MOLeCS: Using Multiobjective Evolutionary Algorithms for Learning

Ester Bernadó i Mansilla and Josep M. Garrell i Guiu

Computer Science Department
Enginyeria i Arquitectura La Salle, Ramon Llull University
Passeig Bonanova, 8, 08022-Barcelona (Spain)
{esterb,josepmg}@salleURL.edu

Abstract. MOLeCS is a classifier system (CS) which addresses its learning as a multiobjective task. Its aim is to develop an optimal set of rules, optimizing the *accuracy* and the *generality* of each rule simultaneously. This is achieved by considering these two goals in the rule fitness. The paper studies four multiobjective strategies that establish a compromise between accuracy and generality in different ways. The results suggest that including the decision maker's preferences in the search process improves the overall performance of the obtained rule set. The paper also studies a third major objective: *covering* (the maintenance of a set of different rules solving together the learning problem), through different niching mechanisms. After a performance analysis using some benchmark problems, MOLeCS is applied to a real-world categorization task: the diagnosis of breast cancer.

1 Introduction

The learning task performed by a classifier system (CS) is itself multiobjective [13]: it has to find a concept description, usually represented by a set of rules, which should be: (a) complete, (b) accurate and (c) minimum. In terms of classification, a rule set is *complete* when it covers (satisfies) all the examples, whereas a rule set is *accurate* when it classifies the examples correctly (i.e., without misclassification errors). The third objective involves *minimizing* the number of rules, in order to obtain concise and comprehensible descriptions. Another related objective is the system's capability to express *generalizations*, that is, to generalize all the similar examples.

These multiple objectives are closely connected. Generalization of equivalent examples allows more concise representations and leads up to smaller rule sets, promoting covering. But these objectives are opposed to accuracy in some way. If the system performs an excessive generalization, the accuracy of classification will be degraded. Thus, optimizing these conflicting objectives simultaneously involves a weak equilibrium, which is difficult for a CS to reach and maintain.

The classifier systems' community has solved this multiobjective learning task in an implicit way. Holland's CS [11] uses a credit allocation algorithm to

evaluate each rule efficiency and a mechanism to allow the formation of hierarchies, although there is not an explicit pressure towards the formation of general rules. XCS [21][22] is a classifier system which has shown a strong tendency to achieve the stated objectives. One of its main strengths is the definition of fitness, based on the prediction accuracy of each rule rather than on the prediction itself. Although it does not use an explicit bias to favour the generalization of rules, this is achieved by the application of the GA to environmental niches [22].

MOLeCS is a recent CS [2][3] designed to solve classification tasks by supervised learning. Its main contribution is the definition of the *accuracy* and *generality* measures for each rule, and the use of multiobjective solution techniques for optimizing them simultaneously. In this sense, the learning task is represented explicitly as a multiobjective problem: the optimization of the accuracy and the generality of each rule.

This paper studies, under the MOLeCS architecture, the performance of different multiobjective evolutionary algorithms in achieving the learning goals, which can be summarized as: obtaining the *minimum* rule set that *covers accurately* all the examples. Different niching strategies are also considered in order to maintain a parallel set of rules and thus, achieve covering. First, we place the MOLeCS system into the CSs and MOEAs (multiobjective evolutionary algorithms) frameworks. In section 3 we describe MOLeCS and discuss different multiobjective approaches and niching methods. Next, we compare them using some benchmark problems often tested in the research community. In section 5, we show the application of MOLeCS to a real classification task based on the prediction of breast cancer. Finally, we give our conclusions and future work.

2 Background

In this section, a brief overview of classifier systems is made, remarking on the main differences between previous CSs and MOLeCS. Next, we review the similarities and differences between a typical multiobjective evolutionary algorithm and a classifier system as MOLeCS.

2.1 Classifier Systems

Typically, learning classifier systems codify each individual as one rule, while the solution that must be obtained is a complete set of rules (that is, all the population). Two major issues arise from this approach: (a) the evaluation of each rule (fitness) and (b) the maintenance of a group of rules.

The fitness evaluation method must provide a scalar measure that weighs the efficiency of each rule. In traditional classifier systems, this fitness measure was based on the payoff prediction; that is, the payoff that the classifier would receive from the environment if its action was selected (e.g. Holland's CS [11]). Recently, XCS [21][22] has migrated the fitness from the payoff prediction to the accuracy of the prediction, which results in better performance. Horn's study

[13] also addresses the classifier's accuracy, which is defined as the percentage of correctly classified examples over the covered training examples.

