Improving the Performance of a Pittsburgh Learning Classifier System Using a Default Rule

Jaume Bacardit[†], **David E. Goldberg**[‡] and **Martin V. Butz**[‡]

[†]Intelligent Systems Research Group
Universitat Ramon Llull
Psg. Bonanova 8, 08022 Barcelona
Catalonia, Spain, Europe
jbacardit@salleURL.edu

[‡]Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
deg@uiuc.edu,butz@illigal.ge.uiuc.edu

Abstract

An interesting feature of encoding the individuals of a Pittsburgh Learning Classifier System as a decision list is the emergent generation of a default rule. However, performance of the system is strongly tied to the learning system choosing the correct class for this default rule. In this paper we experimentally study the use of an explicit (static) default rule. We first test simple policies for setting the class of the default rule, such as the majority/minority class of the problem. Next, we introduce some techniques to automatically determine the most suitable class.

1 Introduction

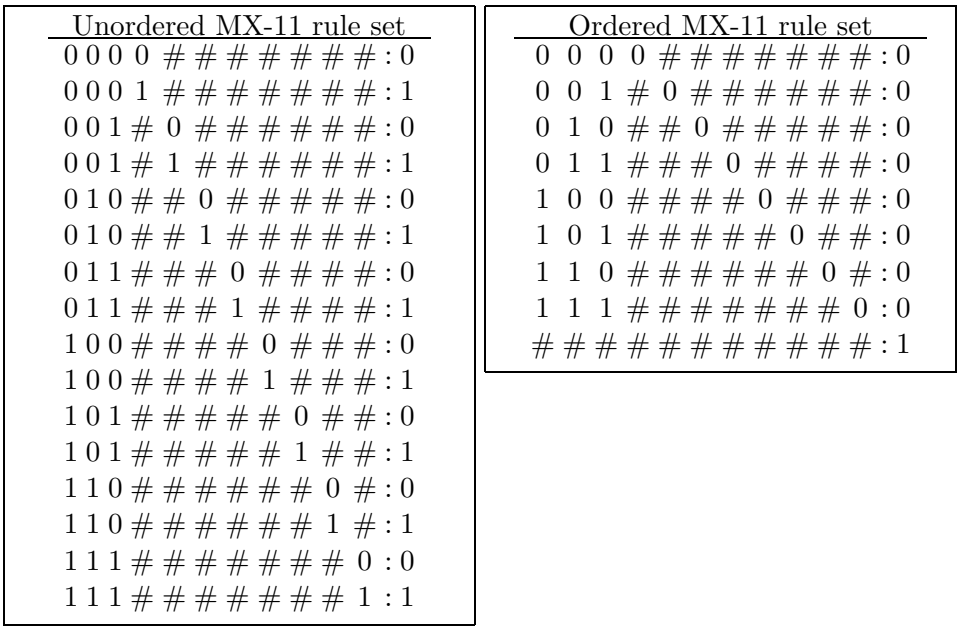
One of the ways to solve classification problems using a genetic algorithm (Holland, 1975; Goldberg, 1989) is called Pittsburgh approach (DeJong, Spears, & Gordon, 1993) or Pittsburgh learning classifier system. The individuals of this system encode a full and variable-length rule set and the solution proposed is the best individual of the population. There are several encoding options for an individual. One of them is coding an individual as a decision list (Rivest, 1987) (an ordered set of rules). If we apply this strategy in the evolutionary framework, often the system evolves a default rule. That is, a rule that matches any input instance.

Default rules can be very useful in combination with a decision list because the size of the rule set can be reduced significantly. For instance, for the 11-bit multiplexer we can obtain a rule set of 9 rules instead of 16 unordered ones, as represented in Figure 1. With a smaller rule set, the search space is reduced resulting in two potential advantages: (1) the learner has to learn less rules (representing only the other classes of the dataset) and (2) with a smaller rule set the system may be less sensitive to over-learning potentially increasing the test accuracy of the system.

The objective of this paper is to investigate the potential benefits of using an explicit and static default rule in a Pitt LCS. Along those lines, the question arises which is the best default class to use. Simple strategies may use the majority class. However, our tests show that dependent on the problem, the minority class may be better as the default class choice. Thus, we develop a mechanism that is able to automatically determine the best class for the default rule.

