MultiObjective Learning in a Genetic Classifier System (MOLeCS)

Ester Bernadó i Mansilla esterb@salleURL.edu
Josep Maria Garrell i Guiu josepmg@salleURL.edu

Departament d'Informàtica
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull (URL)
Passeig Bonanova, 8. 08022-Barcelona
Tel. 932 902 433 Fax. 932 902 416

Abstract

MOLeCS is a new Classifier System which addresses its learning as a multiobjective optimization of two goals: classifier *accuracy* and *generality*. These objectives are both emphasized in the fitness evaluation stage, driving the GA search towards the formation of accurate and general rules. We consider several existing multiobjective optimization strategies which establish a compromise between generality and accuracy in different ways. The system introduces two new proposals which balance both objectives with a bias towards accuracy, resulting in better classification performance.

The system also considers a third major objective: *covering*. It is achieved using some niching mechanisms that favour the maintenance of a set of cooperative rules.

Keywords: Genetic Algorithms, Machine Learning.

1 Introduction

The system described in this paper (MOLeCS) is a Genetic Algorithm (GA) which evolves sets of rules in order to perform classification tasks [2]. The system, which is related to previous Genetic Classifier Systems in some aspects [9, 10, 16], introduces a new way of evaluating the classifiers, promoting the formation of accurate and general classifiers.

Accuracy and generality are two major goals enforced in the rule-level. Accuracy is desirable for each rule, since the performance of the overall system depends on each rule accuracy. Generality is also desirable for several related reasons. First, we are looking for a minimum set of accurate rules. The more general the rules are, the smaller is the rule set. Besides, when the rule set is smaller the search space is reduced and thus it can be obtained more easily. And finally, as the rules are more general they cover more examples, promoting covering.

Covering is also a major objective to achieve: the rule set developed by the system should cover all the training examples. Covering is not guaranteed by the rule-level generality. This is due to the GA convergence, which drives the population toward a uniform distribution of the most highly fit individual (or rule). From the point of view of covering, this uniform convergence is not desirable. We want the GA to evolve a set of diverse and cooperative rules that together cover the problem. To prevent this convergence pressure, a restorative pressure is introduced with niching mechanisms. Niching balances competition and cooperation, promoting the formation of stable subpopulations or niches [10].

The paper analyses the achievement of the three stated goals: accuracy, generality and covering, through the experimentation of the new evaluation method proposed by MOLeCS, combined with different niching mechanisms.

The paper is structured as follows. First, a brief overview of Genetic Classifier Systems (CS) is given. The overview compares the related points between MOLeCS and previous CSs and emphasizes the motivations and aims of our new proposals. Next, MOLeCS is described in detail. Section 4 contains a description of the experimentation and

the obtained results. Finally, section 5 outlines our main conclusions and further work.

2 Brief Overview

The application of GAs to classification tasks has traditionally been addressed from two perspectives: the Pittsburgh approach and the Michigan approach, first exemplified by LS-1 [9] and CS-1 [14] respectively.

The Pittsburgh approach codifies each individual as a complete solution of the problem. Then, the solution returned by the system is the best individual to which the system has converged. Since each individual codifies a complete set of rules, its evaluation is directly based on the percentage of correctly classified examples, computed from the training set of examples [15]. The basic scheme of this approach presents a high computational cost. Besides, it usually has difficulties in the accuracy of learning because it performs a blind search inside each rule set [3]. The accuracy of a complete rule set is not a sufficient measure to guide the GA search to select the more accurate rules and discard the inaccurate ones. Some improved systems based on the Pittsburgh approach have overcome this problem in different ways [11, 8].

In the Michigan approach, each individual codifies one rule, and the solution must be a complete set of rules, that is, all the population. Two major questions arise from this approach: the evaluation of each rule (fitness evaluation) and the maintenance of a group of rules. The fitness evaluation method must provide a scalar measure that weighs the correctness of each rule and permits to establish a competition between rules. In traditional classifier systems, this fitness measure is based on the payoff prediction; that is, the payoff that the classifier would receive from the environment if its action is selected (e.g. Holland's CS [9]). Recently, XCS [16, 17] has migrated the fitness from the payoff prediction to the accuracy of the prediction, which results in better performance. Horn's study [10] also addresses the classifier's accuracy, which is defined as the percentage of correctly classified examples over the covered training examples.

The GA is used as the discovery component, exploring new promising points (rules) in the search space. In this sense, the GA uses the fitness value learned by the performance component, as the basis for exploration. The GA selects the "best" rules, performs crossover and mutation on them and introduces the resulting offspring into the population. Besides exploration, the GA must ensure covering, evolving a set of cooperative rules. Different niching mechanisms are applied in the research community to ensure this co-evolution: sharing payoff between active classifiers [10], performing restricted replacement [7], or translating the panmitic GA to the active classifiers (in the match set or action set) which can be classified as a kind of restricted mating [16].