Different niching mechanisms are proposed in the research community to ensure covering by the co-evolution of different rules: sharing payoff between active classifiers (see [13]), performing restricted replacement [9], or translating the panmictic GA to the active classifiers (match set or action set) which can be classified as a kind of restricted mating [21],[22].

The classifier system's ability to evolve generalizations is a major issue that has recently received a growing interest. In XCS, the application of the GA to the action sets has resulted in a pressure towards generalizations. The generalization hypothesis [21] states that given two classifiers (rules) C1 and C2 (both equally accurate, where C2 is a generalization of C1) the more general rule (C2) tends to displace the more specific one (C1). This is due to the fact that the more general rule participates in more action sets, having more reproductive opportunities and achieving thus more copies. Frey and Slate [8] proposed a CS where fitness was based on accuracy. A pressure towards generalization was induced using an "utility" measure. This was computed as the number of correctly classified examples over the total number of examples seen by the system. Rules with a low "utility" measure were deleted in order to favour those rules more used.

From the **accuracy** perspective, 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 explicitly in the fitness evaluation stage, in contrast to XCS where generality is enforced in an implicit way. Frey and Slate also proposed the use of two different measures similar to ours, although they did not try to optimize them with multiobjective techniques. In this sense, the application of multiobjective techniques is new and offers promising perspectives for learning systems. **Covering** in MOLeCS is induced in the replacement stage and it is based on restricted replacement methods. Although Michigan style classifier systems are incremental, we have started with a non-incremental proposal. Our main reason is to understand the task of the multiobjective evaluation and niching, under a "classic GA" scheme. The migration to an incremental version will be performed in a near future, with the aim of improving the computational cost.

2.2 MOEAs and CSs

A multiobjective evolutionary algorithm (MOEA) and a learning classifier system (CS) have many related points:

- *Maintenance of a group of different solutions.* The solution of an MOEA is usually a set of many points, approximating the set of non-inferior solutions (Pareto optimal set). Similarly, the solution returned by a CS is a set of many rules, solving together a concept description. In order to find and maintain such multiple optima, some niching mechanism is required. MOEAs usually implement sharing, with the aim of obtaining a uniform distribution of the Pareto front or the Pareto optimal set.

– *Evaluation of an individual.* In MOEAs, each solution fitness depends on the other solutions, as it happens to each rule in the CSs.

Nevertheless, we can highlight some differences:

– *Generational and Non-Generational schemes.* One desirable feature in CSs is the on-line performance. This is promoted with a non-generational scheme, which introduces slight changes into the population by replacing only a small fraction of the population in each generation. This is not a very common scheme in MOEAs, although some proposals do exist [17]. Crowding is a natural way of performing niching in a non-generational scheme. For that reason, it has widely been used in classifier systems [11], [9].
– *Solution returned by the system.* Another important issue is the solution that MOEAs and CSs must obtain. MOEAs usually try to find a well-distributed population belonging to the Pareto front (or Pareto optimal set), from where the decision maker (DM) can perform a selection. If we translate this objective to MOLeCS, it should be expected to find the Pareto front corresponding to the *generality* and the *accuracy* goals. When we use the system in the exploitation mode, the DM has to select the best rules from all the available rules in the Pareto optimal set. Which rules should be used? If we want to perform accurate classifications, it seems obvious to choose always the most accurate rules. Therefore, we can state our learning problem as a particular multiobjective problem (MOP) where the DM's choices are known *a priori*. We can take profit of that by guiding the search towards the preferred areas of the DM.

3 Description of the system

Each individual in MOLeCS codifies a rule (classifier) of type: $rule : condition \rightarrow action$. The condition is the conjunction of tests over the problem attributes. It is represented by the ternary string: $\{0, 1, \#\}^k$, with length equal to the number of describing attributes. The symbol '#' (don't care) matches all values of an attribute, so it permits us to express generalizations. The action part of the rule is represented by the binary string: $\{0, 1\}^l$.

Each individual must have a fitness value in order to apply the appropriate selective pressure. This is not an easy task, since each classifier does not represent a complete solution to the overall problem. In fact, each rule r_i can match a different number of examples and can predict those examples in different degrees of accuracy. We compute these two features for each rule in the population:

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

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

If fitness is only based on *accuracy* the search will be biased towards accurate but too specific rules. This can result in an enhancement of the solution set,

poor covering, etc. On the contrary, basing fitness on *generality* will result in low performance (in terms of classification accuracy). The solution is to balance these two characteristics (accuracy and generality) and optimize them simultaneously. Our hypothesis is that a multiobjective approach will lead the search towards general and accurate rules, resulting in a minimum, complete and accurate set of rules. We have tested and compared different multiobjective strategies, which are described in section 3.1.