The rest of the paper is structured as follows: Section 2 shows some related work. Next, Section 3 describes briefly the main characteristics of the system used in this paper. Later, Section 4 illustrates the motivation of using a default rule, followed by the description and some illustrative

Figure 1: Unordered and ordered rule sets for the MX-11 domain



Unordered MX-11 rule set

0 0 0 0 # # # # # # # # : 0
0 0 0 1 # # # # # # # # : 1
0 0 1 # 0 # # # # # # # : 0
0 0 1 # 1 # # # # # # # : 1
0 1 0 # # 0 # # # # # # : 0
0 1 0 # # 1 # # # # # # : 1
0 1 1 # # # 0 # # # # # : 0
0 1 1 # # # 1 # # # # # : 1
1 0 0 # # # # 0 # # # # : 0
1 0 0 # # # # 1 # # # # : 1
1 0 1 # # # # # 0 # # # : 0
1 0 1 # # # # # 1 # # # : 1
1 1 0 # # # # # # 0 # # : 0
1 1 0 # # # # # # 1 # # : 1
1 1 1 # # # # # # # 0 # : 0
1 1 1 # # # # # # # 1 # : 1

Ordered MX-11 rule set

0 0 0 0 # # # # # # # # : 0
0 0 1 # 0 # # # # # # # : 0
0 1 0 # # 0 # # # # # # : 0
0 1 1 # # # 0 # # # # # : 0
1 0 0 # # # # 0 # # # # : 0
1 0 1 # # # # # 0 # # # : 0
1 1 0 # # # # # # 0 # # : 0
1 1 1 # # # # # # # 0 # : 0
: 1

results of the simple policies for the default rule in Section 5. After the simple policies, we describe the more sophisticated ones in Section 6. Section 7 shows the experimentation results of applying the previous described policies. Finally, Section 8 presents conclusions and further work.

2 Related work

We can find previous uses of a static default rule in the *LCS* field, although not in an explicit way: Classic Pitt-approach systems such as *GABIL* (DeJong, Spears, & Gordon, 1993) or GIL (Janikow, 1991), which perform concept learning (learning a concept from sets of positive/negative examples), implicitly have a default rule that covers the negative examples. The rules generated do not have an associated class because all of them cover the positive examples. However, there is no explicit policy to decide which set is the positive or negative in order to learn better. The decision comes from the definition of the dataset.

Looking at the machine learning field in general we find other examples of default rules. The C4.5rules system (Quinlan, 1993) uses an explicit default rule and like our system it generates a rule set acting as a decision list. To select the class for this default rule it uses the class that has less instances covered by the other rules in the rule set. This kind of approach seems feasible when we have induced the rule set beforehand, instead of using it during learning as our system does.

The *IREP* system (Cohen, 1995) induces, in order, the rules modelling each class of the problem (using the instances of the classes still to be learned as negative examples). The criteria of this global order is ascendent frequency of examples. Therefore, the default rule of this system uses a majority class policy.

3 Framework

GAssist (Bacardit & Garrell, 2004) is a Pittsburgh genetic-based machine learning system descendant of *GABIL* (DeJong, Spears, & Gordon, 1993). The system applies a near-standard *GA* that evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable-length rule set. Directly from GABIL we have taken the semantically correct crossover operator for variable-length individuals.

Dealing with variable-length individuals raises some important issues. One of the most important one is the control of the size of the evolving individuals (Soule & Foster, 1998). This control is achieved in GAssist using two different operators:

1. *Rule deletion.* This operator deletes the rules of the individuals that do not match any training example. This rule deletion is done after the fitness computation and has two constraints:
   (a) The process is only activated after a predefined number of iterations (to prevent an irreversible diversity loss)
   (b) The number of rules of an individual never goes below a threshold. This introduces some "neutral code" that can protect the individuals from the disruptive effect of the crossover operator.
2. *Minimum description length-based fitness function.* The minimum description length (*MDL*) principle (Rissanen, 1978) is a metric applied in general to a theory (being a rule set in this paper) which balances the complexity and accuracy of the rule set. In previous work we developed a fitness function based on this principle. A detailed explanation of the fitness function can be found in (Bacardit & Garrell, 2003).

The knowledge representation used for real-valued attributes is called *adaptive discretization intervals* rule representation (*ADI*) (Bacardit & Garrell, 2004). This representation uses the semantics of the *GABIL* rules (conjunctive normal form predicates), but applies non-static intervals formed by joining several neighbor discretization intervals. These intervals can evolve through the learning process splitting or merging among them potentially using several discretizers at the same time.

Parameters of the system are set as follows: Crossover probability 0.6; tournament selection; tournament size 3; population size 300; Individual-wise mutation probability 0.6; initial number of rules per individual 20; probability of "1" in initialization 0.75; Rule Deletion Operator: Iteration of activation: 5; minimum number of rules: number of active rules +3; MDL-based fitness function: Iteration of activation 25; initial theory length ratio: 0.075; weight relax factor: 0.9. ADI knowledge representation: split and merge probability: 0.05; reinitialize probability at initial iteration: 0.02; reinitialize probability at final iteration: 0; merge restriction probability: 0.5; maximum number of intervals: 5; set of uniform discretizers used: 4, 5, 6, 7, 8, 10, 15, 20 and 25 bins; iterations: maximum of 1500.

