness of the micro-classifiers associated to cl_i (i.e., $f_i = F_i/n_i$). Given this scenario, the probability of selecting the best classifier is

$$P_{s_{TS}} = \left[1 - \left(1 - \frac{n_o}{n}\right)^s\right], \tag{33}$$

where n is the numerosity of the niche, i.e., $n = n_b + n_o$. Thus, the probability of selecting the best classifier is one minus the probability that this classifier does not participate in the tournament. With this information, the next subsections model the evolution of the best classifier, provide some particular expressions of the takeover time for tournament selection, and extract the critical bounds beyond which the best representative will not take over its niche.

1) Evolution of the Best Classifier: We first model the evolution of the numerosity of the best classifier cl_b at time $t+1$, $n_{b,t+1}$, given the numerosity of the classifier at time t, $n_{b,t}$, which will

- increase if the GA is applied to the niche, cl_b is selected to participate in the tournament, and another classifier in the population is selected for deletion.
- decrease if (i) the GA is applied to the niche, cl_b is not selected to participate in the tournament, but it is selected for deletion; or if (ii) the GA is applied to another niche, and cl_b is selected for deletion.

- remain the same otherwise.

More formally,

$$n_{b,t+1} = \begin{cases} n_{b,t}+1 & \frac{1}{m}\left[1-\left(1-\frac{n_{b,t}}{n}\right)^s\right]\left(1-\frac{n_{b,t}}{N}\right), \\ n_{b,t}-1 & \frac{1}{m}\left(1-\frac{n_{b,t}}{n}\right)^s\frac{n_{b,t}}{N}+\frac{m-1}{m}\frac{n_{b,t}}{N}, \\ n_{b,t} & \text{otherwise.} \end{cases}$$

Grouping the above equations we can derive the expected numerosity of cl_b,

$$n_{b,t+1} = n_{b,t} + \frac{1}{m}\left[1-\left(1-\frac{n_{b,t}}{n}\right)^s\right]\left(1-\frac{n_{b,t}}{N}\right) - \frac{1}{m}\left(1-\frac{n_{b,t}}{n}\right)^s\frac{n_{b,t}}{N} - \frac{m-1}{m}\frac{n_{b,t}}{N}, \tag{34}$$

from which we obtain

$$n_{b,t+1} = n_{b,t} + \frac{1}{m}\left[1-\left(1-\frac{n_{b,t}}{n}\right)^s\right] - \frac{n_{b,t}}{N}. \tag{35}$$

Following the same procedure used for proportionate selection, we develop the following expression, which represents the takeover time in tournament selection (see Appendix B for derivation details):

$$t = mN\int_{P_0}^{P_F}\frac{1}{1-\left(1-\frac{P_t}{1+P_t(1-m)}\right)^s - mP_t}dp. \tag{36}$$

The above integral cannot be solved in general for any value of s and m. Nonetheless, note that this analysis still provides essential information since (i) it permits to calculate particular expressions of the takeover time and (ii) enables the derivation of the conditions for extinction of starved niches. With the aim of showing some particular cases of the takeover time in tournament selection, the next section provides (i) a closed-form solution of the integral for problems with a single niche and any selection pressure s and (ii) a closed-form solution for problems with two niches ($m = 2$) and tournament size $s = 2$, since $s = 2$ is the lowest pressure that can be applied.

2) Particular Expressions of the Takeover Time for Tournament Selection: In this section, we use Equation 36 to derive some particular expressions of the takeover time. Expressions for other tournament sizes and niche sizes can be computed by replacing the corresponding values into Equation 36.

Number of Niches m=1

Replacing $m = 1$ into equation 36 we obtain the following expression

$$t = N\int_{P_0}^{P_F}\frac{1}{1-(1-P_t)^s - P_t}dp = N\left[\ln\left(\frac{1-P_0}{1-P}\right) + \frac{1}{s-1}\ln\left[\frac{1-(1-P)^{s-1}}{1-(1-P_0)^{s-1}}\right]\right]. \tag{37}$$

Thence, the takeover time of cl_b for tournament selection is

$$t^*_{TS_{m=1}} = N\left[\ln\left(\frac{1-P_0}{1-P}\right) + \frac{1}{s-1}\ln\left[\frac{1-(1-P)^{s-1}}{1-(1-P_0)^{s-1}}\right]\right], \tag{38}$$

which is the same expression obtained in the model derived in [46], which was derived under the assumption of having a single niche in the system.

Number of Niches m=2, Tournament Size s=2

Substituting m and s into equation 36, we obtain

$$t = 2N \int_{P_0}^{P_F} \frac{1}{1 - \left(1 - \frac{P_t}{1-P_t}\right)^s - 2P_t} dp = 2N \left[\frac{1}{P_0} - \frac{1}{P_F} + \frac{1}{2} \ln \frac{1-2P_0}{1-2P_F}\right]. \quad (39)$$

Then, the takeover time for tournament selection for $m = 2$ and $s = 2$ is

$$t_{TS_{m=2,s=2}} = 2N \left[\frac{1}{P_0} - \frac{1}{P_F} + \frac{1}{2} \ln \frac{1-2P_0}{1-2P_F}\right], \quad (40)$$

which depends linearly on the initial and the final proportion of the best classifier in the population and logarithmically on the difference between them. However, it does not depend on any scale between the fitness of cl_b and cl_o. Moreover, as $P_F > P_o$, the takeover time is always positive; this indicates that the best classifier always will be able to takeover its niche, regardless of the imbalance ratio of the problem. Note that this analysis has been made for the lowest possible selection pressure. Therefore, this conclusion can be extended to any tournament size for problems with two niches.