Once the fitness assignment phase is performed, the GA proceeds to the selection and recombination stages. Selection is performed with stochastic universal sampling (SUS) [1].

As stated before, a concept description must be complete; that is, all the input examples must be covered. We use the term *covering* as the ratio of instances covered by the entire rule set RS to the size of the training set:

$$covering = \frac{\#\text{ examples covered by RS}}{\#\text{ examples in the training set}} \tag{3}$$

Promoting general classifiers is not sufficient to reach a 100% of covering. The genetic algorithm can tend, due to the genetic drift [10], 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 set of fit rules by niching mechanisms. Niching in MOLeCS is performed in the replacement stage (section 3.2).

Once the system has learned, it is used under an exploit or test phase. It works as follows. An example coming from the test set is presented. Then, the system finds the matching rules and applies the fittest rule to predict the associated action or class. As explained before, in case of equally fit rules, the most accurate rule is chosen.

3.1 Multiobjective Learning

In the previous section, we have defined our learning as a multiobjective problem (MOP). Now, we formalize the concepts mentioned before. Next, we consider different multiobjective algorithms to solve our MOP.

Definition 1 *The MOP evaluation function, $F : X \rightarrow Y$ maps decision variables $\mathbf{x} = (x_1, ..., x_n) \epsilon X$ to vectors $\mathbf{y} = (y_1 ... y_k) \epsilon Y$. In MOLeCS, the decision variables are the rules, while the objective vectors, of dimensionality k=2, are of type: $\mathbf{y} = F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$, where $f_1(\mathbf{x}) = accuracy(\mathbf{x})$ computed from equation (2) and $f_2(\mathbf{x}) = generality(\mathbf{x})$, from equation (1).*

Definition 2 *Our multiobjective learning problem is defined as follows:*

Maximize $\mathbf{y} = F(\mathbf{x}) = (accuracy(\mathbf{x}), generality(\mathbf{x}))$

where $\mathbf{x}$ *is the decision vector, and* $accuracy(\mathbf{x})$ *and* $generality(\mathbf{x})$ *are described by equations (2) and (1) respectively.*

There are several MOEA techniques [7], [5], [18]. We have tested and compared four different algorithms, representative of three main algorithmic approaches [1]: a Pareto-approach, a non-Pareto approach and a plain aggregating approach.

Pareto-based Approach. We consider the Pareto approach proposed by Goldberg [9], which consists of ranking the population into non-dominated sets and then, assigns fitness according to this rank.

The Pareto approach gives the same selective pressure to non-dominated objective vectors. Suppose we have two objective vectors: $\mathbf{y}_1 = (1, 0.6)$ and $\mathbf{y}_2 = (0.5, 0.9)$, being non-dominated as depicted in figure 1(a). In this case, the Pareto approach assigns them the same rank. This means that the final solution set returned by the system can contain overgeneral rules (e.g., vector $\mathbf{y}_2$) as well as maximally general rules[2] (e.g., vector $\mathbf{y}_1$). Nevertheless, in the exploitation phase, these overgeneral rules will not be selected, because they can degrade the classification performance. The decision maker will always choose the best accurate rules (i.e. the maximally general rules).

Our hypothesis is that these overgeneral rules do not contribute significantly to our final solution. Indeed, they can degrade our search towards an accurate rule set, because they consume resources from the population and thus, they may prevent other accurate rules from being explored.

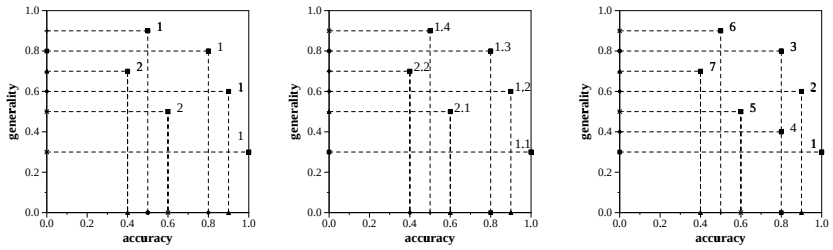
If we know the decision preferences in advance, we may guide the GA more efficiently towards these preferences. For that reason, we have designed a modification of the Pareto approach which gives a bias towards accurate rules. As shown in figure 1(b), inside each group of non-dominated classifiers a second level of ranking is performed, based on the accuracy of classifiers. In the following, we will refer to the Pareto original approach as PR (Pareto ranking) and to the modified algorithm with the accuracy bias as PAR (Pareto-accuracy ranking).