4 Motivation

In order to illustrate the benefits of the default rule, we show the results of running the system with no static default rule for the *Glass* problem from the *UCI* repository (Blake, Keogh, & Merz, 1998) in table 1. We used stratified ten-fold cross validation for the tests and a hundred random seeds for each fold (a total of 1000 runs, unlike the 15 seeds and 150 runs used in the rest of the paper). We can see the benefits of using a default rule and, more importantly, the benefits of choosing the correct class for the default rule.

Table 1: How the generation of a default rule can affect the performance in the *Glass* dataset

Runs generating a default rule	736
Runs not generating a default rule	264
Accuracy of runs with a default rule	66.98±8.00
Accuracy of runs without a default rule	66.27±7.79
Average accuracy of runs using class 1 as default rule	65.45±7.39
Average accuracy of runs using class 2 as default rule	67.76±7.81
Average accuracy of runs using class 3 as default rule	59.40±5.51
Average accuracy of runs using class 4 as default rule	66.18±8.70
Average accuracy of runs using class 5 as default rule	67.66±8.58
Average accuracy of runs using class 6 as default rule	64.48±7.36

Figure 2: Match process using an static default rule

Match process
Input : $RuleSet, Instance$
$Index = 0$
$Found = false$
While $Index < RuleSet.size$ and not $Found$ **Do**
  If $RuleSet.rule[Index]$ matches $Instance$ **Then**
    $Class = RuleSet.rule[Index].class$
    $Found = true$
  Else
    $Index++$
  EndIf
EndWhile
If not $Found$ **Then**
  $Class = DefaultClass$
EndIf
Output : Predict class $Class$ for instance $Instance$

5 Simple default rule

The implementation of the static default rule is very simple. Basically it affects only the matching function classifying any input instance by the default class if no rule matches the instance, represented by the pseudocode in Figure 2. Also, the default class is removed from the classes that can be used by the rest of the rules in the population, effectively reducing the search space. A general representation of the extended rule set is shown in Figure 3. For the specific case of two-class domains, the classification problem is transformed into a concept learning problem and the resulting knowledge representation is quite close to the ones used in other evolutionary concept learning systems like *GABIL* (DeJong, Spears, & Gordon, 1993) or *GIL* (Janikow, 1991).

In order to answer the question of which class is suitable for being the default class we start by experimenting with two simple policies: using the most and least frequent class in the domain. In Section 7 we can see the results of these tests for several datasets. Here we show the results (in Table 2 only of two datasets (*Glass* and *Ionosphere*), also from UCI. For *Glass* the best policy is using the majority class. For *Ionosphere* the best policy is using the minority class. The point of showing these two datasets is that it is very difficult to decide *a priori* which is the most suitable default rule class for each dataset. Also, we can see in the values of the train accuracy and the number of rules a hint about how we can combine the two policies to maximize the performance of the system. In Section 7 we show a simple combination consisting on choosing at the test stage

Figure 3: Representation of the extended rule set with the static default rule

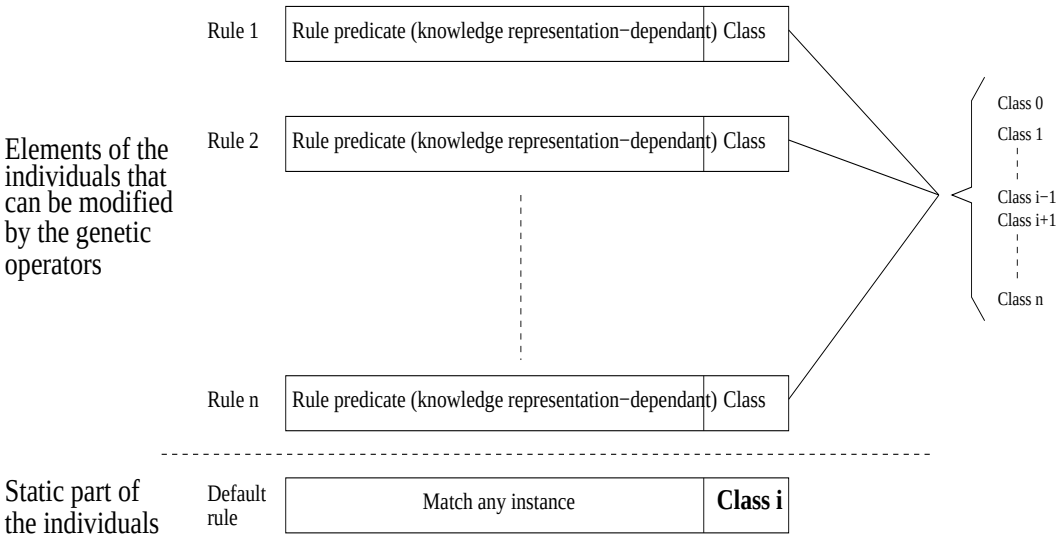


Table 2: Results using majority and minority policy for the default class in the *Glass* and *Ionosphere* datasets.

