Accuracy, Parsimony, and Generality in Evolutionary Learning Systems via Multiobjective Selection

Xavier Llorà[1], David E. Goldberg[1], Ivan Traus[2], and Ester Bernadó[2]

[1] Illinois Genetic Algorithms Laboratory (IlliGAL)
National Center for Supercomputing Applications,
University of Illinois at Urbana-Champaign,
104 S. Mathews Ave, Urbana, IL 61801
{xllora,deg}@illigal.ge.uiuc.edu
[2] Enginyeria i Arquitectura La Salle,
Universitat Ramon Llull,
Psg. Bonanova 8, 08022, Barcelona,
Catalonia, Spain, European Union
{is06376,esterb}@salleURL.edu

Abstract. Evolutionary learning systems (also known as *Pittsburgh* learning classifier systems) need to balance accuracy and parsimony for evolving high quality general hypotheses. The learning process used in evolutionary learning systems is based on a set of training instances that sample the target concept to be learned. Thus, the learning process may overfit the learned hypothesis to the given set of training instances. In order to address some of these issues, this paper introduces a multiobjective approach to evolutionary learning systems. Thus, we translate the selection of promising hypotheses into a two-objective problem that looks for: (1) *accurate* (low error), and (2) *compact* (low complexity) solutions. Using the proposed multiobjective approach a set of compromise hypotheses are spread along the Pareto front. We also introduce a theory of the impact of noise when sampling the target concept to be learned, as well as the appearance of overfitted hypotheses as the result of perturbations on high quality generalization hypotheses in the Pareto front.

1 Introduction

This paper deals with a learning classifier system (LCS) [1–4] known as an evolutionary learning system (ELS), or *Pittsburgh* classifier system [5–8]. Among other characteristics, evolutionary learning systems use variable-size individuals. The main reason for using this kind of individual is because an individual must be a complete solution to the classification problem. That is, an individual codifies an hypothesis of the target concept to be learned. In order to perform the learning, ELSs use a set of instances that are a sample of the target concept to be learned. Thus, like other machine learning algorithms, ELSs assume that the target concept do not change over time. Some variable-size knowledge representation of hypotheses often used in ELSs include rule sets, instance sets, or decision trees [9].

Traditionally, the evolutionary-driven learning process of ELSs has focused on the evolution of accurate hypotheses that correctly classify the available training instances. However, this approach does not solve some relevant issues of machine learning algorithms [10]. The first one is described by *Occam's razor*. Given two equally accurate hypotheses, we prefer the simplest one. Thus, if the hypotheses are represented as a set of rules, we prefer the one with fewer rules, or in other words, the most general and accurate hypothesis that describes the target concept. The second issue is the quality of the evolved hypothesis in terms of generalization accuracy (accuracy of an hypothesis given unseen instances of the target concept). Since ELSs use a training set that is a sample set of the target concept, an ELS may evolve an overfitted hypotheses to the given training data set. This is a critical issue in real-world learning problems. Therefore, the evolved hypotheses, in order to achieve a high quality generalized accuracy on the target concept, must avoid over-adapted solutions. If not, the hypotheses may have poor performance when tested on unseen instances of the target concept.

Another problem that ELSs have to address is the *bloat* phenomenon. *Bloat* can be defined as the individual size growth without fitness improvement. This problem is well-known in the genetic programming (GP) community [11–22], as well as in ELSs [23–25], especially when the ELSs are used for solving data-mining tasks.

Some efforts to address *Occam's razor* and *bloat* introduce direct penalties also known as *parsimony pressure*. The goal of parsimony pressure is to bias the evolution of the hypotheses toward solutions that balance accuracy and size. Introducing explicit selection bias toward generalization, a common parsimony pressure introduces a static (or adaptive) tradeoff between the accuracy and the size of the evolved hypotheses, constraining the search path of the evolutionary algorithm. This fact often guides the learning algorithm toward a collapsed population where all the individuals represent the most general hypothesis. In other words, if the hypotheses are represented using a set of rules, the individuals codify hypotheses that contain only one rule that matches everything.

In order to address the generalization issues discussed previously, here we address parsimony and generality by transforming an ELS into a two-objective problem. This multiobjective approach uses two different objectives for a given hypothesis: (1) *accuracy* (low error), and (2) *compactness* (low complexity hypotheses under the *Occam's razor*). The first one guarantees that we solve the problem accurately, whereas the second introduces generalization pressure toward compact solutions. Our proposal is based on the concept of a Pareto optimal set [26–29]. Roughly explained, we want to spread the evolved hypotheses over the Pareto front of the learning problem. Thus, the multiobjective selection pressure coevolves different solutions with different tradeoffs between accuracy and complexity. Therefore, once the evolutionary learning process is done, we will be able to choose among the different hypotheses (and their associated tradeoff). Moreover, the Pareto front of the evolved population lets us gain some theoretical insights on the behavior of the proposed multiobjective ELSs. Our analysis is twofold. On one hand, we study the effect of noise when sampling the target concept. On the other hand, we also make some considerations about the overfitting of the evolved hypothesis in terms of the Pareto front of the population.

The paper is structured as follows. Section 2 presents some related work from GP and ELSs efforts for controlling *bloat*, as well as some work done using multiobjective optimization. Then, section 3 presents a description of the multiobjective fitness evaluation proposed to achieve the goals of this paper. Section 4 describes how the multiobjective fitness evaluation is used in two learning systems (the first one based on genetic algorithms (GA), whereas the second relies on evolution strategies (ES)). Some experiments using both algorithms are summarized in section 5. Finally, section 6 discusses the conclusions of this work, as well as some future work.

2 Related Work

One of the main problems that arises with the evolution of variable-size individuals is the *bloat* phenomenon [30]. *Bloat* is usually defined (in GP terms) as the code growth of individuals without any fitness improvement. Unfortunately ELSs that use variable-size representations also suffer from *bloat*. In ELSs, this phenomenon may appear in two different forms: (1) the addition of useless rules, or (2) the evolution of over-specific rules.

Early works in GP reported unexpected code growth of individuals that did not improve fitness [11–13, 30]. This phenomenon was called *bloat* [30]. Banzhaf and Langdon [21] categorize being of *bloat* as two disputed types. The first is known as "fitness causes *bloat*" [16], whereas the second is referred as "natural code is protective" [15]. Fitness selection bias favors individuals with the same fitness regardless of their size. This means that given an individual, there is a set of individuals (almost infinite) that share the same fitness value, but with a larger code. Therefore, once a given fitness value has been reached, the search becomes a random walk among these bigger individuals without an improvement in fitness. On the other hand, *bloat* also appears as neutral code that does not take part in fitness computation. This neutral code increases the size of the individual and as a consequence it reduces the probability that the genetic operators disrupt useful code.

Many different approaches and studies to control code growth have been developed in the GP community [14, 17–19, 22]. Some of them impose a *parsimony pressure* toward compact individuals by varying fitness or through specially tailored operators, among others. Recently, an approach has been proposed by Bleuler, Brack, Thiele, and Zitzler [20] in which they address *bloat* as a multiobjective optimization problem. The two objectives are to: (1) maximize fitness, and (2) minimize size. In order to achieve this goal, they used a multiobjective evolutionary algorithm known as SPEA2 [31–33].

Several authors have studied the growth of individuals in Pittsburgh classifier systems, but they have not addressed this problem from a multiobjective point of view. The most common approach is to introduce a parsimony pressure in the fitness function, in such a way that the fitness of larger individuals is decreased [24, 25, 23]. For example, in [23] the *bloat* is controlled by a step fitness function: when the number of rules of an individual exceeds a certain maximum, its fitness is decreased abruptly. One problem of this approach is to set this threshols value appropriately. Bacardit and Garrell [25] define a similar fitness function, as well as a set of operators for the deletion of introns[1]

[1] Non-coding segments. In GP literature this concept has also been termed *non-effective code* [21].

[34] (rules that are not used in the classification) and a tournament-based selection operator that considers the size of individuals. The authors argue that the *bloat* control has an influence over the generalization capability of the solutions. It has been observed that shorter rule sets tend to have more generalization capabilities [35, 25, 34].