Most Classifier Systems compute the classifier fitness incrementally, example by example. The GA application is sparse and its success can depend on the fitness accuracy; the number of representative examples seen by the system, its ordering, etc.

From the **accuracy** point of view, our approach is more related to Horn's study, since fitness is based on accuracy computed as the percentage of correctly classified examples. However, our system is taking account of the classifier **generality** too, which is a more complex task. Generality in MOLeCS is considered explicitally in the fitness evaluation stage, in constrast to XCS where generality is enforced in an implicit way [17]. Besides, fitness is computed non-incrementally. We have started with a non-incremental way because we want to evaluate the feasibility of this approach by itself, although the migration to an incremental system is not discarded in future work. **Covering** in MOLeCS is achieved with niching methods based on restricted replacement as described in the following section.

Our aims with MOLeCS are: first, to analyse the feasibility of this new approach and study its behaviour related to different evaluation methods and niching methods; and second, compare our results with previous classifier systems in solving classification tasks, specifically in solving real world classification tasks. In this paper, we are focused on the first goal, that is, the evaluation of the system and the understanding of its dynamics. The experimentation is given for some benchmark problems usually tested in the CS bibliography, which are the multiplexer and the parity problems.

3 Description of the system

MOLeCS is a Genetic Algorithm, where each individual is a rule or classifier. The rule is a condition represented by the ternary string $\{0, 1, \#\}^k$, where $\#$ is the *don't care* character, and an action repre-

sented by a binary string $\{0,1\}^l$:

$$rule : condition \rightarrow action$$

Each individual must have a fitness value, which permits the GA to compare the rules between them in order to maintain the useful ones and explore new promising rules. This is not an easy task, since each classifier does not represent a complete solution to the overall problem. Each rule matches a set of the examples (of variable sizes: from zero until the maximum of examples) and it can predict the action for each example correctly or not. Let us call these two characteristics *generality* and *accuracy* respectively, and compute them in the following way:

$$generality = \frac{\#\text{ covered examples}}{\#\text{ examples in the training set}} \quad (1)$$

$$accuracy = \frac{\#\text{ correctly classified examples}}{\#\text{ covered examples}} \quad (2)$$

Now we face the fitness evaluation method. Suppose we have two classifiers. If they have the same generality then we should prefer the most accurate. But having different generality, should we prefer the most accurate, though it has less generality? Or, on the contrary, should we choose the most general? Preferring always the most accurate classifiers may drive the search towards accurate but too specific classifiers, resulting in an enhancement of the solution set, poor covering, etc. And, on the other hand, choosing the most general classifiers could result in poor performance (in terms of classification accuracy). Therefore, we have to balance these two characteristics (generality and accuracy) in such a way that we obtain: a complete set of rules covering accurately all examples. This is promoted considering the classifier generality and the classifier accuracy as two goals to maximize. We have now a multiobjective optimization task, which is addressed with different algorithms, some of them inspired on previous works [4, 6]. Details are given in section 3.1.

Once the fitness assignment phase is performed, the Genetic Algorithm proceeds to the selection and recombination stages. These methods are implemented under a steady-state scheme, where only a small fraction G of the population is created. Selection is performed with Stochastic Universal Sampling (SUS) [1] and crossover and mutation are applied with probabilities p_c and p_m respectively.

As stated before, a major goal of the system is covering. Promoting general classifiers is not sufficient to reach it. The Genetic Algorithm can tend, by means of the selective pressure, to one general and accurate classifier, and usually one classifier does not solve the overall problem. Therefore, we must enforce the *co-evolution* of a parallel set of fit rules with niching mechanisms. Niching in MOLeCS is performed in the replacement stage. Niching with restricted replacement tries to replace each individual by an individual of the same niche, preventing thus than a highly fit classifier would be spread over all the population. Different niching algorithms are used under this proposal, which are described in section 3.2.

3.1 Multiobjective evaluation

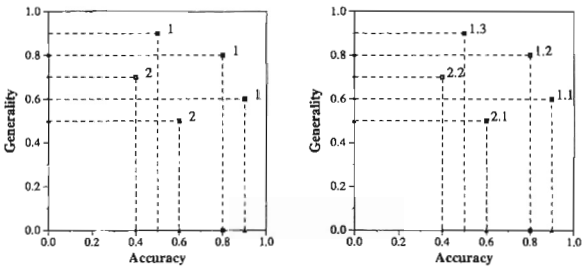
Our multiobjective learning consists of maximizing the *generality-accuracy* objective vector: $\bar{X} = \{g, a\}$. There are several methods to address a multiobjective optimization from the GA algorithm perspective [6, 4]. We are focused on the following approaches: Pareto-based, population-based non-Pareto and plain aggregating algorithms.