Domain	Def. Class. Policy	Train accuracy	Test accuracy	Number of rules
Glass	disabled	79.9±2.6	66.4±8.1	6.4±0.7
Glass	majority	83.2±1.6	69.5±6.9	6.6±0.8
Glass	minority	80.6±2.3	66.7±8.0	7.2±0.8
Ionosphere	disabled	96.0±0.6	92.8±3.6	2.3±0.6
Ionosphere	majority	95.7±0.8	90.0±4.4	5.7±1.2
Ionosphere	minority	96.8±0.7	93.0±3.7	2.6±0.8

the policy which has more train accuracy.

6 Automatically determined default class

Given that the majority class does not always suite best as default class, the next step is to modify the system to automatically determine the best default class. Our initial approach simply assigns a randomly chosen class as default class to each individual in the initial population. Additionally, we introduce a restricted mating mechanism to avoid crossover operations between individuals having different default classes. Having removed the default class from the rest of the rules, crossing individuals with different default classes may create lethals with high probability.

If we run the system in this setting, we mainly observed that all individuals with one default class take over the population. The question is if the system is able to choose the correct default class during the initial iterations. To answer this question, we show the evolution of the train accuracy and the number of rules for the *Ionosphere* tests described in the previous section in Figure 4. We can see that the train accuracy of the default class policy using the suitable class for this problem (that is, the minority class) is lower at the initial iterations than the accuracy of the majority class policy. Also, we can see the reason for the better test accuracy of the minority policy in the smaller (better generalized) rule set created by this policy.

Figure 4: Evolution of the train accuracy and the number of rules for the Ionosphere problem using majority/minority default class policies

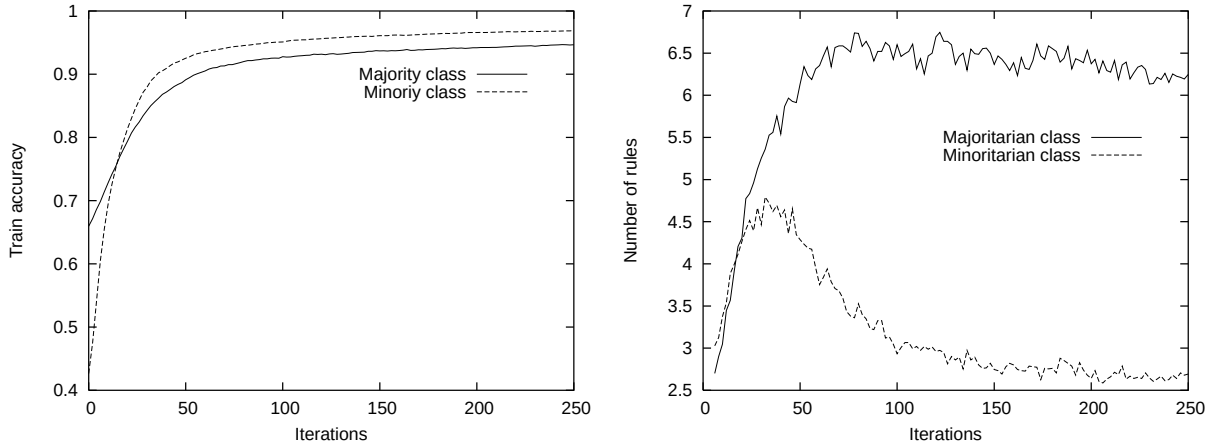


Figure 5: Pseudocode for the niched tournament selection

Niched tournament selection
Input : $Population, PopSize, NumNiches, TournamentSize$
$NextPopulation = \emptyset$
For $i = 1$ to $NumNiches$
    $ProportionNiche[i] = PopSize/NumNiches$
EndFor

For $i = 1$ to $PopSize$
    $Niche$ = select randomly a niche based on ProportionNiche
    $ProportionNiche[Niche] - -$
    Select $TournamentSize$ individuals from $Population$ belonging to $Niche$
    $winner$=Apply tournament
    Add $winner$ to $NextPopulation$
EndFor
Output : $NextPopulation$

Thus, it appears necessary to introduce an additional niching mechanism that preserves individuals for all default classes until the system has learned enough to decide correctly on the best default class. This niching is achieved using a modified tournament selection mechanism, inspired in (Oei, Goldberg, & Chang, 1991) in which the individuals participating in each tournament are forced to belong to the same class. Also, each default class has an equal number of tournaments. This niched tournament selection is represented by the pseudocode in Figure 5. The tournament with niche preservation is used until the best individuals of each default class have similar train accuracy. After this point, the niching is disabled and the system chooses freely among the individuals. Specifically, we compute for each niche the average accuracy for the last 15 iterations of its best individual. When the standard deviation of all these averages is smaller than 0.5%, we disable the niched tournament selection.