Therefore, the use of a parsimony pressure has beneficial effects: it controls the unlimited growth of individuals, increases the efficiency in the search process and leads to solutions with better generalization. Nevertheless, the parsimony pressure must be balanced appropriately. An excessive pressure toward small individuals could result in premature convergence leading to compact solutions but with suboptimal fitness [34], or even in a total population failure (population collapses with individuals of minimal size). Soule and Foster [17] showed that the effect of parsimony pressure can be measured by calculating explicitly the relationship between the size and the performance of individuals within the population. Based on these results, it seems that a multiobjective approach may overcome some of these difficulties. Instead of balancing the parsimony pressure, a multiobjective approach based on the concept of the Pareto front [26–29] can coevolve a set of solutions with different tradeoffs between size and accuracy. Spreading these solutions among the Pareto front implicitly balances the relationship between the generality and the performance of individuals within the population. Doing so may also have the benefit of providing a richer set of diverse rules to enhance the search capability of the scheme.

In the field of evolutionary fuzzy models, there have been some proposals with the use of multiobjective techniques. Gómez-Skarmeta, Jiménez, and Ibáez [36] use a multiobjective evolutionary algorithm to generate and tune fuzzy models. The system obtains a collection of fuzzy rule sets along the discovered Pareto front, which is defined by the minimization of two objectives: the quadratic mean error and the number of rules. Although the minimization of the number of rules is an objective included in the evolutionary search, the number of rules is previously limited by a parameter tuned by the user. In their work, the multiobjective algorithm is used as a tool for providing multiple solutions to the decision maker, who has to decide *a posteriori* the best solution according to the problem environment. Jiménez, Gómez-Skarmeta, Roubos, and Robert [37] also define a multiobjective evolutionary algorithm to obtain fuzzy models. They identify several objectives such as the accuracy, the similarity between fuzzy sets, and the number of rules, but their final approach does not include the number of rules as an objective to minimize because, according to the authors, it led to sub-optimal solutions.

Our proposal uses a multiobjective evolutionary approach which minimizes the classification accuracy and the size (number of rules). Besides controlling the number of rules (*bloat*) dynamically, this would allow the formation of compromise hypotheses. This explicit tradeoff formation let us explore the generalization capabilities of the hypotheses that form the Pareto front. In certain environments like data mining, where the extraction of explanatory models is desirable, high quality general solutions (in terms of accuracy out of sample, or compactness of hypotheses) are useful. For instance, the presence of noise in the data set may lead to accurate but overfitted solutions. Maintaining a Pareto front of compromise solutions we can identify the overfitted perturbations of high quality general hypotheses. Therefore, evolving a set of different compromise solutions between accuracy and generalization, we can postpone the decision of picking the "best rule set" to the final user (decision maker), or combine them all using some *bagging* technique [38, 39, 9].

3 Multiobjective Evolution and Evolutionary Learning Systems

Since multiobjective optimization plays a central role in the work presented here, this section summarizes some relevant issues. First, we briefly summarize some multiobjective optimization definitions in subsection 3.1. Then, subsection 3.2 presents how we can use a multiobjective approach to address the *bloat* phenomenon that usually appears on variable-size individuals evolved in some learning systems. Finally, subsection 3.3 discusses the usefulness of evolving a classification front in a learning system.

3.1 Multiobjective Optimization

In a multiobjective optimization problem (MOP) [29] a solution $\vec{x} \in \Omega$ is represented as a vector of n decision variables $\vec{x} = (x_1, \ldots, x_n)$, where Ω is the decision variable space. We want to optimize k objectives which are defined as $f_i(\vec{x})$, with $i = 1 \ldots k$. These objectives are grouped in a vector function denoted as $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$, where $F(\vec{x}) \in \Lambda$. F is a function which maps points from the decision variable space Ω to the objective function space Λ:

$$\begin{aligned} F : \Omega &\longmapsto \Lambda \\ \vec{x} &\longmapsto \vec{y} = F(\vec{x}) \end{aligned} \tag{1}$$

Without loss of generality, we can define a MOP as the problem of minimizing a set of objectives $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$, subject to some constraints $g_i(\vec{x}) \leq 0$, $i = 1, \ldots, m$. These constraints are necessary for problems where there are invalid solutions in Ω. Although the MOP's definition addresses a minimization problem, MOP is not limited exclusively to minimization. MOP can be applied to maximization problems as well as to problems where some objectives must be minimized and some others maximized. Nevertheless, in the rest of this section we will assume a minimization MOP.

A solution that minimizes all the objectives and satisfies all constraints may not exist. Sometimes, the minimization of a certain objective implies a degradation in another objective. Then, there is not a global optimum that minimizes all the objectives simultaneously. In this context, the concept of optimality must be redefined. Vilfredo Pareto [26] introduced the concept of dominance and Pareto optimum to deal with this issue.

In general terms, a vector $\vec{u}$ *dominates* another vector $\vec{v}$, written as $\vec{u} \preceq \vec{v}$, if and only if every component u_i is less or equal than v_i, and at least there is one component in $\vec{u}$ which is strictly less than the corresponding component in $\vec{v}$. This can be formulated as follows:

$$\vec{u} \preceq \vec{v} \iff \forall i \in 1, \ldots, k\,,\ u_i \leq v_i \wedge \exists i \in 1, \ldots, k : u_i < v_i \tag{2}$$

For example, given a MOP with three objectives and the vectors $\vec{u} = F(\vec{x}_1) = (1, 1, 2)$ and $\vec{v} = F(\vec{x}_2) = (1, 2, 2)$, we notice that $\vec{u} \preceq \vec{v}$. However, if $\vec{u} = (1, 1, 2)$ and $\vec{v} = (1, 2, 1)$, neither $\vec{u}$ dominates $\vec{v}$ nor $\vec{v}$ dominates $\vec{u}$.

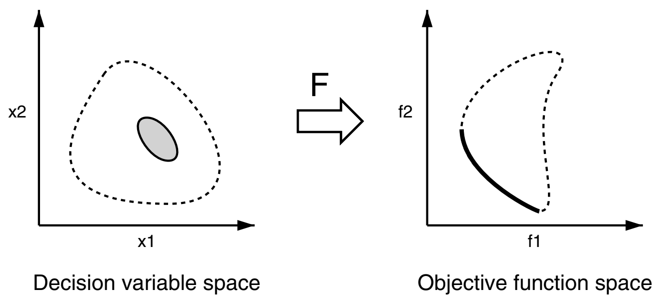


Fig. 1. In this hypothetical example we have a two objective problem with two decision variables. Solutions are mapped from the decision variable space to the objective function space. The shadowed area in decision variable space represents the Pareto optimal set and the continuous line in objective function space the Pareto front. A simple mathematical example of this kind of behavior can be displayed using $f_1(x_1) = (x_1 - 2)^2$ and $f_2(x_2) = (x_2 - 4)^2$.

The concept of *Pareto optimality* is based on the dominance definition. Thus, a solution $\vec{x} \in \Omega$ is Pareto optimal if there is not any other solution $\vec{x}' \in \Omega$ whose objective vector $\vec{u}' = F(\vec{x}')$ dominates $\vec{u} = F(\vec{x})$. In other words, a solution whose objectives can not be improved simultaneously by any other solution is Pareto optimum.

The set of all solutions whose objective vectors are not dominated by any other objective vector is called the Pareto optimal set $\mathcal{P}^*$:

$$\mathcal{P}^* := \left\{ \vec{x}_1 \mid \nexists\, \vec{x}_2 : \vec{F}(\vec{x}_2) \preceq \vec{F}(\vec{x}_1) \right\} \tag{3}$$

Analogously, the set of all vectors $\vec{u} = F(\vec{x})$ such that $\vec{x}$ belongs to the Pareto optimal set is called the Pareto Front $\mathcal{PF}^*$:

$$\mathcal{PF}^* := \left\{ \vec{u} = \vec{F}(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x})) \mid \vec{x} \in \mathcal{P}^* \right\} \tag{4}$$

Figure 1 represents a two objective problem in the decision space (left) and the objective space (right). The shadowed area on the left represents the optimal solutions in the decision space, also called as Pareto optimal set. The thick curve in the objective function space represents the Pareto Front. It is constituted by the objective vectors which are not dominated by any other objective vectors. These solutions represent compromise solutions, i.e., solutions with different tradeoffs between the objectives. We cannot improve an objective without penalizing the other one. Among these compromise solutions, we have to choose the desired tradeoff between the different objectives in order to take the best solution to the problem. This decision is done by a *decision maker*, a human or an expert system with some knowledge about the problem.