Finally, we compare the conclusions provided by this analysis with those obtained with proportionate selection. In IX-B2, we theoretically demonstrated that, with proportionate selection, the best classifier would not be able to take over its niche if $\rho \geq 0.5$ for problems with two niches. Thus, tournament selection appears to be more robust in highly imbalanced data sets when the fitness separation between accurate and over-general classifiers is low.

This subsection provided some specific expressions of the takeover time for tournament selection. Note that, although the general closed-form solution could not be extracted, the analysis is still crucial since it enables us to identify the conditions for niche extinction, which are derived in the following subsection.

3) Conditions for Starved Niches Extinction under Tournament Selection: To derive the conditions for niche extinction for tournament selection, we consider the differential equation obtained during the derivation of the model (see Equation 35) and require that the increase in the numerosity of the best classifier be negative. That is,

$$\frac{1}{m}\left[1 - \left(1 - \frac{n_{b,t}}{n}\right)^s\right] - \frac{n_{b,t}}{N} < 0, \quad (41)$$

which can be rewritten as

$$1 - m\frac{n_{b,t}}{N} < \left(1 - \frac{n_{b,t}}{n}\right)^s. \quad (42)$$

This expression depends on the number of niches m, the number of accurate classifiers in the niche $n_{b,t}$, the niche size n, and the population size N. Note that the left-most term decreases linearly with m, whilst the right-most term decreases exponentially with s. Therefore, the condition will be satisfied, i.e., the best classifier will not be able to take over its niche, for low values of s combined with moderate and large number of niches m.

D. Experimental Validation of the Takeover Time Models

Here, we experimentally validate (i) the theoretical models of takeover time and (ii) the conditions for extinction of starved niches for both proportionate and tournament selection. For this purpose, we ran XCS on the parity problem setting the number of niches m of the system. We initialized the population with $P_0 \cdot N$ copies of

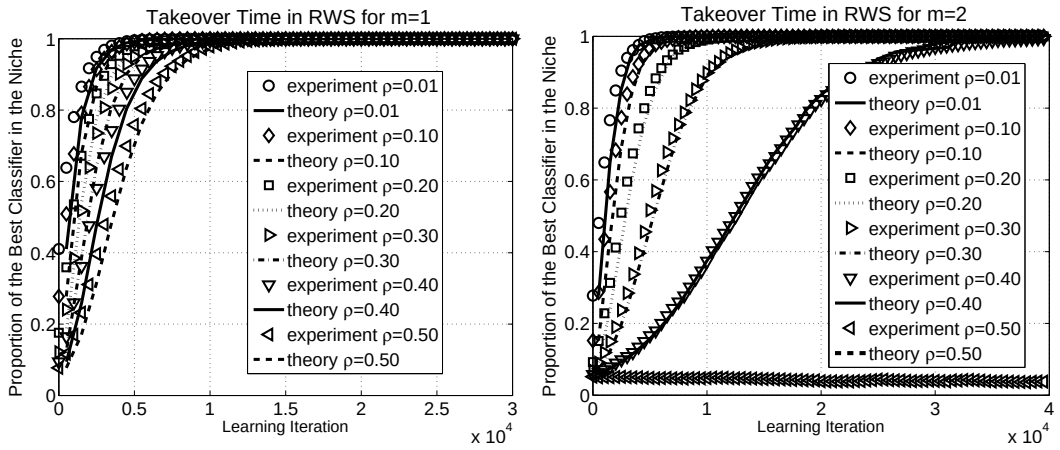
maximally accurate classifiers (equally distributed in the m niches) and $(1 - m \cdot P_0) \cdot N$ copies of an over-general classifier that appeared in all the niches. The prediction error of the over-general classifier was deterministically calculated as $\epsilon_{ovg} = \epsilon_0 \left(\frac{\rho}{\alpha}\right)^{\nu}$. In our experiments, we set α=0.1 and $\nu = 10$. Note that varying ρ, we are changing the fitness scaling between cl_o and cl_b. This could be equivalently done by maintaining ρ and varying ν, as done in [47].

Fig. 8 shows the evolution of the proportion of the best classifier in one of the niches for RWS and (a) m=1, (b) m=2, and (c) m=3 number of niches. The empirical data are averages over 50 runs. According to the model, we computed the proportion of the best classifier in the population. Therefore, the average of this proportion would tend to $1/m$, as approximately the same resources would be placed in each niche. In the figure, we plot the proportion of the best classifier in the niche, which ranges from 0 to 1. Fig. 8 shows a perfect match between the theory and the empirical results. It also shows that, as predicted by the models derived in Section IX-C, the ratio of the accuracy of the over-general classifier to the accuracy of the best classifier is a crucial aspect. For the problem with two niches, the best classifier could not take over its niche if $\rho \geq 0.5$ (see Fig. 8(b)), as predicted by the niche extinction model provided in Equation 32. This behavior is also present in the problem with three niches 8(c), where neither $\rho = 0.4$ nor $\rho = 0.5$ let the best classifiers take over their niches.