Population-based non-Pareto Approach. Using the idea of promoting the most accurate areas, we have designed a population-ranking method, based on the lexicographic ordering. The algorithm ranks the population according to the accuracy objective. When two or more individuals equally accurate are found, they are ordered by the generality objective. In this way, we 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 this ranking, which we term *accuracy-generality ranking* (AGR), is depicted in figure 1(c).

---

[1] We consider here the MOEA classification scheme made by Fonseca and Fleming in [7].

[2] Kovacs [14] defines a maximally general rule as an accurate rule (accuracy=1.0) which cannot be more general without becoming inaccurate. A rule being inaccurate due to excessive generalizations, is called an overgeneral rule. A suboptimally general rule is an accurate rule that can be more general (have more '#') without losing its accuracy.



(a) PR (b) PAR (c) AGR

Fig. 1. Multiobjective evaluation methods: Pareto ranking (PR), Pareto-accuracy ranking (PAR) and accuracy-generality ranking (AGR).

Plain Aggregating Approach. 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: $max \sum_{i=1}^{k} w_i y_i$ where $\mathbf{w}$ is the weight vector that must be set depending on the relative importance of the objectives. When we test MOLeCS with well-known problems (see section 4), we can tune these coefficients properly. In this case, we use the WS as a bound from where the other algorithms can be compared.

A Priori and A Posteriori Preference Articulation. From the DM's perspective, PAR, AGR and WS can be classified as *a priori preference articulation* methods [19], because they use the decision preferences in the fitness/selection stage. Our preferences are: "those accurate rules being as general as possible" or in other words, the *maximally general rules*.

On the contrary, the Pareto based approach (PR) does not use these preferences until the exploitation phase, when the learning process has finished. Therefore, it searches for a compromise between the accuracy and the generality goals. From the resulting set of non-dominated solutions, the DM has to select the maximally general rules *a posteriori*.

In sections 4 and 5, we will test if the methods based on *a priori preference articulation* can outperform the solution given by *a posteriori preference articulation* method as the Pareto approach.

3.2 Niching

Niching methods are the key point for classifier systems to evolve a population of diverse rules. We enforce niching in the replacement stage using crowding strategies. We have analysed crowding, two variants of crowding including selective pressure and deterministic crowding.

Crowding (or crowding factor model, CF) was introduced by De Jong [6]. The algorithm tries to preserve the diversity of population, by replacing each new individual (from the offspring population) by a similar one in the parents

population. To be exact, the new individual is compared to a subpopulation of *cf* members and the member with the highest similarity is replaced.

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. The second variant, which we term CC, was used in the Simple Classifier System (SCS) [9]. The method consists of selecting each member going to the cf-subpopulation from a bucket of *csp* individuals. The worst individual from the csp-bucket is inserted into the cf-subpopulation. Then, the algorithm proceeds as the CF model.

Deterministic crowding (DC) makes a competition between each pair of {parent,offspring}, choosing the competing pairs with a minimum-distance criterion [15]. The child only replaces its parent when its fitness is greater.

4 Experimental Results

4.1 Design of Experiments

We first analyse the learning performance of the system with two well-known learning tasks, usually tested in the CSs' community [21], [16]: the multiplexer problem and the parity problem. This election is made for several reasons. First, for the simplicity to test our system, since the desired solution is known. Second, because they represent two different types of problems. The multiplexer allows generalizations in its rules; so the bias towards generality is desirable for the achievement of a minimal set. On the contrary, the parity problem does not need any generality pressure, since all the rules required to describe the problem must be specific. In this sense, we will test the ability of MOLeCS (and its multiobjective algorithms) to scale to the different levels of generalization.

The results are shown for the multiplexer with 11 inputs (11-mux) and the parity problem with 5 inputs (5-par). Each problem is tested with: four multiobjective strategies (PR, PAR, AGR and WS) and four different niching methods (CF, CIF, CC and DC). We also compare our results with a single-objective EA, optimizing only accuracy (i.e., using the WS approach with $\mathbf{w} = (1, 0)$).