3.2 Classification and Multiobjective Optimization

The goal of our multiobjective approach introduced in ELSs is to trade off two objectives: (1) the accuracy of an individual, and (2) its size. In particular, we are interested in evolving accurate (with a reduced classification error) general solutions to the classification problem. This means that we prefer maximally general solutions [3] describing the knowledge pattern behind that classification data. In an ELS where an individual is a complete solution to the classification problem, this can be achieved biasing the evolution toward compact (small sized) individuals. Therefore, we have two different objectives to optimize at the same time, accuracy and size.

Let's define $\vec{x}$ as an individual that is a complete solution to the classification problem; $\mathcal{D}$ the training data set for the given problem; $|\mathcal{D}|$ number of instances in $\mathcal{D}$; $miss(\vec{x}, \mathcal{D})$ the number of incorrectly classified instances of $\mathcal{D}$ performed by $\vec{x}$; and finally, $size(\vec{x})$ a measure of the current size of $\vec{x}$ (e.g. the number of rules it contains). Using this notation, a simple multiobjective approach can be defined as follows:

$$\min F(\vec{x}) = (f_e(\vec{x}), f_s(\vec{x})) \tag{5}$$

$$f_e(\vec{x}) = \frac{miss(\vec{x}, \mathcal{D})}{|\mathcal{D}|} \tag{6}$$

$$f_s(\vec{x}) = size(\vec{x}) \tag{7}$$

Thus, our multiobjective approach minimizes $F(\vec{x})^2$. With this simple multiobjective definition, the evolution is biased toward the hypotheses that form the Pareto optimal set. Therefore, the population may evolve hypotheses with different tradeoffs between accuracy and generality. Besides, the *bloat* phenomenon is also addressed due to the bias toward the Pareto front. For implementation purposes f_s was divided by the data set size (number of available instances).

3.3 What Is the Purpose of the Classification Front?

The main purpose of the evolved Pareto front (or classification front) is to keep solutions with different tradeoffs between accuracy and size. Coevolving these compromise solutions, we can delay the need of choosing a solution until the evolution is over and the classification front is provided. However, this decision is critical for achieving a high quality accuracy generalization when tested with unseen instances of the target concept.

The *decision maker* has several hypotheses among which to choose, all provided by the classification front. The *decision maker* can be a human or an expert system with some knowledge about the problem. However, there are other approaches already explored in the ML and GBML community. Among others, some interesting approaches are based on the *bagging* technique [38, 39]. The goal is to combine different hypotheses (e.g. the ones that form the classification front) into a new single classification hypothesis. The goal is to obtain a new combined hypothesis that reduce the impact of the overfitting of the used hypotheses, producing a high quality general hypothesis. This technique tends to reduce the deviation among runs, and it often improves the generalization capability of the combined solution [9].

However, in this paper we use a simpler approach. In order to test unseen instances, we pick only one solution from the evolved classification front. This solution may be

[2] We used error instead of accuracy for simplifying the implementation details.

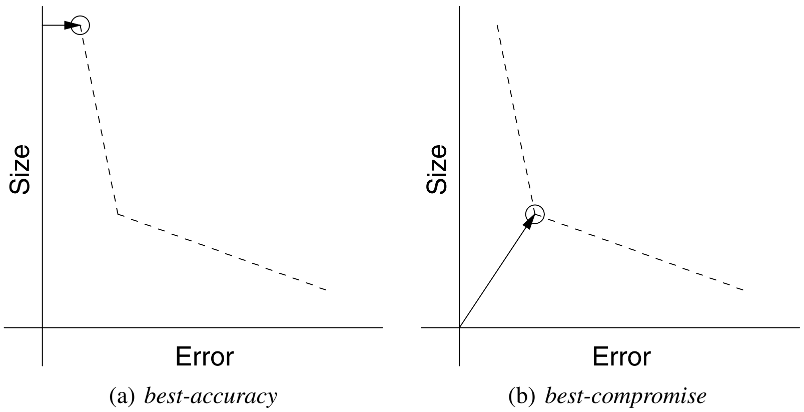


Fig. 2. Selected hypotheses, using two different strategies, among the Pareto front compromise solutions for testing unseen instances.

chosen using one of the two different strategies, shown in figure 2. The first one (*best-accuracy*) chooses the solution $\vec{x}$ of the front with the best accuracy, that is, the one that minimizes $f_e(\vec{x})$. On the other hand, the second one (*best-compromise*) picks the hypothesis of the front that minimizes the objective vector $\vec{u} = F(\vec{x})$. Thus, the selected solution is the one that balances equally both objectives. In other words, the solution $\vec{x}$ that minimizes $|F(\vec{x})| = \sqrt{f_e(\vec{x})^2 + f_s(\vec{x})^2}$. For further insight about this approach is provided on section 5.3.

4 Multiobjective Learning Systems

There are several approaches to LCSs [2, 8, 9]. Among the different alternatives, we chose to implement our multiobjective approach, presented in section 3, using two different ELSs. The first one uses an evolutionary model based on genetic algorithms (MOLS-GA), whereas the second exploits an evolutionary learning approach based on evolution strategies (MOLS-ES) [40]. Both ELSs share some common elements, mainly related to the multiobjective mechanisms. This section describes briefly the two systems, and afterwards it focuses on the evaluation phase where the multiobjective techniques are introduced.

4.1 MOLS-GA

MOLS-GA is a learning system based on genetic algorithms. The knowledge representation is based on rule sets or instance sets [9, 41, 42]. If the problem's attributes are nominal, MOLS-GA uses rule sets, represented by the ternary alphabet (`0`, `1`, `#`) often used in other LCSs [1–3]. Otherwise, if the problem is defined by continuous-valued attributes, instance sets—based on a nearest neighbor classification—are used.

The GA learning cycle works as follows. First, the fitness of each individual in the population is computed. This is done on a multiobjective basis, taking into account the misclassification error and the size of each individual. This phase is explained in details in section 4.3. Then, selection is applied using a tournament selection algorithm [43–45] with elitism. Elitism is often applied in evolutionary multiobjective optimization algorithms and it usually consists in keeping the solutions of the Pareto Front evolved in each generation [29]. MOLS-GA performs similarly: it keeps all the distinct solutions of the evolved Pareto Front, and also a 30% of the individuals with the lowest error. This guarantees that the best compromise solutions evolved so far are not lost, as well as the best low-error solutions which are important to drive the evolution toward accurate solutions.

After selection, crossover and mutation are applied. The crossover operator is based on the operator described in [6]. It is a variant of the classical two-point crossover, adapted to deal with variable-size individuals. It works in the following way. The crossover point can occur anywhere (i.e., both on the rule/instance boundaries as well as within a rule/instance). The only requirement is that the crossover points in the two parents must be equivalent in order to produce valid solutions. That is, if one parent is cut on a rule/instance boundary, then the other parent must also be cut on a rule/instance boundary. Similarly, if one parent is cut within a rule/instance, then the other parent must be cut in a similar spot. The mutation consists in generating a random new gene value.

4.2 MOLS-ES

MOLS-ES [40] is a learning system that uses an evolution strategy scheme [46–48] instead of a genetic algorithm. Each individual of the population codifies a set of rules. The rules are represented in the ternary alphabet if the attributes are binary, and in a (n+1)-alphabet if the attributes are nominal (where n is the number of nominal values). If the attributes are real-valued, the hyper-rectangle codification proposed by Wilson [49] is used.

The multiobjective approach has been introduced in MOLS-ES in the same way as in MOLS-GA. It is described in the following section. Besides the fitness computation stage, the other phases of MOLS-ES differ from MOLS-GA, since MOLS-ES is based on an Evolution Strategy approach. MOLS-ES uses a $(\mu + \lambda)$ selection scheme, which means that from the recombination and mutation of μ parents λ children are obtained. From the resulting overlapping population, the best μ individuals are selected for the next generation, where the concept of *best* is defined according to the multiobjective evaluation algorithm. It can be noticed that this selection also induces a kind of elitism.