Fig. 9 shows the evolution of the proportion of the best classifier in the niche for TS and (a) m=1, (b) m=2, and (c) m=3 number of niches. For the problem with one niche, we depict the selection pressures of s=$\{$1,2,10$\}$. Fig. 9(a) shows a perfect match between the theoretical model and the experiments. Tournament selection was not influenced by the ratio of the accuracy of the best classifier to the accuracy of the over-general classifier. That is, we ran experiments with different values of ρ, obtaining equivalent results to those plotted in the figure. Moreover, it is also shown that increasing the tournament size resulted in faster convergence times. For s>10, the takeover time barely decreased (these curves are not depicted in the figure for clarity).

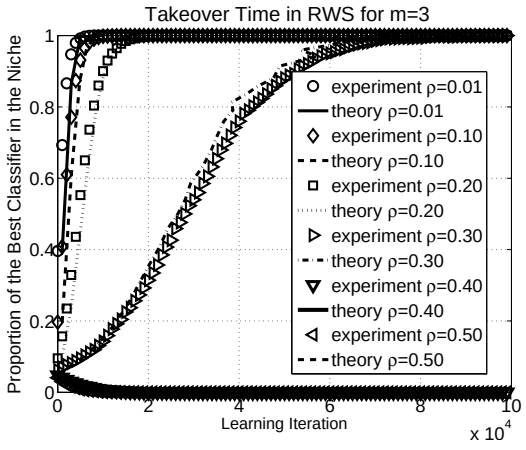
For the problems with two and three niches, we plot the evolution of the best classifier under tournament selection for s=2 and s=3 (see Fig. 9(b) and 9(c)). The theoretical model was calculated for each case by replacing m and s into Equation 36 and solving the integral. Again, the theory nicely approximates the experimental results. For m=2, the best classifier could take over its niche, even for the lowest possible selection pressure. For m=3, the best classifier could take over its niche only for $s \geq 3$. This experimental evidence agrees with the niche extinction model supplied in Equation 42. That is, as the number of niches increases, the selection pressure needs to be stronger to let the best classifier emerge, regardless of the initial proportion of this classifier in the population.

The overall analysis also permits to compare the two selection approximations and relate them to the class imbalance problem. For low values of ρ, that is, when the fitness of the best classifiers is much higher than the fitness of the over-general classifiers, proportionate selection can yield the fastest takeover times. Therefore, proportionate selection appears as the most appealing alternative for domains in which there is a high separation among the fitness of accurate and inaccurate rules. As pointed out in [47], in balanced domains, this can be easily done by tuning the fitness pressure ν. Nonetheless, in imbalanced domains, the error of over-general classifiers decreases with the imbalance ratio (see Equation 2). In these cases, proportionate selection may promote the existence of over-general



(a) m=1

(b) m=2



(c) m=3

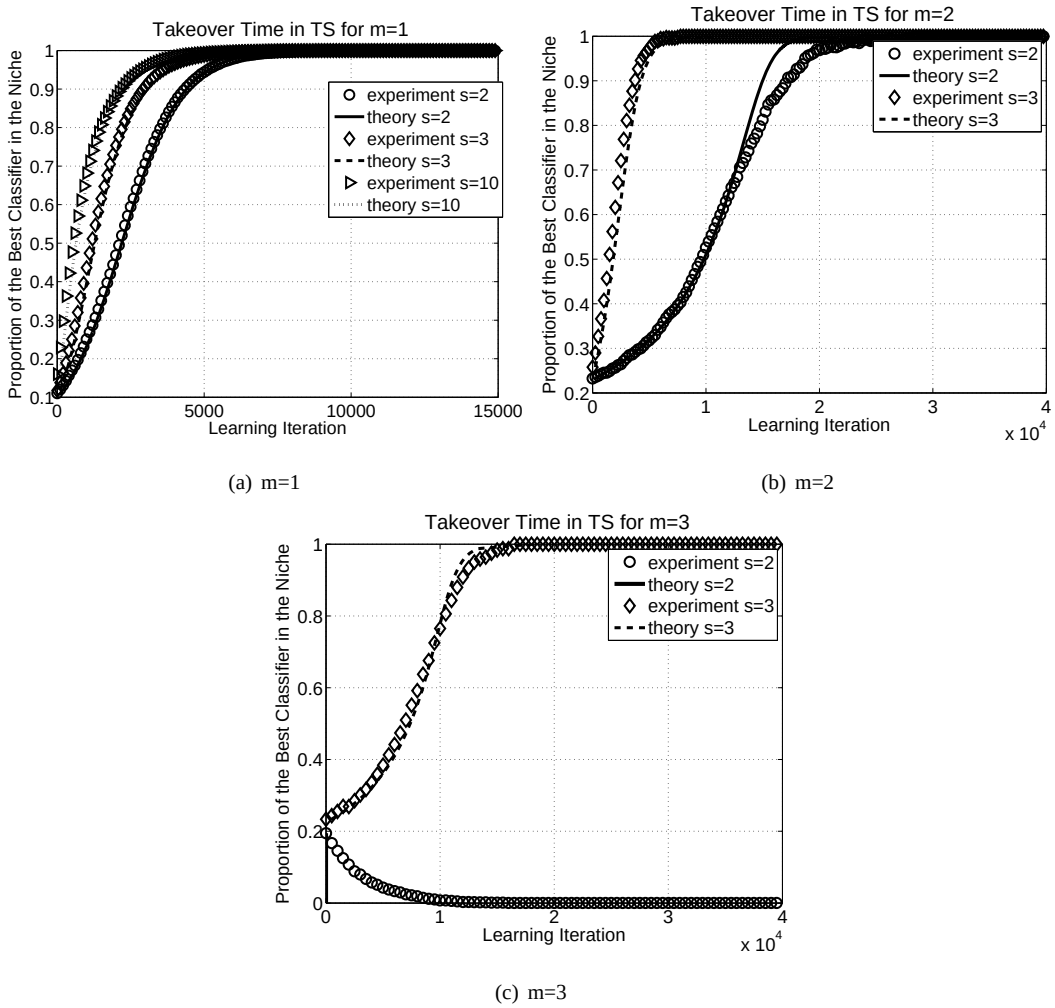
Fig. 8. Takeover time in roulette wheel selection for (a) m=1, (b) m=2, and (c) m=3 and ρ={0.01,0.10,0.20,0.30,0.40,0.50}.