Table 3: Features of the datasets used in the experimentation of this paper

Dataset Properties											
Domain	#Inst.	#Attr.	#Real	#Nom.	#Cla.	Dev.cla.	Maj.cla.	Min.cla.	MV Inst.	MV Attr.	MV values
bpa	345	6	6	—	2	7.97%	57.97%	42.03%	—	—	—
bps	1027	24	24	—	2	1.60%	51.61%	48.39%	—	—	—
bre	699	9	9	—	2	15.52%	65.52%	34.48%	2.29%	1	0.23%
gls	214	9	9	—	6	12.69%	35.51%	4.21%	—	—	—
h-s	270	13	13	—	2	5.56%	55.56%	44.44%	—	—	—
ion	351	34	34	—	2	14.10%	64.10%	35.90%	—	—	—
lrn	648	6	4	2	5	14.90%	45.83%	1.54%	—	—	—
mmg	216	21	21	—	2	6.01%	56.02%	43.98%	—	—	—
pim	768	8	8	—	2	15.10%	65.10%	34.90%	—	—	—
son	208	60	60	—	2	3.37%	53.37%	46.63%	—	—	—
thy	215	5	5	—	3	25.78%	69.77%	13.95%	—	—	—
veh	846	18	18	—	4	0.89%	25.77%	23.52%	—	—	—
wdbc	569	30	30	—	2	12.74%	62.74%	37.26%	—	—	—
wine	178	13	13	—	3	5.28%	39.89%	26.97%	—	—	—
wpbc	198	33	33	—	2	26.26%	76.26%	23.74%	2.02%	1	0.06%

7 Results

In this section, we show the results of comparing the three policies tested for the default class (*majority,minority,auto* to the original system (*orig*) with emergent default rule. The tests include 15 datasets used previously in (Bacardit & Garrell, 2004), summarized in table 3. Each dataset has been partitioned into training/test sets using stratified ten-fold cross-validation (Kohavi, 1995), and having for each fold the tests repeated 15 times.

Table 4 shows the results for these tests, also including a fifth configuration (*majority+minority*): Choosing for the test stage the majority/minority policy that obtained more training accuracy. This configuration usually chooses the correct policy (although there are some exceptions, like *bpa*). These results were analyzed using pair-wise statistical t-tests with Bonferroni correction to determine how many times each method could significantly outperform or be outperformed by the other methods. These statistical tests are summarized in table **??**.

A second set of tests was performed increasing the population size from 300 to 400. The rationale of these tests was to provide a fair learning environment for the *auto* setup. Because of the additional niching process, only a higher population size can guarantee that each niche has enough individuals to learn correctly. The results are shown in table 6. The summary of the statistical t-tests applied to these results is in table 7.

We can extract some interesting observations from these results. First of all, all methods using a default rule obtain better accuracy on average than the original system. Moreover, we can see how the policy using an automatically determination of default class benefits from the population size increase, which matches the usual requirements of niching methods.

Also, we can see how the only method that degrades performance when we increase the population size is the majority class policy, suggesting that the system is sensitive to over-learning in domains where the majority class policy is not suitable. The larger average number of rules and the better training accuracy of the solutions generated by this policy confirm the overlearning problem compared to all other non-composed policies. The statistical tests show us how only the auto and major+minor policies show a robust behavior.

Table 4: Results of the tests comparing the studied default class policies to the original configuration using pop. size 300