Pareto-based approaches. Pareto-based approaches rank the population in non-dominated sets and then, assign fitness according to this rank.

The ranking procedure is based on the concept of Pareto optimally, where a vector $\bar{x}^*$ is said to be optimal if there exists no other vector $\bar{x}$ with an improvement on one objective component without causing a decreasing in at least one of the other components. Such an optimal vector is called non-dominated vector.

The ranking method, first proposed by Goldberg [7] consists of assigning rank 1 to the first set of non-dominated solutions, then removing them from further contention assigns the next rank to the following non-dominated set, and so forth (see figure 1(a)). Fitness is assigned by interpolating a decreasing function (we applied a linear function) from the best individuals (rank 1) to the worst ones, and then averaging the fitnesses of those individuals of equal rank.

This algorithm explores the Pareto optimal front, that is the set of non-dominated solutions. There is usually needed a Decision Maker (DM) that selects the appropriate solution from all the available solutions in the Pareto-optimal front. In our system, the DM is automatically performed by the exploit or test phase. For each input example, a matching classifier must be selected in order to give a suitable



(a) PR (b) PAR

Figure 1: Multiobjective evaluation methods: Pareto Ranking (PR) and Pareto and Accuracy Ranking (PAR)

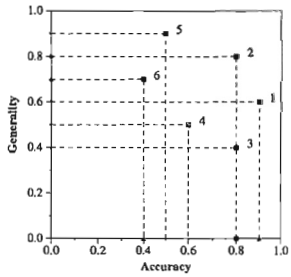


Figure 2: Accuracy and Generality Ranking (AGR)

classification. Our criterion is to choose the classifier with the highest fitness; that is, the classifier which is in the Pareto front, if possible. In case of two of more matching classifiers inside the Pareto front, the system automatically chooses the most accurate. Therefore, we are preferring the Pareto area closer to the 100% of accuracy.

Since the DM prefers the most accurate classifiers, we also studied a slight change on the rank map defined in 1(a), with the aim of biasing the search to the most accurate area. As shown in figure 1(b), inside each group of non-dominated classifiers a second level of ranking is performed, based on the accuracy of classifiers. Both approaches, which we called Pareto Ranking (PR) and Pareto-Accuracy Ranking (PAR) respectively, are compared in the results section.

Population-based non-Pareto approaches. Following the idea of promoting the most accurate areas, we have designed a population-ranking method which ranks the population according to the accuracy objective. When two or more individuals of the same accuracy are found, they are ordered in the generality objective. In this way, we clearly state that the first goal to be achieved is accuracy (in order to obtain accurate classifiers) and second, these classifiers must be "as general as possible". An example of such ranking, called Accuracy and Generality Ranking (AGR), is depicted in figure 2.

Plain Aggregating Approaches. The Weighted Sum (WS) approach weighs and sums up all the objectives obtaining directly a scalar fitness value. Our multiobjective problem is then solved as:

$$max \sum_{i=1}^{k} \bar{\omega}_i \bar{X}_i \quad (3)$$

where ω_i are the weighting coefficients that must be set depending on the relative importance of the objectives.

It is a fast method compared to the Pareto-approaches, but it needs the appropriate settings of the weighting coefficients, which are usually unknown or depending on the problem.

We have tuned our coefficients to the values $\bar{w} = (0.25, 0.75)$ corresponding to the objective vector $\bar{X} = (g, a)$. As described later, these coefficients are appropriately set for the multiplexer and parity problems used in the experimentation.

3.2 Niching

Niching methods are the key point for Classifiers Systems to evolve a parallel set of rules. The general niching methods are basically divided in two families: *crowding* and *sharing*. The former was introduced by De Jong in [5] with the algorithm "Crowding Factor model". This algorithm tries to preserve the diversity of population, by replacing each new individual (from the offspring subpopulation) by a similar one in the parents population.

Sharing is based on the concept of sharing fitness between the individuals in the same niche. The explicit fitness sharing [7] tries to allocate a number of individuals per niche proportional to the niche fitness. In Classifier Systems, sharing is usually implemented dividing the payoff by the number of active classifiers.

In this paper, we are focused on the first type of niching. We have analysed the most common types of crowding: from De Jong's Crowding Factor (CF),

through some variants including convergence pressure (as CIF and CC) until Deterministic Crowding (DC) which is the most promising method. They are described in the following sections.