classifiers. Thence, tournament selection appears to be the most robust selection scheme for imbalanced domains, provided that s is sized properly. A combination of both schemes seems to be an attractive alternative to deal with new real-world problems with unknown characteristics.

With the study of the takeover time and the conditions of extinction of starved niches, we completed the analysis of the different facets proposed in Section IV-D. Facetwise models have provided key insights and points of view of the problem. The next section, integrates all these models, provides configuration guidelines based on the lessons learned from them, and shows that, following these recommendations, XCS is able to solve highly imbalanced problems that previously eluded solution.

X. Lessons Learned from the Models

In this section, we use qualitative arguments to integrate the different models and extract lessons from the whole design decomposition and facetwise analysis. Then, we use the derived facetwise analysis as a tool for designing



(a) m=1

(b) m=2

(c) m=3

Fig. 9. Takeover time in tournament selection for (a) m=1, (b) m=2, and (c) m=3.

a set of guidelines that should be satisfied to warrant that XCS will be able to extract novel knowledge from rare classes. We experimentally show that, if the system is configured according to these recommendations, XCS is able to solve problems with large imbalance ratios that previously eluded solution. More specifically, we show that XCS with a proper configuration solves the multiplexer problem [6], one of the most used benchmarks in the LCSs field, with large imbalance ratios.

A. Patchquilt Integration of the Facetwise Models

Along this paper, we have studied the five subproblems, identified by the design decomposition, that may impair XCS from learning from rare classes. Facetwise models explained the behavior of different components of XCS. Now, we integrate the models in a general framework and highlight the lessons derived from each particular model and, in general, from their interaction. This integration permits us to (i) identify under which cases XCS will not

be able to learn from the minority class and (ii) establish configuration recommendations to ensure convergence if possible. To achieve this, we revisit the models from the most restrictive one to the less restrictive one, setting the three steps that need to be guaranteed to ensure convergence and pointing out several configuration guidelines.