The crossover operator applied to the solutions is the same as in MOLS-GA, which is a two-point crossover adapted to the variable size of individuals. The mutation is applied according to Evolution Strategies. For each gen x_i, there is a standard deviation σ_i associated with it which is used to mutate the gen. Both the solutions and the standard deviations are mutated as follows:

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \tag{8}$$

$$x_i' = x_i + \sigma_i' \cdot N_i(0,1) \tag{9}$$

where $N(0,1)$ and $N_i(0,1)$ are random numbers distributed normally with mean 0 and standard deviation 1 and τ and τ' are set as recommended in [47]. Thus, the solutions

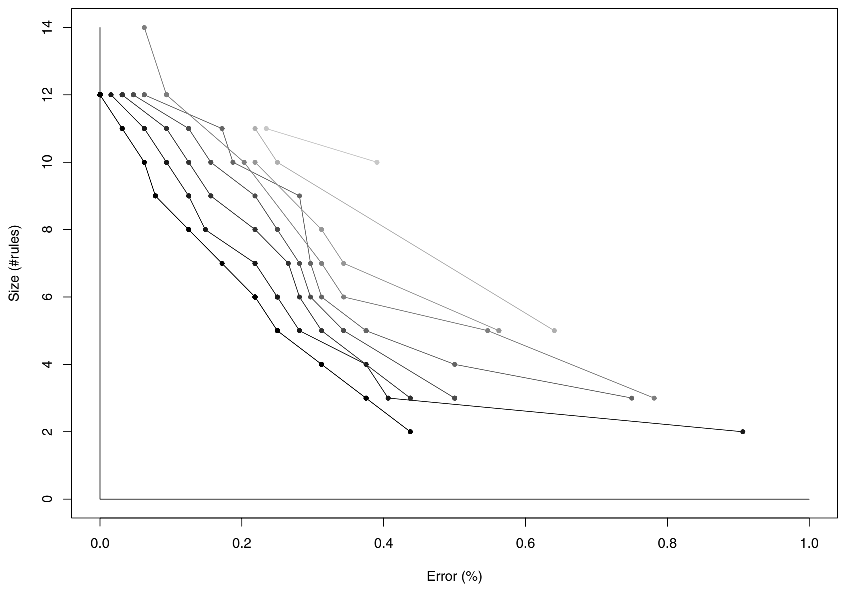


Fig. 3. Sorted population fronts at a given iteration of MOLS in the `mux` problem.

and their standard deviations are adapted along the evolution process. For real-valued attributes, this scheme fits perfectly. Nevertheless, for nominal attributes the mutation of x_i has to be modified. In this case, σ_i is proportional to the mutation probability. The new mutated x_i value is chosen randomly among the available symbols.

4.3 Multiobjective Fitness in MOLS-GA and MOLS-ES

In both systems, the multiobjective fitness scheme is introduced in the same way. The procedure is inspired by NSGA [50, 51] and NSGA-II [52] and it works as follows.

The individuals of the population are sorted in equivalent classes. These classes are determined by the Pareto Fronts that can be defined among the population. That is, given a population of individuals $\mathcal{I}$, the first equivalence class $\mathcal{I}^0$ is the set of individuals which belongs to the evolved Pareto optimal set $\mathcal{I}^0 = \mathcal{P}^*(\mathcal{I})$. The next equivalence class $\mathcal{I}^1$ is computed without considering the individuals in $\mathcal{I}^0$, as $\mathcal{I}^1 = \mathcal{P}^*(\mathcal{I} \setminus \mathcal{I}^0)$, and so forth. Figure 3 shows an example of the different equivalence classes, presented using the fronts that appear in a population at a given iteration. This plot is obtained with the `mux` problem. In this example, the population is classified into nine different fronts. The left front is $\mathcal{I}^0$, which corresponds to the non-dominated vectors of the population. The next front to the right represents $\mathcal{I}^1$ and so on.

Once the population of individuals $\mathcal{I}$ is sorted, fitness values are assigned. Since the evolution must bias the population toward non-dominated solutions, we impose the constraint:

$$fitness(\mathcal{I}^i) > fitness(\mathcal{I}^{i+1}) \tag{10}$$

Thus, the evolution will try to guide the population toward the left part of the plot, i.e., the real Pareto Front. The fitness of each individual depends on the front where the individual belongs. That is, all the individuals of the same equivalence class $\mathcal{I}^i$ receive the same constant value $(n-i)\delta$, where n is the number of equivalence classes and δ is a constant. Moreover, in order to spread the population along the Pareto Front, a *sharing* function is applied. Thus, the final fitness of an individual j in a given equivalence class $\mathcal{I}^i$ is:

$$fitness(\mathcal{I}^i_j) = \frac{(n-i)\delta}{\sum_{k \in \mathcal{I}} \phi(d_{\mathcal{I}^i_j \mathcal{I}_k})} \tag{11}$$

where $\phi(d_{\mathcal{I}^i_j \mathcal{I}_k})$ is the sharing function [53]. The *sharing* function is computed using the phenotipical distance between the individuals; that is, the Euclidean distance between their multiobjective vectors. The radius of the *sharing* function σ_{sh} was set to $\sigma_{sh} = 0.1$.

5 Experiments

This section discusses the results obtained using the multiobjective ELSs presented in section 4. The experiments explore different facets of the behavior of the proposed ELSs. Both ELSs were used to solve artificial and real-world problems, paying special attention to their performance in terms of the evolved Pareto fronts. The section starts describing briefly the data sets and algorithms used in the experiments (subsection 5.1). Then subsection 5.2 presents the fronts obtained on two artificial problems where the optimal Pareto front is known. After presenting these results, subsection 5.3 shows some interesting properties of the Pareto front evolved by the proposed ELS in the presence of noise in the data set. These results let us justify the results obtained in real-world problems. Thus, subsection 5.4 analyzes some interesting facets of the behavior of both ELSs on these real-world problems.

5.1 Test Suite

In order to evaluate the performances of the proposed multiobjective ELSs on different domains, we performed experiments on nine data sets. These data sets can be grouped into two different categories: artificial and real-world. Table 1 describes their characteristics.

We used two *artificial data sets* to tune both ELSs, because we knew their solutions in advance. `Mux` is the eleven input multiplexer, widely used by the LCS community [3]. `Led` is the seven-segments problem [54]. Given seven light emitting diodes that represent a digit (seven binary input attributes), the goal of the `led` problem is to identify the digit represented by the active diodes (the ten available classes). The data set used in the `Led` problem was generated using the program provided by the UCI repository [55].

The *public data sets* were obtained from the UCI repository [55]. We chose seven data sets: *Bupa Liver Disorders* (`bpa`), *Wisconsin Breast Cancer* (`bre`), *Glass* (`gls`), *Ionosphere* (`ion`), *Iris* (`irs`), *Primary Tumor* (`prt`), and *Sonar* (`son`). These data sets contain categorical and numeric attributes, as well as binary and n-ary classification tasks.

We also run several evolutionary and non-evolutionary classifier schemes on the previous data sets. The evolutionary classifier schemes were GALE [39, 41, 42, 9] and XCS [3, 56], whereas the non-evolutionary ones were IB1 [57], C4.5 [58, 59], and PART [60]. The non-evolutionary schemes were obtained from the *Weka* package [61] developed at the University of Waikato in New Zealand. The code is available from the http address: `http://www.cs.waikato.ac.nz/ml/weka`. These algorithms were run with the default configuration provided by their authors.

In order to allow the replication of the results presented in this section, we briefly summarize the settings used in MOLS-GA and MOLS-ES. The parameter values used in MOLS-GA were: σ_{sh}=0.1, δ=1000, pop_size=285, crossover probability p_χ=0.4, probability of mutation of an individual p_{mut}=0.25, and the gene perturbation probability p_{gen}=0.02. On the other hand, the parameter values used in MOLS-ES as follows: σ_{sh}=0.1, δ=1000, μ=50, λ=250, and ϵ_σ=0.01 (in the `son` ϵ_σ=0.0001). In the `led`, `mux` and `prt`, p_χ=0.5, σ_0=0.75, whereas in the rest of the problems p_χ=1, σ_0=1. The maximum number of iterations allowed in both ELSs were 250 iterations, exception made in the `led`, `mux` and `prt` problems where it was extended to 1000.