Each result is the average of five runs using different seed numbers. The parameters settings for the 11-mux problem are: population size = 800, Pgen=0.3 (probability of generalization in the initialization of population and in the mutation operator), G=0.2 (generation gap), $p_c = 0.9$ (probability of crossover), p_m=0.01 (probability of mutation per gene). If DC is used, $p_c = 1.0$ according to the algorithm definition. The crowding methods CF, CC and CIF require the subpopulation sizes. They are tuned previously, and here we only show the results obtained with the best parameters (whose values are reported in the corresponding figures). In case of the 5-par problem, the parameters settings are the same, except for the population size that is 250.

4.2 Metrics of Performance

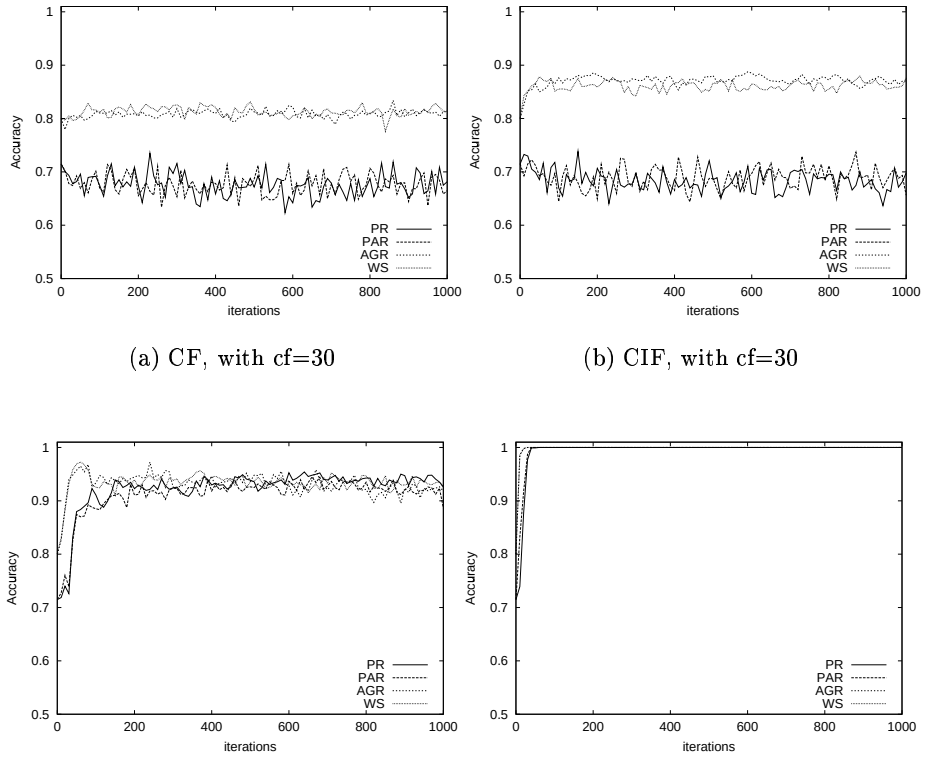
In order to test the learning performance of the system, we will use the following metrics:

- *Covering.* This is the ratio of training instances covered by the population, as defined in equation (3). This measure is related to the ability of the niching methods to maintain multiple rules. Covering can also be improved if generalizations are found.
- *Accuracy.* Two metrics of accuracy can be performed on the overall rule set. The first one, termed as *crude accuracy* (CA) [12], is defined as the number of correctly classified examples over the covered examples. The second measure, or *corrected crude accuracy* (CCA), is the ratio between the correct classified examples and the total number of examples presented to the system.
- *Size of the solution set.* One desirable goal in learning is to minimize the size of the solution set. If the set is small, it is more explanatory and easier to understand by the human experts. We will consider the size of the solution set as the number of different rules.
- *Optimal Population.* Kovacs [14] defines an optimal population (denoted as [O]) as having three characteristics: it is complete, non-overlapping and minimal. In case of the 6-multiplexer problem, the optimal population in MOLeCS consists of 8 rules, of type: 01#0##:0. In the 11-multiplexer problem, [O] consists of 16 rules, while the 5-parity problem needs 32 rules. This measure is useful for testing if the developed rules have reached the optimal generality, in comparison to accuracy and covering that do not necessary give us this information. In the multiplexer problem, reaching a 100% of accuracy and covering does not imply directly that [O] has been reached.
- *Learning speed.* Although the speed measure is more important in the test epoch (exploit phase), the speed in the training epoch is also desirable, specially when the system has to learn from real-world applications.

4.3 Results

Figure 2 shows a summary of the results obtained in the 11-multiplexer problem. A graph represents a fixed niching method, and inside each one there is a curve for each multiobjective strategy. The curves plot the CCA (corrected crude accuracy), which is measured over all the training examples. Covering is not shown since all methods achieved the 100%. The WS approach was previously tuned to $\mathbf{w} = (0.75, 0.25)$.