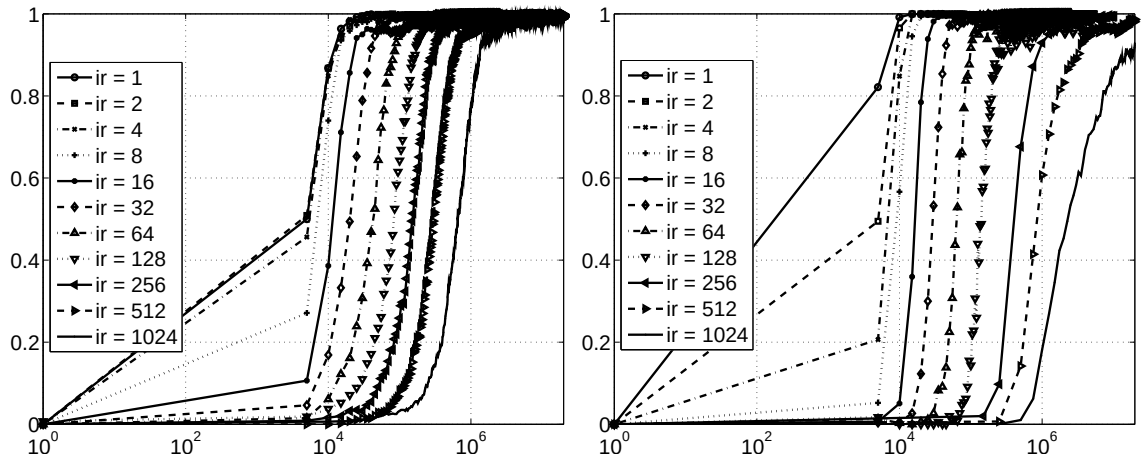
1) Niche extinction models (see Section IX) set the conditions on the maximum imbalance ratio beyond which XCS will not be able to learn from the minority class for proportionate and tournament selection (see Equations 32 and 42). If the requirements are met, takeover time models predict the convergence time inside each niche, that is, the number of learning iterations that an accurate, maximally general representative will need to take over its niche. Satisfying the requirements identified by the extinction time models is a necessary but not sufficient condition. That is, these models indicate that, if the identified restrictions are met, representative classifiers will take over their niche. Therefore, we have to guarantee that, at some point, these representatives are fed into the population.
2) Classifiers parameters have to be correctly estimated by one of the methods presented in Section V. Otherwise, XCS will not be able to distinguish between over-general and accurate classifiers. We experimentally showed that the Widrow-Hoff rule provided very accurate estimates of classifier parameters if β was properly set. For this reason, we took this approach in our experiments and proposed an heuristic that automatically sets the value of β ensuring that the error estimate of over-general classifiers is close to the theoretical value (see Section VII-E).
3) If the above two conditions are satisfied, we can ensure convergence by either (i) sizing the population in function of the imbalance ratio or (ii) setting θ_{GA} depending on the imbalance ratio according to the models evolved in Sects. VI and VII. It is worth noting that the models work independently of whether covering is able to provide the initial population with schemas of starved niches, as identified in the model derived in Section V.

In the next section, we show that, if the recommendations derived from the models are followed, XCS can solve extremely imbalanced data sets.

B. Solving Problems with Large Imbalance Ratios

In this section, we solve a typical benchmark in the realm of LCSs, the multiplexer problem, which is artificially unbalanced to analyze whether the original concepts can be learned even when one of the classes is progressively unbalanced. We adopt the multiplexer problem since it is a complex problem that has been widely used to test the performance of XCS and LCSs in general. As proceeds, we first give a brief description of the problem, present the configuration of XCS, and analyze the experimental results.

1) The Imbalanced Multiplexer Problem: The multiplexer problem [6] is a completely-balanced problem defined for binary strings of size ℓ, where the first $\log_2 \ell$ bits are the address bits and the remaining bits are the position bits. Then, the output is the value of the position bit referred by the decimal value of the address bits. For example, in the 6-bit multiplexer (i.e., ℓ = 6), f(00 1001) = 1 or f(10 0101) = 0. The concept complexity of the multiplexer is controlled by the input length ℓ. To obtain the optimal classification model, learners need to discover the linkages



(a) %[O] (b) TP rate · TN rate

Fig. 10. Evolution of (a) the proportion of the optimal population and (b) the product of TP rate and TN rate in the 11-bit multiplexer with ir={1,2,4,8,16,32,64,128,256,512,1024}.

between the address bits and the position bits, which increase exponentially with ℓ. That is, the system is expected to evolve populations that consist of $2^{log_2 \ell+2}$ optimal classifiers with all address bits and the corresponding position bit specified and all the other bits set to '#' (see [39] for further details). For this reason, multiplexers pose a big challenge to many well-known learners, especially as ℓ increases.

To obtain an imbalanced problem, examples of the class labeled as '1' are under-sampled according to the given imbalance ratio. Thence, the problem contains $2^{log_2 \ell+1}$ starved niches and $2^{log_2 \ell+1}$ nourished niches. In [44], it is shown that standard XCS is not able to solve the imbalanced multiplexer problem for $ir > 32$. In the next section, we show that, when properly configured, XCS can solve the multiplexer problem with much larger imbalance ratios.

2) Experimental Results: Before proceeding with the analysis of the results, we first explain the parameter settings used for XCS. We configured XCS with the default configuration, but using the typical deletion scheme of XCS, setting $N = 1\,000$ and probability of crossover $\chi = 0.8$, and using proportionate selection. The conditions required by the different models were satisfied as follows:

1) As we used proportionate selection, we needed to guarantee that $\rho < \frac{1}{m}$ to ensure convergence. Here, we show that this condition held for the highest imbalance ratio used in the experiments, that is, ir = 1024; then, this conclusion can be extended to lower imbalance ratios. To validate this inequality, as $\rho = \frac{\kappa_o}{\kappa_b}$, we first have to compute the accuracy of both the most over-general classifier κ_o and a representative classifier κ_b. Using Equation 2, we obtain that the error of the most over-general classifier is $\epsilon_o = 1.99$, from which we can compute the accuracy as indicated in Equation 1, i.e., $\kappa_o = 3.15 \cdot 10^{-3}$. The accuracy of the representative classifier κ_b is $\kappa_b = 1$. On the other hand, the 11-bit multiplexer consists of 32 niches; thence, $\frac{1}{m} = 3.15 \cdot 10^{-2}$. Therefore, we satisfy the condition that $\rho = 3.15 \cdot 10^{-3} < \frac{1}{m} = 3.15 \cdot 10^{-2}$, which indicates that, if discovered, representatives of starved niches will take over their niches. It is worth noting that similar results can be

achieved with tournament selection.