Table 1. Summary of the data sets used in the experiments.

id	Data set	Size	Missing values(%)	Numeric Attributes	Nominal Attributes	Classes
`bpa`	*Bupa Liver Disorders*	345	0.0	6	-	2
`bre`	*Wisconsin Breast Cancer*	699	0.3	9	-	2
`gls`	*Glass*	214	0.0	9	-	6
`ion`	*Ionosphere*	351	0.0	34	-	2
`irs`	*Iris*	150	0.0	4	-	3
`led`	*Led (10% noise)*	2000	0.0	-	7	10
`mux`	*Multiplexer (11 inputs)*	2048	0.0	-	1	2
`prt`	*Primary Tumor*	339	3.9	-	17	22
`son`	*Sonar*	208	0.0	60	-	2

5.2 Spreading the Population along the Pareto Front

The first results we present are obtained using both multiobjective ELSs on the two artificial data sets (`mux` and `led`). Initially, we used a version of the `led` data set free of noise, leaving noise considerations for the next subsection. In order to identify the optimal Pareto front, we analyze first the optimal solutions that should be obtained in each problem. These optimal solutions are shown in figure 4. For each problem, the optimal Pareto front can be obtained removing one rule iteratively. Each time we remove a rule, we compute the accuracy of the resulting rule set. Therefore the resulting objective vector $F(\vec{x})$ is represented as a point in the optimal Pareto front. The computed optimal Pareto fronts are printed in figure 5.

s_0	s_1	s_2	s_3	s_4	s_5	s_6		*Class*
1	1	1	1	1	1	0	:	0
0	0	0	0	1	1	0	:	1
1	0	1	1	0	1	1	:	2
1	0	0	1	1	1	1	:	3
0	1	0	0	1	1	1	:	4
1	1	0	1	1	0	1	:	5
1	1	1	1	1	0	1	:	6
1	0	0	0	1	1	0	:	7
1	1	1	1	1	1	1	:	8
1	1	0	0	1	1	1	:	9

(a) `led` problem

a_2	a_1	a_0	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7		o
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

(b) `mux` problem

Fig. 4. Optimal solutions for the `led` and `mux` problems. The `mux` problem has two optimal solutions using ordered activation of classifiers. We only show one of these solutions; the other can be obtained swapping the `0`s for `1`s of the i_j and o attributes.

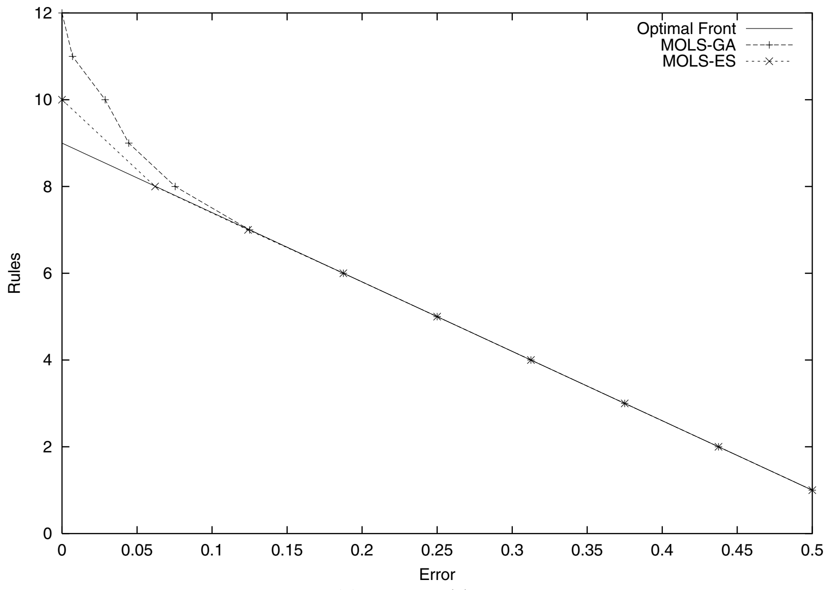
Figure 5 also shows the results obtained using both multiobjective ELSs. As it can be seen, both ELSs evolve the perfect Pareto front in the noise-free `led` problem. However, the results obtained in the `mux` problem are slightly different. The evolved Pareto fronts clearly approximate the optimal front. The evolved fronts differ in their top-left part, showing that some generalization of the solutions trapped in that part of the front are still needed. Inspecting the evolved solutions contained in the front, the differences are the result of still having (see the optimal solution in figure 4) more than one rule codifying class `1`. These extra rules can be removed if we increase the number of iterations of both ELSs.

Figure 6 shows the evolution of the learning performed by both multiobjective ELSs, averaged across five different runs. The error is the best-so-far obtained at a given iteration, whereas the size of the individuals (number of rules) is the average size of the population along the different runs. As it can be seen, the multiobjective approach easily balances the pressure toward accurate and compact (general) solutions. This approach can efficiently reduce at the same time both objectives.

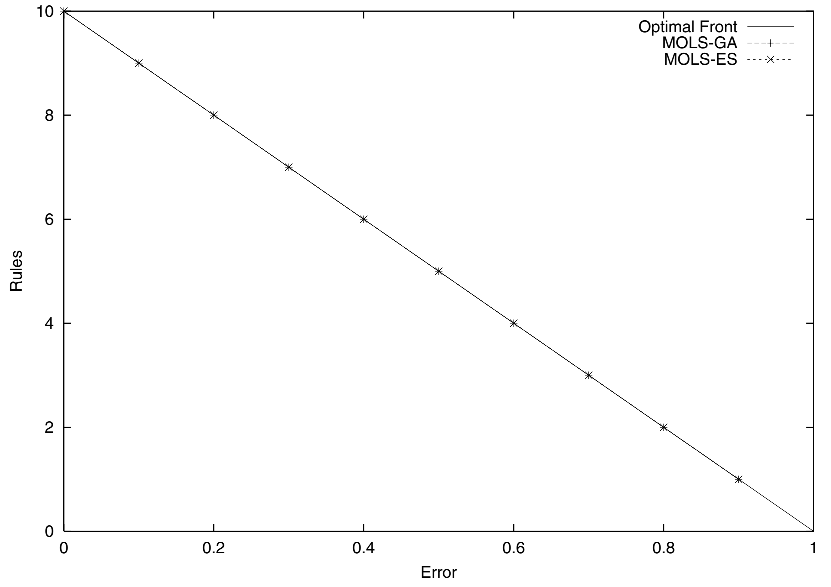
5.3 Noise in the Data Set

Recently, Llorà and Goldberg [62, 63] have shown the relevance of performing accurate noise analyses in ELSs theoretically. This kind of theoretical analyses helps in bounding the performance of ELSs. The remainder of this section introduces these results, focussing on the ELSs presented in this paper. This background becomes useful in the later analysis of the results obtained by the proposed ELSs when solving real-world problems.

The next experiment done introduced 10% noise in the `led` problem. Noise was introduced by swapping the antecedent values of the instances of the training data set with a probability equal to 0.1. This procedure is explained in detail elsewhere [62]. The instances of the noisy data set were generated using the program provided by the UCI repository [55]. The goal of this experiment was twofold. First, we were interested in the impact of the added noise on the performance of the two proposed multiobjective ELSs. Thus, when we solve real-world problems we easily would be able to understand