Domain	Result	Default rule policy				
		Disabled	Major	Minor	Auto	Major+Minor
bpa	Train	78.6±1.6	81.4±1.3	80.1±1.6	80.8±1.4	81.4±1.3
	Test	63.8±7.4	62.9±7.8	65.2±6.5	64.0±6.9	62.9±7.8
	#rules	6.7±1.0	8.9±1.4	8.3±1.5	8.5±1.6	8.9±1.4
bps	Train	84.8±0.9	86.0±0.7	86.8±0.7	86.6±0.7	86.8±0.7
	Test	80.1±3.9	81.2±3.6	81.5±3.6	81.4±3.7	81.5±3.6
	#rules	5.1±0.4	6.1±1.1	5.7±0.9	5.6±0.8	5.7±0.9
bre	Train	97.7±0.3	98.2±0.3	98.4±0.3	98.4±0.3	98.4±0.3
	Test	95.9±2.2	95.0±2.5	95.7±2.0	95.6±2.2	95.7±2.0
	#rules	2.6±0.7	5.8±1.2	3.2±0.6	3.3±0.7	3.2±0.6
gls	Train	79.9±2.6	83.2±1.6	80.6±2.3	79.0±1.8	83.2±1.6
	Test	66.4±8.1	69.5±6.9	66.7±8.0	66.9±7.4	69.5±6.9
	#rules	6.4±0.7	6.6±0.8	7.2±0.8	6.9±0.9	6.6±0.8
h-s	Train	89.8±1.2	91.6±0.9	92.1±0.8	91.9±0.9	92.1±0.8
	Test	79.5±6.2	79.3±6.4	81.3±6.8	81.3±6.1	81.3±6.8
	#rules	6.7±0.9	7.6±1.2	7.3±1.2	7.4±1.3	7.3±1.2
ion	Train	96.0±0.6	95.7±0.8	96.8±0.7	96.8±0.7	96.8±0.7
	Test	92.8±3.6	90.0±4.4	93.0±3.7	93.1±3.9	93.0±3.7
	#rules	2.3±0.6	5.7±1.2	2.6±0.8	2.6±0.7	2.6±0.8
lrn	Train	75.2±1.9	76.8±0.8	75.4±1.4	75.4±1.0	76.8±0.8
	Test	68.5±4.7	68.9±5.7	68.9±4.5	68.6±5.6	68.9±5.7
	#rules	8.5±1.9	9.6±1.9	9.2±1.9	8.6±1.7	9.6±1.9
mmg	Train	79.7±1.8	83.2±1.3	83.1±1.3	83.0±1.4	83.2±1.3
	Test	66.2±7.8	68.9±8.3	67.8±8.4	66.8±9.0	68.9±8.3
	#rules	6.5±0.8	6.7±0.9	6.7±0.8	6.6±0.9	6.7±0.9
pim	Train	79.7±0.9	81.3±0.8	80.9±0.7	81.1±0.8	81.3±0.8
	Test	74.7±4.7	75.4±4.8	75.0±4.7	75.0±4.5	75.4±4.8
	#rules	5.2±0.4	6.2±1.0	5.6±0.8	6.1±1.0	6.2±1.0
son	Train	92.2±1.6	96.1±1.2	94.8±1.4	95.5±1.4	96.1±1.2
	Test	72.6±11.5	77.0±9.0	76.1±9.7	76.1±9.3	77.0±9.0
	#rules	6.7±1.1	7.6±1.4	7.7±1.3	7.4±1.1	7.6±1.4
thy	Train	97.4±1.0	98.4±0.7	98.4±0.7	98.1±0.8	98.4±0.7
	Test	91.9±5.6	92.8±4.8	92.3±5.3	92.2±5.6	92.8±4.8
	#rules	5.2±0.4	5.7±0.6	5.4±0.5	5.5±0.6	5.7±0.6
veh	Train	71.1±2.2	73.5±1.4	73.5±1.4	72.0±1.5	73.5±1.4
	Test	66.4±4.7	68.1±4.5	67.4±4.9	67.5±4.7	68.1±4.5
	#rules	6.6±1.2	9.3±2.0	9.9±1.6	8.0±1.8	9.3±2.0
wdbc	Train	97.2±0.8	97.8±0.6	97.8±0.6	97.8±0.7	97.8±0.6
	Test	94.1±3.0	94.2±3.1	94.0±3.0	94.3±3.1	94.2±3.1
	#rules	4.3±1.1	4.6±0.9	4.4±1.0	4.5±1.0	4.6±0.9
wine	Train	99.4±0.5	99.7±0.4	99.9±0.3	99.6±0.4	99.9±0.3
	Test	92.7±5.9	93.3±6.2	92.2±6.3	93.9±5.9	92.2±6.3
	#rules	3.8±0.7	3.6±0.6	4.1±0.5	3.8±0.6	4.1±0.5
wpbc	Train	84.3±3.0	89.4±2.0	86.4±3.4	88.7±2.3	89.4±2.0
	Test	76.0±7.3	75.8±7.4	72.6±8.5	75.2±7.5	75.8±7.4
	#rules	2.8±0.8	3.8±0.9	4.2±1.2	3.6±1.0	3.8±0.9
ave.	Train	86.9±9.0	88.8±8.4	88.3±8.8	88.3±9.0	89.0±8.5
	Test	78.8±11.4	79.5±10.7	79.3±11.0	79.5±11.3	79.8±10.9
	#rules	5.3±1.8	6.5±1.8	6.1±2.1	5.9±1.9	6.1±2.1

Table 5: Summary of the statistical t-tests applied to the experimentation results of popsize 300, with a confidence level of 0.05. Cells in table count how many times the method in the row significantly outperforms the method in the column.

Policy	Disabled	Major	Minor	Auto	Major+Minor	Total
Disabled	-	2	1	0	0	3
Major	3	-	2	1	0	6
Minor	2	2	-	0	0	4
Auto	2	1	1	-	0	4
Major+Minor	4	2	2	1	-	9
Total	11	7	6	2	0	

8 Conclusions and Future Work

In this paper we have tested some methods that extend the rule-based and decision-list-style knowledge representations for a Pittsburgh Learning Classifier System by using a static default rule. This kind of systems tend to generate an emergent default rule, which can increase the performance of the system. By forcing the representation of a default rule, we intended to guarantee these positive effects.

Simple policies such as using the majority/minority class as the default class perform quite well compared to the original system. However, they perform poorly on certain datasets somewhat showing a lack of robustness. We can almost integrate the best results of both policies by using the simple heuristic of selecting the policy with more training accuracy. This mechanism introduces a good performance boost, but doubles the run-time.