2) The classifier parameters are estimated according to the Widrow-Hoff rule, but setting β appropriately so that the error of over-general classifiers does not decrease rapidly to zero. For this purpose, we used the heuristic method mentioned in the previous section.
3) Finally, to guarantee that all the niches receive the same number of genetic opportunities, and so, warrant that representatives of starved niches will be created regardless of the imbalance ratio, we set $\theta_{GA} = n \cdot m \cdot ir$ as proposed in Section VIII.

Fig. 10 plots the results obtained by XCS in the imbalanced 11-bit multiplexer problem with imbalance ratios ranging from $ir = 1$ to $ir = 1024$. More specifically, Fig. 10(a) illustrates the evolution of the proportion of the optimal population %[O] [39] achieved by XCS. That is, XCS was expected to evolve 32 optimal classifiers, each one representing a different niche. In this way, we measured the capacity of XCS to generalize and obtain the best representative of each niche at high imbalance ratios. Moreover, Fig. 10(b) depicts the evolution of the product of TN rate and TP rate. Therefore, in addition to measure whether XCS evolved the optimal population, this figure visualizes whether the instances of the two classes were correctly predicted by the system. Note that we tested extremely imbalanced problems with imbalance ratios up to $ir = 1024$.

Fig. 10(a) shows that XCS was able to obtain 100% of the optimal population at any imbalance ratio with the same population size. This indicates that class imbalances did not reduce the generalization capabilities of XCS. Similarly, Fig. 10(b) illustrates that the product of TP rate and TN rate raised to 100% for all the tested imbalance ratios. Notice that before the analysis, XCS could only solve the problem for $ir \leq 32$ [44]. Hence, the lessons extracted from the design decomposition served to increase our understanding of the system and solve much more complex problems without introducing new mechanisms to the system.

The whole experimental and theoretical analysis performed herein highlights the importance of, previously to design new approaches to enhance a system whose behavior is only partially understood, really comprehend the underlying problems of the learning architecture. In this way, better approaches that focus on the actual problems of the system can be designed more effectively. Here, we showed that the increased understanding provided by the facetwise analysis enabled first generation XCS to solve problems that seemed to be initially intractable.

XI. Summary and Conclusion

In this paper, we analyzed the behavior of XCS when learning from domains that contain rare classes. As XCS learns a set of distributed solutions online, we investigated whether the system may lose, or may never discover, some sub-solutions whose representative examples are infrequently sampled to the system. XCS learning is driven by the interaction of several components, which makes difficult the derivation of models that explain the behavior of the system in its whole. For this reason, we followed the design decomposition approach to study the effect of class imbalances on different components of XCS. That is, we decomposed the problem of learning from imbalanced domains into five subproblems and derived simpler, tractable models that focused on explaining the behavior of concrete parts of the system. This enabled us to highlight several aspects—mainly related to classifier evaluation,

and creation, maintenance and growth of representatives of starved niches—that were crucial to guarantee that XCS extracts key knowledge from rare classes. In addition, the patchquilt integration of all these models permitted us to draw the *domain of applicability* of XCS in imbalanced domains. We derived critical bounds on the system behavior, identifying under which conditions the system would not be able to learn from the minority class. Moreover, the study resulted in several recommendations on the system configuration to effectively deal with rare classes. Finally, we showed that all the insights provided by this analysis served to solve new complex problems with large imbalance ratios which previously eluded solution. As example, we empirically demonstrated that XCS was able to solve the 11-bit multiplexer problem with large amounts of class imbalance provided that the system was properly configured according to the guidelines indicated by the models.

The importance of the lessons extracted from the whole analysis goes beyond the application of XCS to imbalanced domains. The analysis sets the conditions required to ensure complete solution sustenance—i.e., discovery, maintenance, and growth of all niches of the system—in problems where some niches are activated with less frequency than other niches. That is, the study clearly identified the challenges of learning a problem in which some of the niches are poorly represented and so may be overlooked by the system in detriment of more nourished niches. This is a common characteristic in real-world classification problems, regardless of whether the problem contains class imbalances or not. In these problems, the system tends to create small niches that cover instances that lay close to the class boundary. As the class boundary tends to concentrate a large proportion of examples of different classes, classifiers with very specific conditions tend to be created to approximate this decision boundary accurately. On the other hand, larger niches are created in regions of the feature space that are not close to the decision boundary. Therefore, in almost all real-world problems, there exists the competition between nourished niches, starved niches, and over-general classifiers that has been modeled along this paper.