(a) mux problem

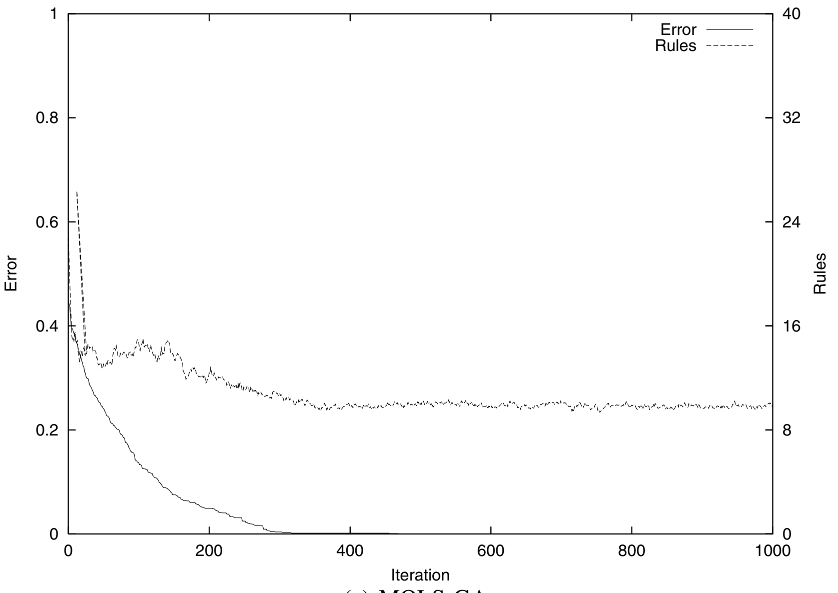


(b) led problem

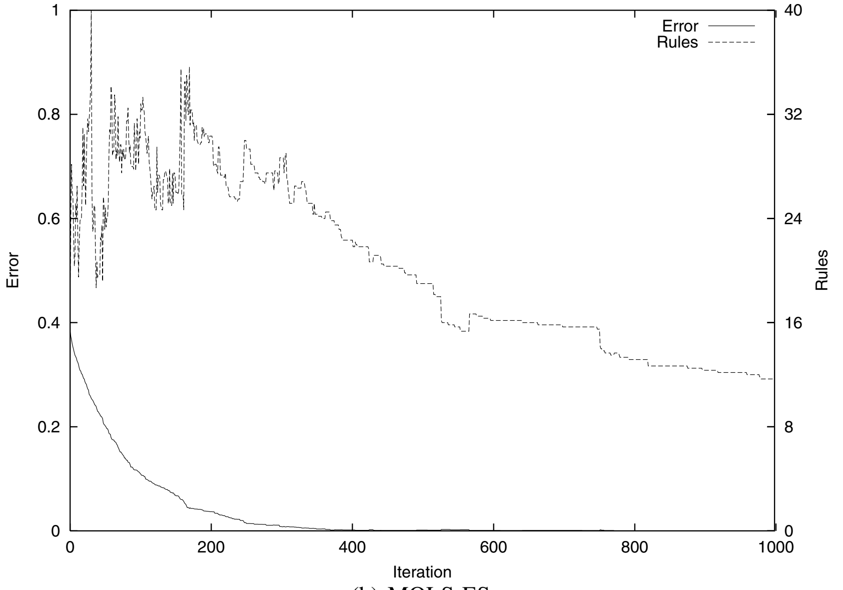
Fig. 5. Optimal Pareto fronts and evolved Pareto fronts achieved in `mux` and `led` problems.

the results that we obtained. The second reason for this experiment was to display the impact of the noise on the evolved Pareto front.

Before discussing the results, we need to introduce some theoretical results obtained for the noisy `led` problem. Detailed descriptions of these results can be found in [62,



(a) MOLS-GA



(b) MOLS-ES

Fig. 6. Evolution of the multiobjective ELSs in the `mux` problem.

63]. The optimal subset of instances $\mathcal{O}$ for the `LED` problem was shown in figure 4.a. The number of possible antecedents in the `led` problem is $2^7 = 128$. In the noise-free `led` problem, only ten of all the possible antecedents are part of $\mathcal{O}$, and thus part of the available data set $\mathcal{D}$. The remaining 118 antecedents only appear in $\mathcal{D}$ as the effect of the presence of noise. Therefore, we can intuitively understand the addition of noise as a disruptive element toward the appearance of inconsistencies[3] in $\mathcal{D}$.

As it has been proved elsewhere [62], the inconsistencies introduced by the noise addition bound the *minimal achievable error* (MAE) that a learning algorithm can reach on the noisy `led` problem. The main issue is that MAE only depends on the added noise ratio ϵ. MAE can be computed theoretically as follows. In order to simplify the notation we assume that each antecedent is indexed by the number it codifies (its binary number representation). Moreover, d_{ij} is the Hamming distance between antecedents i and j and ϵ is the noise ratio added to the data set. The first step is computing the *jumping matrix* $\mathcal{J}$. This matrix contains the jumping probabilities between antecedents. Rows represent the original antecedent (one of the ten that appear in $\mathcal{O}$, indexed by the instance's class $\chi(i)$), whereas the columns show the final antecedent of the instance after noise perturbation. This matrix $\mathcal{J}$ is then defined as

$$\mathcal{J}_{\chi(i)j} = \epsilon^{d_{ij}} \cdot (1-\epsilon)^{(7-d_{ij})}. \tag{12}$$

Using the jumping matrix $\mathcal{J}$, we can compute the probability distribution of the appearance of all possible antecedents α. Where α_a is the probability of appearance of the antecedent a on the noisy `led` data set $\mathcal{D}$. Moreover, using $\mathcal{J}$ we can also obtain the minimal classification achievable error e_a for a given antecedent a. The error e_a is the result of the inconsistencies that the noise ϵ introduces for each antecedent a. Thus, the minimal classification error is achieved only when we assign the majority class of the inconsistencies to the antecedent a. Having computed α and e, MAE is defined as follows:

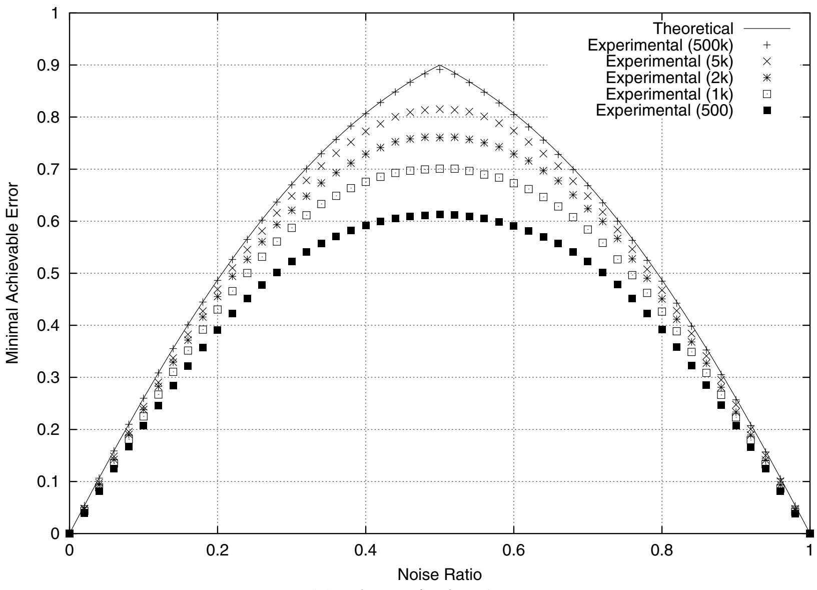
$$MAE = \sum_i \alpha_i e_i \tag{13}$$

Figure 7 shows the empirical validation of the MAE model for the `led` problem. The empirical data were obtained using different noisy $\mathcal{D}_\epsilon$ data sets sizes (500, 1,000, 2,000, 5,000, and 500,000 noisy instances) and computing the error based on the degree of inconsistencies.