CF. Given P1 a population of size *PopSize* and P2 the population with $G * PopSize$ offspring, the algorithm has to introduce each child of P2 into P1, deleting individuals from P1 to make room. For each child of P2, the algorithm proceeds as follows:

1. A subpopulation of CF members is selected randomly from P1.
2. The similarity between the child and all the individuals in CF is computed.
3. The child replaces the individual whose similarity is greater.

Different settings of the CF parameter give different behaviours. If CF is too high, there is a lack of convergence pressure [12], while for CF low it is difficult to maintain more than two or three niches. We have tested it as the lower bound we should obtain compared to other niching methods.

Closest of the Worst Crowding 1: CIF. In order to induce a convergence pressure in the CF model, two variants are tested: CIF and CC. Both try to replace a "low fitness and similar individual". The former differs from the CF model in the selection of the subpopulation of CF members. Instead of selecting them randomly, the selection is performed with a probability inversely proportional to fitness. Then, the worst individuals have more chances to be replaced.

Closest of the Worst Crowding 2: CC. Another way of introducing convergence pressure towards the best individuals is with the model used in Simple Classifier System (SCS) [7], which we call CC. The method consists of selecting each member going to the CF-subpopulation from a bucket of CSP individuals. The worst individual of the CSP-bucket is inserted into the CF-subpopulation. Then, the algorithm proceeds as the CF-model.

Deterministic Crowding. Proofs with the CF-model, with CF=PopSize, demonstrate that a high percentage of time the child replaces its parent [12]. Deterministic Crowding takes profit of this and introducing a convergence pressure, defines its algorithm as follows:

- All population undergoes crossover each generation. That means that the offspring population has PopSize members.
- Each pair of offspring competes with its respective parents. There are two possible competition models: a) offspring 1 against parent 1 and offspring 2 against parent 2 and b) offspring 1 against parent 2 and offspring 2 against parent 1. The model having the greatest sum of similarity is used.
- Once the competition model is chosen, a winner is selected for each pair of competing individuals. This winner is the individual with the highest fitness value. Therefore, a child only replaces its parent when its fitness is greater.

When applying Deterministic Crowding with PR, PAR and AGR evaluation methods, a consideration must be done. DC needs the fitness computation of children in order to be compared to the parents fitness. In the mentioned evaluation methods, the fitness of each individual depends on the fitness of other individuals of the population, unlike the WS algorithm. Then, the fitness of the offspring depends on the offspring population, while the fitness of the parents depends on the parents population. The comparison between a child fitness and a parent fitness may be unfair. To avoid this problem, the competing individuals are compared with their objective vector, using the appropriate rank map. Therefore, a child replaces its parent when it dominates its parent in terms of the rank map criterion.

4 Experimental Results

4.1 Design of experiments

Two types of problems are chosen for testing our system. They are the multiplexer problem (6-mux and 11-mux, with 6 and 11 inputs respectively) and the parity problem (5-par, with 5 inputs), both often tested by the CS community [16, 13]. The 6-multiplexer can be solved with 8 general rules (shown in table 1), while 11-mux is solved with 16 general rules. On the contrary, the 5-parity problem needs 32 specific rules. These two types of problems are useful to test the ability of the system to adapt the generality of rules according to the problem.

The goals of the experiments are: to test the feasibility of this new approach of CS, obtain the best

000###:0
001###:1
01#0##:0
01#1##:1
10##0#:0
10##1#:1
11###0:0
11###1:1

Table 1: Set of maximally general rules solving the 6-mux problem.

Fitness assignment	Replacement method
PR	Worst
PAR	CF
AGR	CIF
WS	CC
	DC

Table 2: Fitness assignment methods and Niching Replacement methods tested in the MOLeCS system.

way of evaluating the rules, demonstrate the need of niching methods and which niching method is more suitable. Table 2 shows the different fitness assignment and replacement methods, all of them described in the previous section.

The parameters required by the system are described in table 3.

4.2 Results

All the results shown for the WS algorithm are obtained with the weight vector $\bar{W} = (0.25, 0.75)$. These weights are adjusted from previous experiments (not reported here) but they can also be deduced for these known-problems as follows. The idea is to evaluate with the highest fitness the classifiers with a 100% of accuracy and the maximum generality. Next table shows the fitness of three classifiers for the 11-mux problem: an optimal classifier, with maximum accuracy and generality, an overgeneral classifier and a specific one.