Finally, let us highlight that, as we applied a design decomposition principle, the provided analysis is not restricted to XCS. In fact, we developed a framework in which several subproblems were analyzed separately and simplified models were provided. In all the derived models, we tried to keep the analysis as simple as possible and used intuitive arguments to patch the pieces together. This supplied a high flexibility and power to the theoretical framework, which can be adapted with low cost to model other Michigan-style LCSs or other online learning architectures that are based on a competition-collaboration scheme. For this purpose, models that are affected by the architecture change may be revisited and plugged again into the theoretical framework. Other models may still be valid; for example, takeover time models may still be accurate for most of the Michigan-style LCSs.

APPENDIX

In this section we provide the derivation details for proportionate selection (see Section IX-B) and tournament selection (see Section IX-C).

A. Takeover Time for Proportionate Selection

We start considering Equation 27, that is,

$$\frac{P_t(1-m\rho)+\rho}{P_t\left[(1-\rho m)-mP_t(1-\rho m)\right]}dp = \frac{1}{Nm}dt. \tag{43}$$

We integrate the left-most integral between the initial proportion P_0 of cl_b and the final proportion P_F of cl_b up to which cl_b has taken over the population, resulting in

$$\frac{t}{Nm} = \int_{P_0}^{P_F} \frac{1}{1-mP_t}dp + \frac{\rho}{1-m\rho}\int_{P_0}^{P_F} \frac{1}{P_t[1-mP_t]} = \tag{44}$$

$$= -\frac{1}{m}\ln\left(\frac{1-mP_0}{1-mP_F}\right) + \frac{\rho}{1-m\rho}\ln\left|\frac{P_F(1-mP_0)}{P_0(1-mP_F)}\right|. \tag{45}$$

Since $P_t < 1/m$, we can rewrite the expression as

$$\frac{t}{Nm} = -\frac{1}{m}\ln\left(\frac{1-mP_0}{1-mP_F}\right) + \frac{\rho}{1-m\rho}\ln\left(\frac{P_F(1-mP_0)}{P_0(1-mP_F)}\right), \tag{46}$$

from which we derive the takeover time of cl_b in roulette wheel selection

$$t^*_{rws} = Nm\left[\frac{1}{m}\ln\left(\frac{1-mP_0}{1-mP_F}\right) + \frac{\rho}{1-m\rho}\ln\left(\frac{P_F(1-mP_0)}{P_0(1-mP_F)}\right)\right] \tag{47}$$

B. Takeover Time for Tournament Selection

We start considering Equation 35,

$$n_{b,t+1} = n_{b,t} + \frac{1}{m}\left[1-\left(1-\frac{n_b}{n}\right)^s\right] - \frac{n_{b,t}}{N}. \tag{48}$$

As did for roulette wheel selection, we rewrite the formula above in function of the proportion P_t of classifiers cl_b in the whole population, i.e., $P_t = \frac{n_{b,t}}{N}$. From which we can calculate n_o and n as

$$n_o = N - mn_b = N - mNP_t = N(1-mP_t), \tag{49}$$

$$n = n_b + n_o = NP_t + N(1-mP_t). \tag{50}$$

Substituting equations 49 and 50 into equation 48, we obtain

$$P_{t+1} = P_t + \frac{1}{mN}\left[1-\left(1-\frac{P_t}{1+P_t(1-m)}\right)^s\right] - \frac{1}{N}P_t. \tag{51}$$

Assuming $\frac{dp}{dt} \approx P_{t+1} - P_t$, we derive

$$\frac{dp}{dt} \approx P_{t+1} - P_t = \frac{1}{mN}\left[1-\left(1-\frac{P_t}{1+P_t(1-m)}\right)^s - mP_t\right]. \tag{52}$$

That is,

$$\frac{1}{mN}dt = \frac{1}{1-\left(1-\frac{P_t}{1+P_t(1-m)}\right)^s - mP_t}dp. \tag{53}$$

If we integrate each side of the expression, we obtain

$$\frac{1}{mN}dt = \frac{t}{mN} = \int_{P_0}^{P_F} \frac{1}{1-\left(1-\frac{P_t}{1+P_t(1-m)}\right)^s - mP_t}dp \tag{54}$$

REFERENCES

[1] J. H. Holland, "Adaptation," in *Progress in Theoretical Biology*, R. Rosen and F. Snell, Eds., vol. 4. New York: Academic Press, 1976, pp. 263–293.

[2] ——, "Adaptive algorithms for discovering and using general patterns in growing knowledge-bases," *International Journal of Policy Analysis and Information Systems*, vol. 4, no. 3, pp. 245–268, 1980.

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

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

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

[6] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[7] ——, "Generalization in the XCS classifier system," in *3rd Annual Conference on Genetic Programming*. Morgan Kaufmann, 1998, pp. 665–674.

[8] L. Bull, E. Bernadó-Mansilla, and J. Holmes, Eds., *Learning Classifier Systems in Data Mining*, ser. Studies in Computational Intelligence. Springer, 2008.

[9] M. V. Butz, D. E. Goldberg, and P. L. Lanzi, "Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 452–473, 2005.

[10] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction," *IEEE Transactions on Evolutionary Computation*, vol. 10.1109/TEVC.2007.903551 (in press), 2008.

[11] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla, "Fuzzy-UCS: a michigan-style learning fuzzy-classifier system for supervised learning," *IEEE Transactions on Evolutionary Computation*, vol. (in press), 2008.