For this reason, we have developed a mechanism that decides automatically the class for the default rule. This technique works by integrating in a single population individuals using all possible default classes, and letting them compete among themselves. This approach has a problem, however, which is providing a fair competition framework, because each default rule can have a different learning rate. In order to achieve this fairness, we use a niched tournament selection that guarantees that all niches (different default rules) survive in the population until they can compete successfully by themselves. This automatic mechanism performs best when we increase the population size, which is an usual requirement in most systems that use niching, because we have to guarantee that each niche has enough individuals to ensure building block supply and thus successful and reliable learning.

The increase in population size for the majority/minority policies, however, showed no performance increase or even some performance decrease, suggesting the amplification of the policy weaknesses This weaknesses are derived from overlearning, which is reflected in the larger training accuracy and larger average rule set sizes and also on the statistical tests.

Although the automatic policy does not outperform the major+minor policy, the accuracy difference is quite small in most datasets and the computational cost is significantly lower. Therefore, it appears that in most situations the automatic policy is the best method.

As further work, additional statistical tests are necessary to determine the best policy or, at least, the most robust one. Also, we think that it is interesting to study how we can mix the knowledge of individuals across different niches, which is disabled in the current automatic policy to avoid creating lethals. If we can find a feasible mechanism to perform this recombination process, it is quite probable that we can reduce the currently increased population size requirements of the

Table 6: Results of the tests comparing the studied default class policies to the original configuration using pop. size 400

Domain	Result	Default rule policy				
		Disabled	Major	Minor	Auto	Major+Minor
bpa	Train	79.3±1.7	82.0±1.4	80.7±1.4	81.0±1.6	82.0±1.4
	Test	64.0±7.5	62.6±7.5	64.4±6.9	64.5±7.3	62.6±7.5
	#rules	6.8±1.0	8.9±1.4	8.3±1.6	8.7±1.4	8.9±1.4
bps	Train	84.9±0.9	86.2±0.7	87.1±0.6	86.9±0.8	87.1±0.6
	Test	80.4±4.5	80.9±3.8	81.6±3.8	81.2±3.9	81.6±3.8
	#rules	5.1±0.4	6.1±1.1	5.9±1.0	5.8±1.0	5.9±1.0
bre	Train	97.7±0.4	98.3±0.3	98.5±0.4	98.4±0.4	98.5±0.4
	Test	95.7±2.3	95.0±2.6	95.7±1.9	95.8±1.9	95.7±1.9
	#rules	2.6±0.8	5.8±1.1	3.3±0.7	3.2±0.7	3.3±0.7
gls	Train	80.8±2.5	83.8±1.6	81.3±2.1	79.5±1.7	83.8±1.6
	Test	66.8±7.0	69.1±7.7	68.0±8.3	67.1±7.4	69.1±7.7
	#rules	6.5±0.7	6.8±0.8	7.5±0.9	6.7±0.8	6.8±0.8
h-s	Train	90.1±1.0	92.0±0.9	92.4±0.8	92.2±0.8	92.4±0.8
	Test	79.4±7.0	79.2±5.8	81.6±6.9	81.2±6.6	81.6±6.9
	#rules	6.6±0.8	7.8±1.3	7.4±1.2	7.4±1.2	7.4±1.2
ion	Train	96.1±0.6	95.9±0.8	97.1±0.7	96.9±0.7	97.1±0.7
	Test	93.5±3.5	90.4±4.3	93.4±3.5	92.8±4.0	93.4±3.5
	#rules	2.3±0.7	5.7±1.2	2.6±0.7	2.6±0.9	2.6±0.7
lrn	Train	75.7±1.7	77.2±0.8	75.8±1.4	75.7±1.0	77.2±0.8
	Test	68.0±5.0	69.1±5.4	68.7±5.2	69.1±4.9	69.1±5.4
	#rules	8.4±1.9	9.5±1.6	9.3±1.9	8.8±1.8	9.5±1.6
mmg	Train	80.3±1.7	83.4±1.3	83.4±1.3	83.5±1.1	83.4±1.3
	Test	65.9±8.3	69.0±8.0	67.3±8.9	69.7±7.7	69.0±8.0
	#rules	6.5±0.8	6.5±0.9	6.8±1.0	6.6±0.9	6.5±0.9
pim	Train	80.0±1.0	81.5±0.7	81.2±0.7	81.4±0.7	81.5±0.7
	Test	74.7±4.6	75.2±4.4	74.8±4.7	74.9±4.6	75.2±4.4
	#rules	5.3±0.6	6.3±1.1	5.8±0.9	6.1±1.0	6.3±1.1
son	Train	92.7±1.5	96.7±1.1	95.3±1.3	96.1±1.3	96.7±1.1
	Test	71.3±9.4	76.2±9.1	74.6±10.1	76.3±8.9	76.2±9.1
	#rules	6.7±1.0	7.6±1.3	7.7±1.5	7.6±1.4	7.6±1.3
thy	Train	97.6±0.9	98.6±0.7	98.6±0.7	98.3±0.8	98.6±0.7
	Test	91.5±6.2	92.0±5.2	92.4±4.8	91.4±5.6	92.4±4.8
	#rules	5.2±0.5	5.7±0.7	5.4±0.6	5.5±0.6	5.4±0.6
veh	Train	71.9±1.9	74.1±1.3	74.2±1.2	72.6±1.3	74.2±1.2
	Test	66.9±4.3	67.6±4.2	68.3±4.5	67.9±4.8	68.3±4.5
	#rules	6.5±1.3	9.4±1.8	10.0±1.8	8.4±1.8	10.0±1.8
wdbc	Train	97.2±0.8	98.0±0.5	97.9±0.6	97.8±0.6	98.0±0.5
	Test	93.9±2.9	94.4±3.1	94.4±3.2	94.4±3.1	94.4±3.1
	#rules	4.3±1.2	4.8±1.1	4.2±0.7	4.5±0.9	4.8±1.1
wine	Train	99.4±0.6	99.7±0.4	99.8±0.3	99.6±0.4	99.8±0.3
	Test	94.1±6.0	93.2±6.4	92.0±6.5	93.2±6.3	92.0±6.5
	#rules	3.8±0.7	3.7±0.6	4.2±0.5	3.8±0.7	4.2±0.5
wpbc	Train	84.9±2.8	89.9±1.8	87.1±3.3	89.0±2.1	89.9±1.8
	Test	76.6±6.7	75.3±7.0	72.4±9.1	76.3±7.1	75.3±7.0
	#rules	2.8±0.9	3.9±0.9	4.4±1.2	3.7±1.0	3.9±0.9
ave	Train	87.2±8.8	89.2±8.3	88.7±8.6	88.6±8.9	89.3±8.3
	Test	78.8±11.5	79.3±10.7	79.3±11.1	79.7±10.8	79.7±11.7
	#rules	5.3±1.7	6.6±1.7	6.2±2.1	6.0±2.0	6.2±2.2