Paramater	Description
PopSize	Population Size
Pgen	Probability of generalization (in the initialization of population)
G	generation Gap
p_c	probability of crossover
p_m	probability of mutation per gen
$\bar{w} = (w_g, w_a)$	weights of generality and accuracy in the WS algorithm
CF	Crowding Factor in CF-model and CIF
CSP/CF	size of CSP-subpopulation and CF-subpopulation in the CC algorithm

Table 3: Parameters of MOLeCS.

classifier	generality-accuracy	fitness
0000#######:0	(1/16, 1)	0.76
##########:0	(1, 0.5)	0.625
00000000000:0	(1/2048, 1)	0.75

This information is obviously not available in a real world problem. Then, the weights should be adjusted using a previous tuning process, or by a means of an expert. Otherwise, this evaluation method is not suitable and other types of evaluation must be chosen. Nevertheless, our study is interesting since we are performing a comparison between the WS algorithm with the proper weight settings and other methods that do not need parameters. The experiments will show us the upper bound that the WS algorithm can reach.

The replacement methods based on the Crowding Factor (CF, CIF and CC) are tested with different subpopulation sizes. In the 6-mux and 5-par problems, CF-model and CIF are tested with $CF = \{3, 10, 50\}$, and CC model with $CSP/CF = \{10/3, 10/10, 30/10\}$, using a population size of 250. In the 11-mux problem, which works with a population size of 800, $CF = \{10, 30, 160\}$, and $CSP/CF = \{30/10, 30/30, 160/30\}$. These values maintain the same proportion (relative to PopSize) as the 6-mux problem.

4.2.1 Multiplexer

We show the results obtained in the 11-multiplexer problem, which has more complexity than the 6-multiplexer problem. The results show a summary of the more representative results for each replacement algorithm, and some of them are detailed for the 4 evaluation methods when a significant difference between them exists. Each figure shows the average of five runs, each one with a different seed number, along the GA iterations. The measures shown are *covering* and *global accuracy*. Covering is defined as the percentage of covered examples over the training set. Global accuracy is the percentage of correctly classified examples over the total number of examples. The parameters used for

the 11-multiplexer in the experiments are: PopSize=800, Pgen=0.7, G=0.2, p_c=0.9, p_m=0.005, except in DC, where G changes to 1 and $p_c = 1$ according to the method definition.

Worst. This experiment shows the need of niching methods. Here, replacement is performed over the worst individuals of the population. That is, there is only selective pressure, without any restorative pressure enforcing the diversity of population. The population converges to a small part of classifiers (sometimes only one classifier, whose fitness was high in the earlier generations) and consequently covering is not achieved.

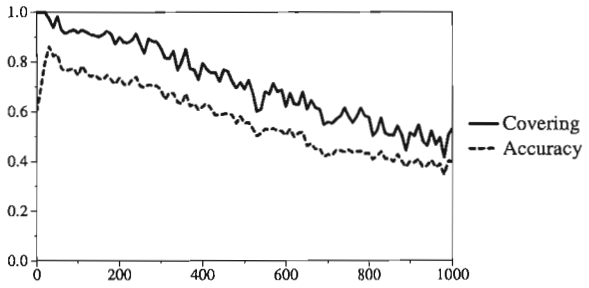


Figure 3: Results obtained in the 11-multiplexer problem, with replacement by the worst individuals of the population and WS algorithm. Covering and accuracy are shown along the GA iterations.

CF model. The results obtained with CF vary slightly respect to the CF value. Experiments are run for the values 10, 30 and 160. Figure 4(a) shows the best results, which are obtained with CF=160.

The introduction of this niching method has improved significantly the results obtained with the Worst Replacement method. This slight measure to promote diversity is useful for ensuring covering, but its is not enough to develop a good set of accurate rules. Accuracy moves from 70% to 80%.

Comparing the different evaluation methods, the best ones are the WS and AGR algorithms. Both outperform the Pareto based algorithms, because the rule accuracy is more enforced.

CIF. Adding a bias for replacing low fitness individuals improves the results obtained with the CF-model. Results are shown in figure 4(b) for the best CF parameter, which was of 160 too. Again the best fitness assignment methods are the WS and AGR algorithms, with a 100% of covering and nearly 90% of accuracy.

CC. This method provides a convergence pressure in a different way than the previous algorithm. There is an improvement of the results on the Pareto-based algorithms and also in the AGR algorithm, as shown in figure 4(c). Nevertheless, the results, which are shown for the parameter values $CSP/CF = \{30/30\}$, are very sensitive to the setting of these parameters. When the size of CSP-subpopulation raises up, e.g. to 160, the results (not reported here) begin to show a behaviour similar to the replacement by the worst individuals. In this sense, the CIF method is more stable in a wide range of CF parameter.

Deterministic Crowding. The results achieved with Deterministic Crowding (figure 5) outperform all previous results. Covering is 100% and accuracy reaches the 100% in the early generations (except when combined with PR). Therefore, it is the most efficient method and the most fast. The method balances appropriately the selective pressure and the maintenance of niches, reaching the optimal performance.