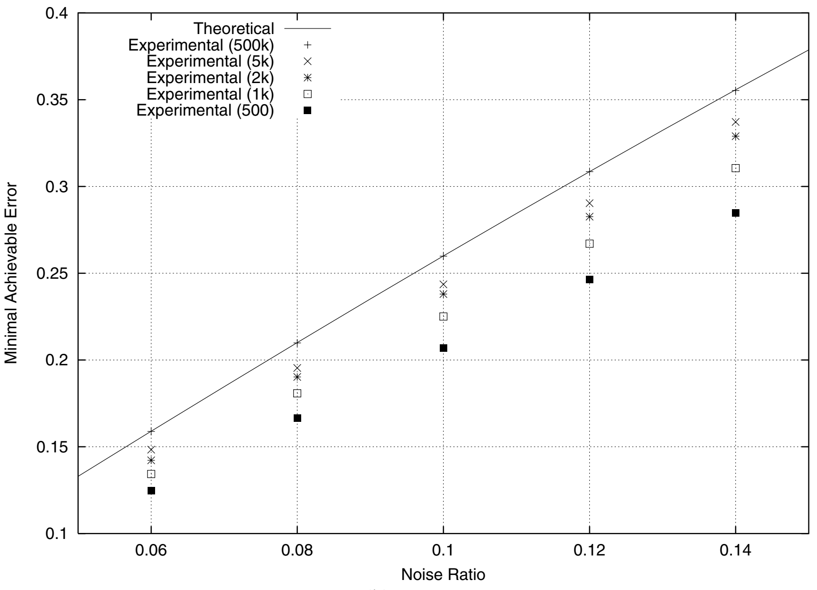
There are two interesting observations that arise from the data shown in figure 7. The first one is that, when enough instances are provided to the $\mathcal{D}_\epsilon$ data set, the theoretical model and the experimental results match perfectly (e.g. 500,000 instances). The second interesting observation provided by the results appears when we analyze the results achieved for the sizes of the $\mathcal{D}_\epsilon$ data set: 500, 1,000, 2,000, and 5,000. The empirical results using these small data sets show smaller MAE values that the ones theoretically predicted, showing some interesting deviations. These are the result of the random number generator bias used and the `led` instances distribution. Therefore, the experimental noise ratio ϵ is different than that theoretically expected, since not enough instances are generated. This fact leads to data sets that maintain some regularities that reduce the number of inconsistencies in $\mathcal{D}_\epsilon$.

Using these theoretical results we can now explain the Pareto front achieved in the noisy `led` problem. The data set contained 2,000 instances. These instances were generated using a noise ratio $\epsilon = .1$, that theoretically leads to a MAE equal to 0.26

[3] Two instances are inconsistent if they have the same antecedent, but different consequents.



(a) Theoretical MAE



(b) Zoom

Fig. 7. Theoretical and empirical *minimal achievable error* (MAE) in the noisy `led` problem.

(see figure 7.b). However, the empirical MAE obtained from the $\mathcal{D}_\epsilon$ used is 0.23. Figure 8 presents the Pareto front achieved by both multiobjective ELSs proposed. The figure also shows the theoretical and empirical MAE boundaries, as well as the optimal non-noisy Pareto front and size of $\mathcal{O}$. Let us call *rupture point* the point defined as (MAE($\epsilon =$

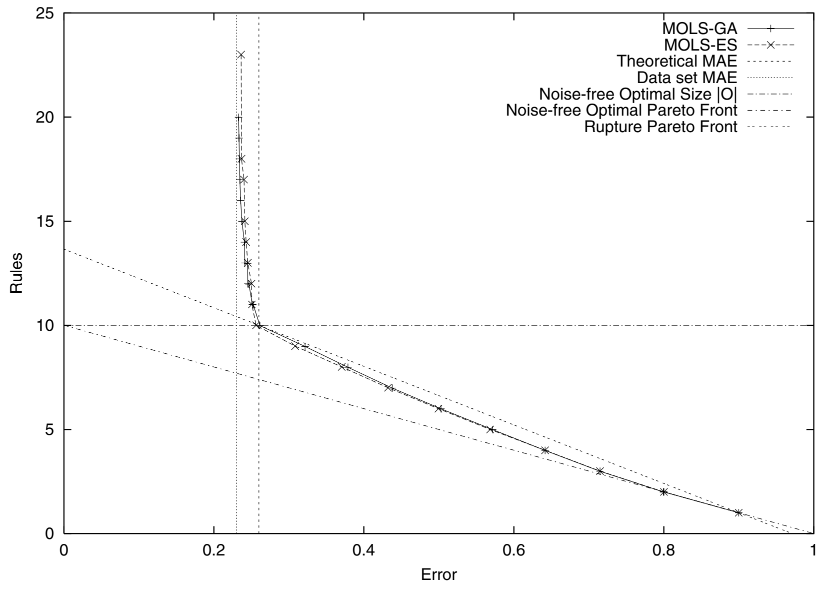


Fig. 8. Pareto fronts achieved in `led` problem. The data set contains 2000 instances perturbed with a noise ratio $\epsilon = 0.1$.

$0.1)$,$|\mathcal{O}|$)=(0.26,10). This point would be the performing point of $\mathcal{O}$ on a data set $\mathcal{D}_\epsilon$ with an experimental MAE equal to the theoretical one.

The *rupture point* indicates the place where the evolved Pareto front abruptly changes its slope. The front that appears to the left of the *rupture point* is the result of the deviation of the empirical MAE from its theoretical value. This has an interesting interpretation. All the points that define this segment of the front are over-fitted solutions. This means that they are learning some misleading noisy pattern as the result of the MAE value deviation. Therefore, if any of these points is tested using a different randomly generated $\mathcal{D}_\epsilon$ data set, they would experience a significant drop in accuracy. Thus, this leads to a reduction of the generalization capabilities (in terms of classification accuracy) of the solutions kept in that part of the front. Moreover, these solutions are closer to the *bloat* phenomenon, because very small (misleading) improvements require a large individual growth. All these problems disappear when we force the theoretical and the empirical MAE to be the same. This constraint removes the part of the front that appears at the left of the *rupture point*. Moreover, the optimal noisy Pareto front is bounded by the optimal noise-free front and the *rupture front* that appears between the *rupture point* and the *random guess point*. The *random guess point* is defined by the majority rule that describes solutions like $F(\vec{x}) = (0.9, 1)$.

5.4 Some Real-World Problems

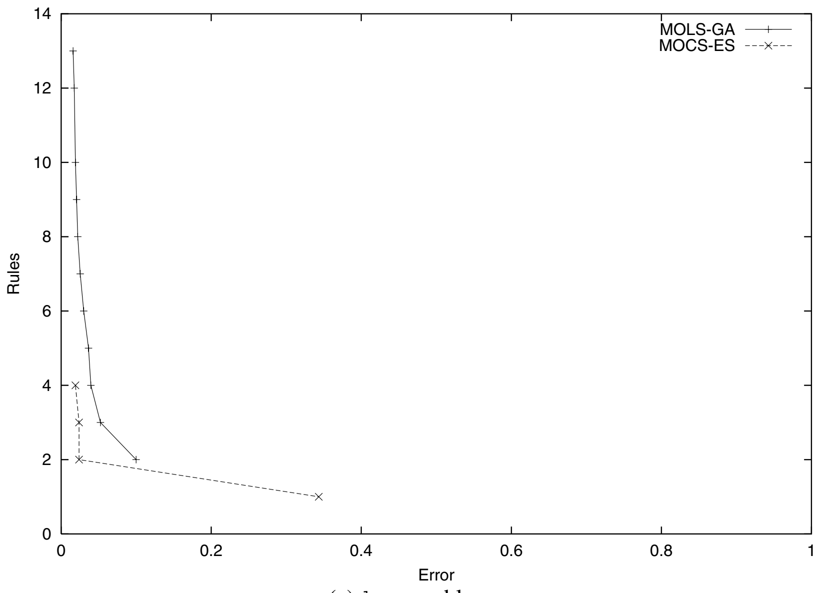
The last kind of experiments are focused on the real-world problems summarized in table 1. The results, shown in table 2, were obtained from *stratified ten-fold cross-validations runs* [10, 61] using the different learning algorithms on the selected data sets. MOLS-GA used a *best-accuracy* strategy for the test phase, whereas MOLS-ES used *best-compromise* (see figure 2). The main interest in these results is the fact that they prove the competence of the multiobjective approach. Moreover, for some particular data sets (like `bpa` or `prt`) some interesting improvements were achieved.

Table 2. Results obtained using the data sets presented in table 1. The table shows the mean and standard deviation of the *stratified ten-fold cross-validation runs* done using each system.