[12] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 28–46, 2004, also, IlliGAL technical report num. 2002011.

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

[14] M. Butz, D. E. Goldberg, P. L. Lanzi, and K. Sastry, "Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity," *Genetic Programming and Evolvable Machines*, vol. 8, no. 1, pp. 5–37, 2007.

[15] G. M. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[16] N. V. Chawla, N. Japkowicz, and A. Kolcz, Eds., *Special issue on learning from imbalanced datasets*, ser. SIGKDD Explorations Newsletter, 2004, vol. 6, no. 1.

[17] D. E. Goldberg, *The design of innovation: Lessons from and for competent genetic algorithms*, 1st ed. Kluwer Academic Publishers, 2002.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2000.

[19] P. Chan and S. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection," in *Knowledge Discovery and Data Mining*, 1998, pp. 164–168.

[20] G. M. Weiss and H. Hirsh, "Learning to predict rare events in event sequences," in *KDD'98: Fourth International Conference on Knowledge Discovery and Data Mining*, R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, Eds. New York, NY: AAAI Press, Menlo Park, CA, 1998, pp. 359–363.

[21] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analisis*, vol. 6, no. 5, pp. 429–450, November 2002.

[22] N. Japkowicz and J. Taeho, "Class imbalances versus small disjuncts," *SIGKDD Explorations*, vol. 6, no. 1, pp. 40–49, June 2004.

[23] G. M. Weiss and F. Provost, "Learning when training data are costly: the effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[24] T. Fawcett, "PRIE: A system for generating rulelists to maximize ROC performance," *Data Mining and Knowledge Discovery*, vol. 10.1007/s10618-008-0089-y (in press), 2008.

[25] D. R. Carvalho and A. A. Freitas, "A genetic-algorithm for discovering small-disjunct rules in data mining," *Applied Soft Computing*, vol. 2, no. 2, pp. 75–88, 2002.

[26] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, "Reducing the misclassification costs," in *Proceedings of the Eleventh International Conference on Machine Learning*, 1994, pp. 217–225.

[27] G. M. Weiss, "Timeweaver: A genetic algorithm for identifying predictive patterns in sequences of events," in *GECCO'09: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds. Morgan Kaufmann, 1999, pp. 718–725.

[28] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[29] G. Batista, R. Prati, and M. Monrad, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[30] S. García and F. Herrera, "Evolutionary under-sampling for classification with imbalanced data sets: Proposals and taxonomy," *Evolutionary Computation*, vol. in press, 2008.

[31] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced datasets," *Soft Computing Journal*, vol. 10.1007/s00500-008-0319-7 (in press), 2008.

[32] J. Holmes, "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*. Morgan Kaufmann, 1998, pp. 635–642.

[33] A. Orriols-Puig and E. Bernadó-Mansilla, "The class imbalance problem in learning classifier systems: A preliminary study," in *GECCO'05: Proceedings of the 2005 Genetic and Evolutionary Computation Conference workshop program*. Washington, D.C., USA: ACM Press, 2005, pp. 74–78.

[34] ——, "The class imbalance problem in UCS classifier system: Fitness adaptation," in *Proceedings of the 2005 Congress on Evolutionary Computation*, vol. 1. Edinburgh, UK: IEEE, 2005, pp. 604–611.

[35] M. V. Butz and S. W. Wilson, "An algorithmic description of XCS," in *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, ser. Lecture Notes in Artificial Intelligence, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds., vol. 1996. Springer, 2001, pp. 253–272.

[36] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *Neurocomputing: foundations of research*, pp. 123–134, 1988.

[37] M. V. Butz, K. Sastry, and D. E. Goldberg, "Strong, stable, and reliable fitness pressure in XCS due to tournament selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 1, pp. 53–77, 2005.

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

[39] T. Kovacs and M. Kerber, "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*. Springer-Verlag, 2001, pp. 80–99.

[40] M. V. Butz, D. E. Goldberg, and T. Tharankunnel, "Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy," *Evolutionary Computation*, vol. 11, no. 3, pp. 239–277, 2003.

[41] M. V. Butz, D. E. Goldberg, and P. L. Lanzi, "Bounding learning time in XCS," in *GECCO'2004: Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, ser. LNCS. Springer, 2004, pp. 739–750.

[42] D. Thierens, D. E. Goldberg, and A. G. Pereira, "Domino convergence, drift, and the temporal-salience structure of problems," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1998, pp. 535–540.

[43] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of Genetic Algorithms*, pp. 69–93, 2003.

[44] A. Orriols-Puig and E. Bernadó-Mansilla, "Bounding XCS parameters for unbalanced datasets," in *GECCO'06: Proceedings of the 2006 Genetic and Evolutionary Computation Conference*. ACM Press, 2006, pp. 1561–1568.

[45] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

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

[47] F. Kharbat, L. Bull, and M. Odeh, "Revisiting genetic selection in the XCS learning classifier system," in *Congress on Evolutionary Computation*. Edinburgh, UK: IEEE, 2005, pp. 2061–2068.