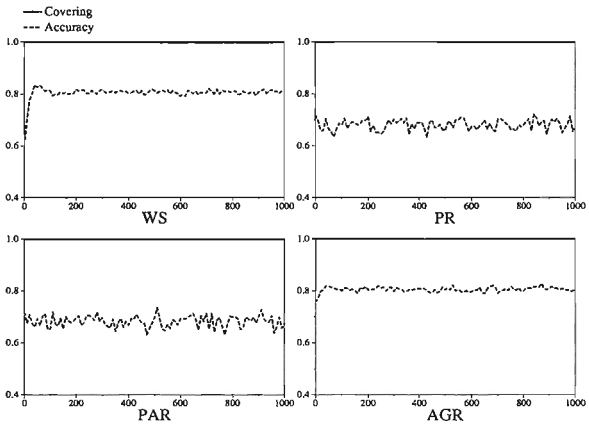
Our results with DC are consistent with other niching studies which demonstrate the superiority of this method on different test problems [13].

4.2.2 Parity problem

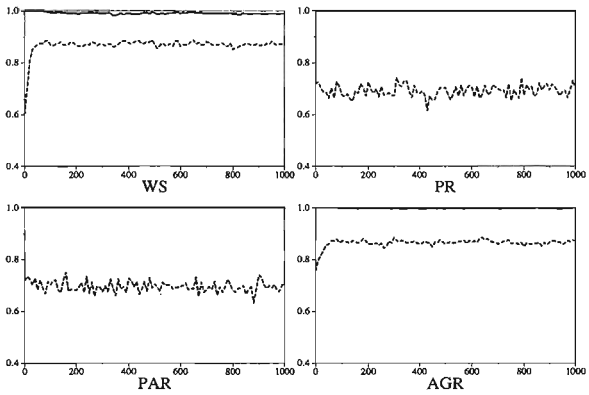
The parity problem with 5 inputs has a search space size similar to the 6-multiplexer problem. Nevertheless, its complexity is even greater than the 11-multiplexer, because it is necessary to find and maintain 32 different rules. Therefore, it is a difficult task for the niching methods. On the other hand, all the evaluation methods are designed for developing general and accurate rules. However, this problem does not need any general rule: the 32 solution rules do not contain any *don't care* symbol. We will study if our evaluation mechanisms are able to guide the GA search correctly.

The parameters used for the 5-par problem are: PopSize=250, Pgen=0.3, G=0.2, p_c=0.9, p_m=0.005, except in DC, where $G = 1$ and $p_c = 1$.

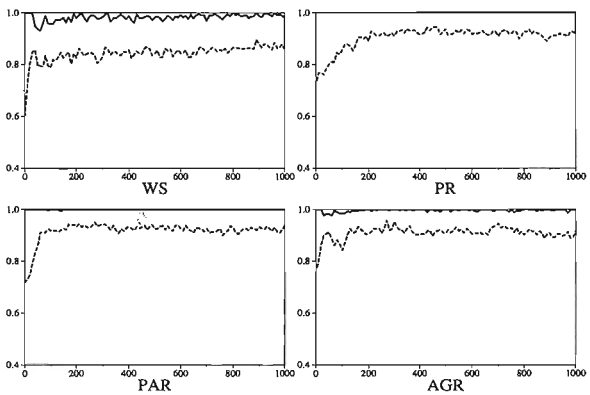
The results with the CF-model are shown for CF=50 in figure 6(a). They exhibit a clear lack of pressure towards accurate classifiers, resulting in a poor global accuracy.



(a) CF, with CF=160



(b) CIF, with CF=160



(c) CC, with CSP/CF=30/30

Figure 4: Results in the 11-multiplexer problem. Comparison between CF, CIF and CC. Each niching method is shown for each fitness evaluation method; that is, WS, PR, PAR and AGR algorithms. Curves are traced along 1000 iterations.