The main differences arise between the crowding methods. CF is the method with the worst accuracy, which ranges from 0.70 to 0.80 (see figure 2(a)). Adding a selective pressure in the replacement stage improves the performance. This happens specially with CC -see figure 2(c)-, where accuracy is about 0.90. These results are obtained for subpopulation sizes of 30/30. Nevertheless, this method is very sensitive to the parameter settings. When the size of the csp-subpopulation raises up, increasing the selective pressure, the accuracy (not reported here)



(a) CF, with cf=30

(b) CIF, with cf=30

(c) CC, with csp/cf=30/30

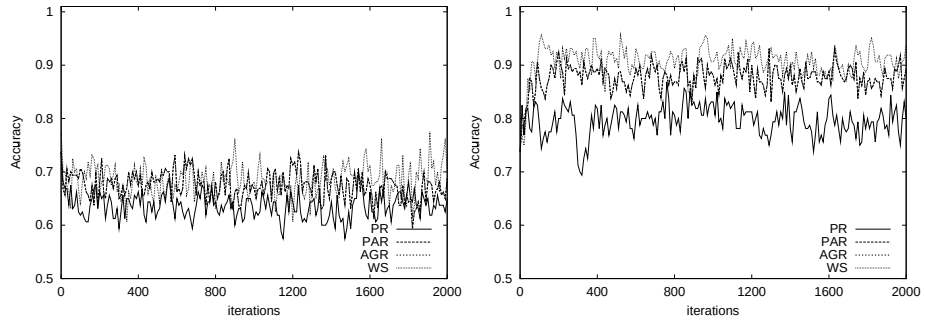
(d) DC

Fig. 2. Results in the 11-multiplexer problem. Comparison between four different crowding methods: CF, CIF, CC and DC. Each niching method is shown for each multiobjective evaluation method: PR, PAR, AGR and WS with $\mathbf{w} = (0.75, 0.25)$. Curves show the corrected accuracy average over five runs, traced along 1000 iterations.

decreases. The results achieved with deterministic crowding -figure 2(d)- outperform all previous results. Accuracy reaches 1.0 in the early generations. The method balances appropriately the selective pressure and the maintenance of niches, reaching the optimal performance. These results are consistent with other niching studies which demonstrate the superiority of DC on different test problems [16].

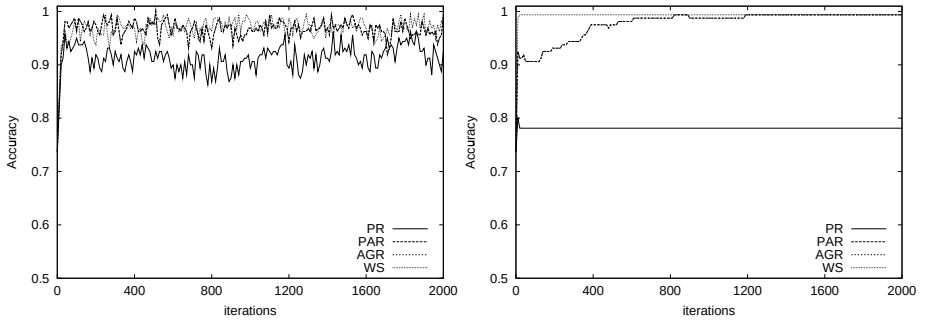
When the appropriate crowding method is used, there are no significant differences between the four multiobjective algorithms, in terms of accuracy and speed performance.

In the 5-par problem, the behaviour of the different niching methods is similar to the 11-mux problem (see figure 3). What is important to mention here is the difference in performance that arises between some multiobjective methods. Figure 3(d) shows that the Pareto approach has the poorest accuracy (with a value of 0.78). This is because PR does not establish any preference towards



(a) CF, with cf=10

(b) CIF, with cf=10



(c) CC, with csp/cf=10/10

(d) DC

Fig. 3. Results in the 5-parity problem. Comparison between four different crowding methods: CF, CIF, CC and DC. Each niching method is shown for each multiobjective evaluation method: PR, PAR, AGR and WS with $\mathbf{w} = (1, 0)$. Curves show the corrected accuracy average over five runs, traced along 2000 iterations.

the rule accuracy, but towards a compromise between generality and accuracy. Thus, PR seeks for overgeneral rules as well as for maximally general rules. These results confirm our hypothesis about the presence of overgeneral rules in the population. They do not contribute to the expected solution, preventing other desirable accurate rules from being explored. Adding a preference towards accurate rules, as PAR and AGR do, improves the previous results. WS has the best speed, although its application depends on the appropriate knowledge about the weight vector settings. In these results, $\mathbf{w} = (1, 0)$, which is the same as a single objective optimization towards accuracy.