Table 7: Summary of the statistical t-tests applied to the experimentation results of popsize 400, with a confidence level of 0.05. Cells in table count how many times the method in the row significantly outperforms the method in the column.

Policy	Disabled	Major	Minor	Auto	Major+Minor	Total
Disabled	-	2	1	0	0	3
Major	1	-	1	0	0	2
Minor	1	3	-	0	0	4
Auto	1	3	1	-	0	5
Major+Minor	2	3	1	0	-	6
Total	5	11	4	0	0	

automatic policy. As an alternative, although it has more computational cost, it is also interesting to refine the major+minor policy to determine why it cannot choose properly in certain datasets, and try to fix the problem.

Acknowledgments

The authors acknowledge the support provided by the Spanish Research Agency (CICYT) under grant numbers TIC2002-04160-C02-02 and TIC 2002-04036-C05-03, the support provided by the Department of Universities, Research and Information Society (DURSI) of the Autonomous Government of Catalonia under grants 2002SGR 00155 and 2001FI 00514. Additional funding from the German research foundation (DFG) under grant DFG HO1301/4-3 is acknowledged. Additional support from the Computational Science and Engineering graduate option program (CSE) at the University of Illinois at Urbana-Champaign is acknowledged.

Also, this work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-03-1-0129, and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research under grant N00014-01-1-0175. The US 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 Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

References

Bacardit, J., & Garrell, J. (2004). Analysis and improvements of the adaptive discretization intervals knowledge representation. In *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference* Springer (to appear).

Bacardit, J., & Garrell, J. M. (2003). Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system. In *Proceedings of the 6th International Workshop on Learning Classifier Systems* (in press), LNAI, Springer.

Blake, C., Keogh, E., & Merz, C. (1998). UCI repository of machine learning databases. (www.ics.uci.edu/mlearn/MLRepository.html).

Cohen, W. W. (1995). Fast effective rule induction. In *International Conference on Machine Learning* pp. 115–123.

DeJong, K. A., Spears, W. M., & Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, *13*(2/3), 161–188.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company, Inc.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Janikow, C. (1991). *Indictive learning of decision rules in attribute-based examples: a knowledge-intensive genetic algorithm approach*. Phd dissertation, University of North Carolina.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI* pp. 1137–1145.

Oei, C. K., Goldberg, D. E., & Chang, S.-J. (1991). *Tournament selection, niching, and the preservation of diversity* (IlliGAL Report No. 91011). Urbana, IL: University of Illinois at Urbana-Champaign.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *vol. 14*, 465–471.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*(3), 229–246.

Soule, T., & Foster, J. A. (1998, Winter). Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, *6*(4), 293–309.