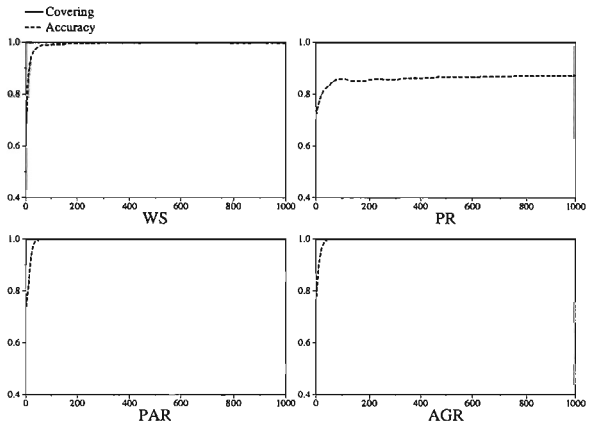


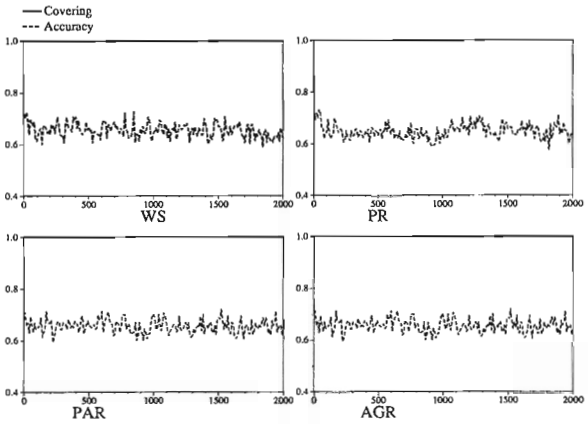
Figure 5: Results obtained in the 11-multiplexer problem, using replacement with DC.

Crowding models like CIF and CC improve these results -see figures 6(b) and 6(c)- specially the CC model, as happened with the 11-multiplexer problem. From the evaluation method point of view, the worst results are achieved with the Pareto Ranking method, because this evaluation does not show any pressure towards accurate classifiers. Given two classifiers C1=(g1,a1) and C2=(g2,a2), with $g1 > g2$ and $a1 < a2$, PR evaluates them equally, while C2 should be clearly fitter in this environment. The search is not biased towards accurate classifiers, but towards a compromise between general and accurate classifiers. Nevertheless, the exploit phase, which acts as the Decision Maker, would choose C2 as the classifier sending the action. This fact improves the results that we would have obtained if such a preference was not established in the exploit process.

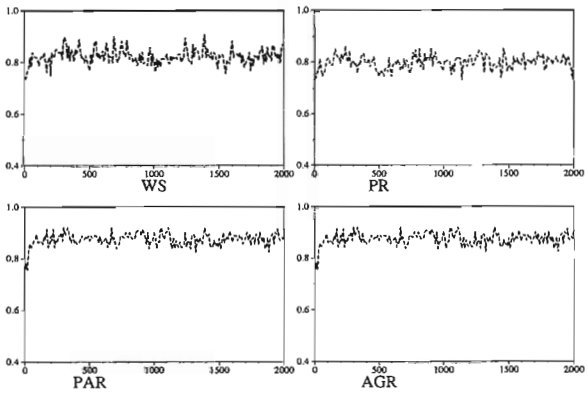
Finally, Deterministic Crowding (see figure 7) also shows a good performance, although it takes more time than with the 11-mux problem to reach the optimal accuracy. Besides, DC shows a more stable performance curve, while CC and CIF fluctuate a bit. This small fluctuation is due to the replacement errors that can cause the loss of an important classifier.

5 Conclusions

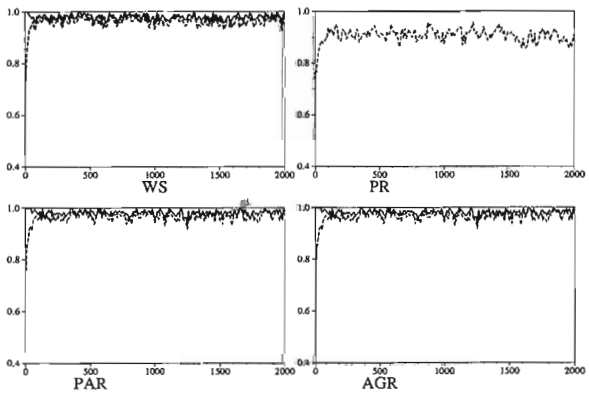
This paper has studied a new approach of a Genetic Classifier System. MOLeCS has introduced a new way of evaluating the individuals, on the multiobjective optimization of the classifier generality and accuracy.



(a) CF, with CF=50



(b) CIF, with CF=50



(c) CC, with CSP/CF=10/10

Figure 6: Results in the 5-parity problem with CF, CIF and CC niching methods. Each niching method is detailed for each evaluation algorithm: WS, PR, PAR and AGR. Curves are displayed along 2000 iterations.

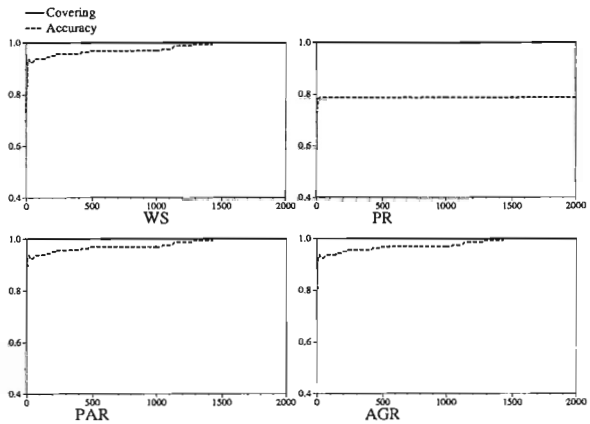


Figure 7: Results obtained in the 5-parity problem, with replacement with DC. The results are detailed for the evaluation algorithms: WS, PR, PAR and AGR.