Table 1 reports the different performance measures obtained from the final rule set (after learning is performed). In the 11-mux problem, all the multiobjective approaches have achieved the same performance, except for PR which presents the highest rule set size (217). The last row in the table shows the results obtained by a single-objective learning (SO), optimizing accuracy. Covering

Table 1. Results in the 11-mux and 5-par problems, summarized for the different multiobjective strategies and compared to a single optimization method (row labeled as SO). DC is used as the crowding method. In case of the 11-mux, $\mathbf{w} = (0.75, 0.25)$. In case of 5-par problem, $\mathbf{w} = (1, 0)$, which is the same as the single-objective algorithm. **Cov** is covering, **CCA** is the corrected accuracy, **Size** is the number of different rules and **%[O]** is the percentage of the optimal population reached by the final rule set. These measures are computed from the final rule set (obtained when the training epoch has finished).

	11-multiplexer				5-parity			
	Cov	CCA	Size	%[O]	Cov	CCA	Size	%[O]
PR	1	1	**217**	1	1	**0.78**	21	**0.60**
PAR	1	1	40	1	1	0.99	34	0.99
AGR	1	1	40	1	1	0.99	34	0.99
WS	1	1	40	1	1	0.99	32	0.98
SO	**0.93**	0.93	**791**	**0**				

achieves only 93% of the examples, while the final rule set is much more complex than the other approaches (with 791 different rules) and the optimal population is not reached at all.

In the 5-par problem, PR has converged with 0.78 of accuracy and only 60% of the expected optimal population. In fact, the population has collapsed to the non-inferior solutions, which represent only two points in the objective space. The first one, with objective vector $\mathbf{y}_1 = (0.03, 1)$, corresponds to rules of type: 00010 : 1. The second one, with vector $\mathbf{y}_2 = (0.5, 0.5)$, corresponds to the overgeneral rules: ##### : 1 and ##### : 0. Once the population has converged to these points, it is very difficult to increase the number of specific rules, because of the presence of too many #'s in the population schemata. This is also the reason for such a small set size.

5 Application to a Real-World Classification Problem

In this section, we apply MOLeCS to the Wisconsin breast cancer database, obtained from the UCI repository [4]. The database contains 699 instances, with 9 numerical attributes ranging from 1 to 10, and two classes (benign or malignant). There are 16 instances with missing attribute values. The class distributions are unbalanced, having 458 (65.5%) benign and 241 (34.5%) malignant instances.

As the describing attributes are numerical and not binary, we must consider again our rule representation. First, we can discretize each numerical feature into a string of bits. This allows us to maintain our binary representation in the rules. The second possibility is to represent a rule as a set of real-valued intervals, as proposed by Wilson in [23]. We have implemented and tested both representations, without significant differences for the Wisconsin database.

Each experiment is averaged for five different seed numbers. Accuracy is estimated using ten-fold cross-validation (for details see [20]). The results reported in table 2 show the covering, the crude accuracy and the corrected accuracy,

Table 2. Results using the Wisconsin database, obtained with a ten-fold cross-validation experiment. We compare the four multiobjective strategies and the single-objective algorithm, using DC as crowding method. The table reports: **Cov** (ratio of covered examples), **CA** (crude accuracy) and **CCA** (corrected crude accuracy).

	Cov	CA	CCA
PR	1	0.65	0.65
PAR	0.98	0.97	**0.95**
AGR	0.98	0.97	**0.95**
WS (.75,.25)	0.98	0.94	0.92
SO	0.94	0.95	0.90

measured on test sets. MOLeCS is run with DC and the four multiobjective strategies. We also ran the single-objective optimization algorithm, obtaining a corrected accuracy of 0.90. PR achieved a result of 0.65, while PAR and AGR reached the maximum value, with 0.95 of accuracy. This confirms the results obtained with the multiplexer and parity problems. First, it seems suitable to apply multiobjective methods in order to optimize each rule accuracy and generality. Enforcing only the accuracy leads the system to develop a high number of specific rules. This makes the learning more difficult, because more rules are needed to describe the problem. SO resulted in less covering and even less crude accuracy than PAR and AGR. If we give the same pressure (or preference) to accuracy than to generality (as PR does) we degrade the final rule set accuracy, as happened with 5-par problem. Therefore, the best learning performance is achieved when we optimize generality and accuracy, but with a preference towards accuracy as it is implemented by the methods PAR and AGR. The obtained accuracy is of 0.95 ± 0.016 with a 95% confidence interval, which is comparable to other learning classifier systems. XCS [24] also reached an accuracy of 0.95 in a similar experiment using the same database.