The classical Pareto approach (PR) shows very little preference towards accurate classifiers and results in low performance. The Pareto and Accuracy Ranking (PAR) algorithm sometimes outperforms the PR algorithm, because it tries to induce a bias towards accurate classifiers inside a non-dominant group. Nevertheless, we believe that the rank map defined by the Pareto approaches is not suitable enough for developing accurate and general classifiers. It establishes a compromise between generality and accuracy, but we prefer accuracy (in order to reach the 100%) and later on, generality (in order to have a small set of accurate rules and also promote covering). For this reason, the rank map defined by the Accuracy and Generality Ranking (AGR) method results in the best performance. It biases the search towards the most accurate classifiers and in case of equal accuracy, the most general are preferred. Then, the GA search first looks for accurate classifiers and later on improves them on the generality objective. On the other hand, the Weighted Sum (WS) algorithm also shows good results although its applicability depends on the setting of an appropriate weighting vector.

A third major objective is also studied along the paper: *covering*, which is promoted with the niching mechanisms. Without niching, the selective pressure tends to bias the GA search to a small area of the search space, preventing thus the system to solve the overall classification problem. The best results are obtained with Deterministic Crowding, because it balances appropriately the convergence

pressure (in order to obtain well-fitted classifiers) and the maintenance of different niches (replacing children by their respective parents).

MOLeCS has been applied to the multiplexer and parity problems. Current work is applying MOLeCS to more difficult classification tasks, including real world problems which need a change on the rule representation. We are also analysing the feasibility of introducing new niching mechanisms based on the concept of sharing, as proposed by other types of Classifier Systems.

Acknowledgements

The results of this work were obtained with the equipment co-funded by *Direcció de Recerca de la Generalitat de Catalunya (D.O.G.C 30/12/1997)*. The first author acknowledges the support provided by Epson Iberica, under Rosina Ribalta Award, 1999. We also would like to thank *Enginyeria i Arquitectura La Salle* for their support to the AI Research Group.

References

[1] J.E. Baker. Reducing bias and inefficieny in the selection algorithm. In J.J.Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987.

[2] Ester Bernadó Mansilla and Josep Maria Garrell i Guiu. MOLeCS: A MultiObjective Classifier System. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, page 390, 2000.

[3] Ester Bernadó Mansilla, Abdelouahab Mekaouche, and Josep Maria Garrell i Guiu. A Study of a Genetic Classifier System Based on the Pittsburgh Approach on a Medical Domain. In *12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-99*, pages 175–184, 1999.

[4] Carlos A. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

[5] K.A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. (Doctoral Dissertation, University of Michigan) , 1975.

[6] Carlos M. Fonseca and Peter Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[7] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.

[8] John J. Grefenstette, Connie Loggia Ramsey, and Alan C. Schultz. Learning Sequential Decision Rules Using Simulation Models and Competition. *Machine Learning, 5(4)*, pages 355–381, 1990.

[9] John H. Holland. Escaping Brittleness: The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. *Machine Learning: An Artificial Intelligence Approach, Vol. II*, pages 593–623, 1986.

[10] J. Horn, D.E. Goldberg, and K. Deb. Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation, 2(1)*, pages 37–66, August, 1994.

[11] Francesc Xavier Llorà i Fàbrega and Josep Maria Garrell i Guiu. GENIFER: A Nearest Neighbour based Classifier System using GA. In Wolfgang Banzhaf et al, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, page 797. Morgan Kaufmann, 1999.

[12] Mahfoud, Samir W. Crowding and preselection revisited. In R.Maenner and B.Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 27–36. Elsevier:Amsterdam, 1992.

[13] Mahfoud, Samir W. *Niching Methods for Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[14] S. F. Smith. Flexible Learning of Problem Solving Heuristics through Adaptive Search. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 422–425, 1983.

[15] William M. Spears and Kenneth A. De Jong. Using Genetic Algorithms For Supervised Concept Learning. *Machine Learning, 13*, pages 161–188, 1993.

[16] Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[17] Stewart W. Wilson. Generalization in the XCS Classifier System. In J.Koza et al., editor, *Genetic Programming: Proceedings of the Third Annual Conference.* San Francisco, CA: Morgan Kaufmann, 1998.