6 Conclusions and Future Work

This paper has studied the performance of MOLeCS using different MOEA techniques. The results are compared to a single-objective EA optimizing only the accuracy goal, in order to prove the suitability of the multiobjective approach. The experiments with single-objective optimization demonstrate that the system evolves too many specific rules. This produces an enhancement of the solution set, making the learning more difficult and achieving poor covering. If we optimize the accuracy and generality of each rule, we improve the learning performance. Nevertheless, giving the same importance to these attributes (as Pareto ranking does) makes the system evolve overgeneral rules in the search process, preventing other maximally general rules from being explored and maintained. The overall accuracy of the final rule set is thus degraded. In this sense, including the decision preferences in the search (e.g., with PAR or AGR) leads up

to a better achievement of the learning goals. The results with the Wisconsin database have reached an accuracy of 0.95, performing as well as XCS.

The use of niching methods in MOLeCS is necessary to ensure the maintenance of a set of rules that covers the examples. The paper has studied different crowding algorithms under a non-generational scheme. In terms of covering and stability the best results are obtained with deterministic crowding.

As a future work, it is necessary to perform further investigation on the applicability of MOLeCS to real world databases. We can test the MOLeCS performance on more complex problems, having more describing attributes, performing multiple categorization rather than binary, etc.

When we deal with medical databases with two unbalanced classes, it is interesting to distinguish between the correct predictions made by the system when the true decision is "benign" and the correct predictions when the true decision is "malignant". In this case, the accuracy measure does not give enough information. Other measures as sensitivity, specificity and area under the ROC curve must be included in our further analysis.

Another important future research with MOLeCS is to study our approach with problems with highly unbalanced classes. Giving a pressure towards generalization might displace specific rules (that cover few examples from a certain class) by other general rules (covering examples from other more numerous classes). This can be prevented in the replacement stage or by measuring the generality of each rule relatively to its niche.

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 authors acknowledge the support provided by Epson Iberica, under 1999 Rosina Ribalta Award, and the support of *Enginyeria i Arquitectura La Salle*.

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 and Josep Maria Garrell i Guiu. MultiObjective Learning in a Genetic Classifier System (MOLeCS). In *Butlletí de l'ACIA, 22. 3r Congrés Català d'Intel.ligència Artificial*, 2000.
4. C.L. Blake and C.J. Merz. UCI Repository of machine learning databases, [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
5. 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.

6. K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems. (Doctoral Dissertation)* . PhD thesis, University of Michigan, 1975.
7. Carlos M. Fonseca and Peter Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
8. P.W. Frey and D.J. Slate. Letter recognition using Holland-style adaptive classif iers. *Machine Learning*, 6:161–182, 1991.
9. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.
10. D.E. Goldberg and P. Segrest. Finite Markov chain analysis of Genetic Algorithms. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8. Lawrence Erlbaum, 1987.
11. 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.
12. John H. Holmes. Quantitative Methods for Evaluating Learning Classifier System Performance in Forced Two-Choice Decision Tasks. In *Second International Workshop on Learning Classifier Systems (IWLCS-99)*, 1999.
13. 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, 1994.
14. Tim Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions. In Roy, Chawdhryand, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, 1997.
15. 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.
16. Mahfoud, Samir W. *Niching Methods for Genetic Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.
17. Manuel Valenzuela-Rendón and Eduardo Uresti-Charre. A Non-Generational Genetic Algorithm for Multiobjective Optimization. *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665, 1997.
18. David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Dept. of Electrical and Computer Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.
19. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–174, Summer 2000.
20. Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn. Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems.* Morgan Kaufmann, 1991.
21. Stewart W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
22. 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.
23. Stewart W. Wilson. Get Real! XCS with Continuous-Valued Inputs. In L. Booker, S. Forrest, Mitchell M., and Riolo R, editors, *Festschrift in Honor of John H. Holland.* Center for the Study of Complex Systems, University of Michigan, 1999.
24. Stewart W. Wilson. Mining Oblique Data with XCS. In *Third International Workshop on Learning Classifier Systems (IWLCS-2000)*, 2000.