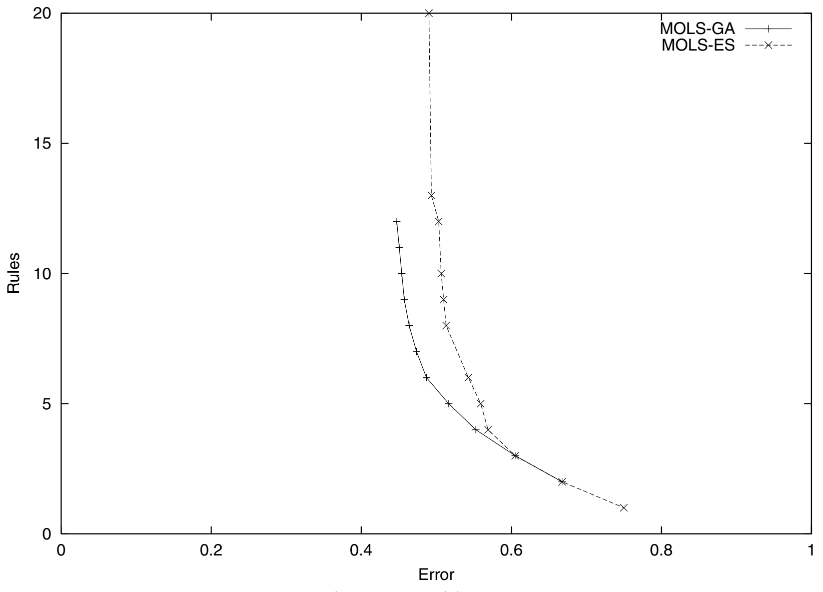
id	MOLS-GA	MOLS-ES	GALE	XCS	C4.5	PART	IB1
`bpa`	76.5±13.4	68.7±6.7	68.4±6.7	65.4±6.9	65.8±6.9	65.8±10.0	64.2±9.1
`bre`	96.0±1.1	96.1±2.2	95.7±2.2	96.7±2.5	95.4±1.6	95.3±2.2	95.9±1.5
`gls`	67.1±9.3	63.4±7.3	65.6±11.9	70.5±8.5	68.5±10.4	69.0±10.0	66.4±10.9
`ion`	91.5±3.6	92.8±2.7	94.0±3.3	89.6±3.1	89.8±0.5	90.6±0.9	90.9±3.8
`irs`	99.3±1.9	95.3±3.1	98.7±2.8	94.7±5.3	95.3±3.2	95.3±3.2	95.3±3.3
`led`	74.9±13.7	74.4±3.4	75.0±0.0	74.5±0.2	74.9±0.2	75.1±0.3	74.3±3.7
`mux`	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0	99.9±0.2	100.0±0.0	99.8±0.2
`prt`	51.2±15.8	40.6±5.7	37.0±8.3	39.8±6.6	41.6±6.4	41.6±6.4	42.5±6.3
`son`	90.8±9.1	71.6±12.5	79.3±6.1	77.5±3.6	71.5±0.5	73.5±2.2	83.6±9.6

Another interesting issue that can be drawn from the results achieved using the selected real-world problems is related to the Pareto front behavior. Figure 9 plots the fronts evolved in two real-world problems (`bre` and `prt`). Looking at the results presented in table 2, it may seem that, for instance, MOLS-GA and MOLS-ES had a similar behavior in the `bre` data set in terms of classification accuracy. If we analyze the evolved Pareto fronts printed in figure 9.a, we realize that they are spreading the population in quite a different way. MOLS-GA achieves its performance through the evolution of bigger hypotheses than the ones obtained by MOLS-ES. These results were achieved using the same amount of iterations. Nevertheless, in other problems (see figure 9.b.) MOLS-ES produce bigger hypotheses than MOLS-GA. Further analysis should be done about this problem dependences.

The Pareto fronts presented in figure 9 also suggest another interesting vision of the analysis of the results. For instance, the front presented in figure 9.b shows an interesting resemblance to the fronts obtained in the noisy `led` problem (see figure 8). This clearly suggests the presence of inconsistencies in the `prt` data set that bounds the MAE. A preliminary inspection of the data set shows that 8.8$\%$ of the instances were replicated, and that only the 83.2$\%$ have different antecedents. However, the experimental MAE equals to .085. In fact in this data set we can identify some extra elements that force the appearance of large fronts. One of these is the large number of classes contained in the `prt` problem. For instance, in the noisy `led` data set the $instances/classes$ ratio (r_{ic}) was r_{ic}=200, whereas in the `prt` problem this ratio drops to r_{ic}=15.4. This fact suggest interesting connections to some results obtained in the *probably approximately correct* models in the *computational learning theory* field [10]. These models compute a theoretical bound to the number of training examples required for successful learning. Therefore, some new interesting questions arise for further research.



(a) `bre` problem



(b) `prt` problem

Fig. 9. Pareto fronts achieved in real-world problems.

6 Conclusions and Further Work

In this paper we have presented a multiobjective optimization approach to evolutionary learning systems. The main motivation was twofold. On one hand we aimed to minimize simultaneously the classification error and the number of rules of a given individual. On the other hand, we were also interested in the relation among generalization and overfitting capabilities of hypotheses. Given this scenario, multiobjective optimization was an elegant solution for achieving the desired goals. In order to validate our approach, we used two different multiobjective learning classifier systems. The first one uses an evolutionary model based on genetic algorithms (MOLS-GA), whereas the second exploits a learning approach based on evolution strategies (MOLS-ES). Both multiobjective learning classifier systems were tested using different data sets. The results obtained show that this multiobjective tradeoff is beneficial to searching for points of appropriate parsimony and accuracy. Moreover, the *bloat* phenomenon is no longer an issue in these systems. Results also show the competence of this approach when compared to previous evolutionary and non-evolutionary learning algorithms.

The multiobjective approach also let us gain some theoretical notions on the effect of noise in the data set. When properly obtained, the Pareto front of the evolved population let us identify overfitting conditions. As we have shown, when the noise in the data set is smaller than the theoretical *minimal achievable error*, the Pareto front shows a *rupture* point. All the trapped solutions are located at left-hand side of the *rupture* point, clearly represented in terms of overfitted hypotheses of the optimal generalization achievable of the classification performance. We also provided a theoretical model to compute this *rupture* point for the `led` data set, given a noise ratio ϵ. Moreover, if the theoretical and the empirical *minimal achievable error* are equal, then the overfitted part of the front disappears.

The work presented in this paper has opened some new directions for further research. The first one is to perform a statistical comparison between the proposed multiobjective learning systems and the other evolutionary and non-evolutionary learning algorithms (see [56]). This comparison should include more data sets, and analyse the impact on accuracy generalization of the *best-accuracy* and *best-compromise* picking strategies. The second one is related to the *minimal achievable error* measure. This measure was computed theoretically only for the `led` problem. The main question that should be investigated is if in a given problem we can compute this measure theoretically, since it is possible to compute it empirically. Unfortunately, it seems that for computing the theoretical MAE model we require extra background knowledge about the problem (for the `led` problem we used the optimal solution $\mathcal{O}$) in addition to the corresponding data set. This background knowledge is not usually available on real-world problems. The results obtained in the `prt` problem also show that in some real-world problems, with few data available and a large set of possible classes, the *minimal achievable error* should include some extra facets like the $instances/classes$ ratio r_{ic}. Finally, some further research should be conducted in order to bring some of the theoretical models obtained in the *computational learning theory* field over the learning classifier systems discipline. The initial goal should be to determine how many instances are required for an ELS in order to obtain high quality general hypotheses of the target concept.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163, and by the Technology Research Center (TRECC), a program of the 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. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. 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 National Science Foundation, or the U.S. Government.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press/ Bradford Books edition (1975)
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc. (1989)
3. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation **3** (1995) 149–175
4. Butz, M.V.: Anticipatory learning classifier systems. Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, Boston, MA (2002)
5. Smith, S.F.: Flexible Learning of Problem Solving Heuristics through Adaptive Search. In: Proceedings of the 8th International Joint Conference on Artificial Intelligence. (1983) 422–425
6. De Jong, K.A., Spears, W.M.: Learning Concept Classification Rules Using Genetic Algorithms. In: Proceedings of the International Joint Conference on Artificial Intelligence, Sidney, Australia (1991) 651–656
7. Janikow, C.: Inductive Learning of Decision Rules in Attribute-Based Examples: a Knowledge-Intensive Genetic Algorithm Approach. PhD thesis, University of North Carolina at Chapel Hill (1991)
8. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag (1992)
9. Llorà, X.: Genetic Based Machine Learning using Fine-grained Parallelism for Data Mining. PhD thesis, Enginyeria i Arquitectura La Salle. Ramon Llull University, Barcelona, Catalonia, European Union (February, 2002)
10. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
11. Koza, J.R.: Genetic Programing: On the Programing of Computers by Means of Natural Selection (Complex Adaptive Systems). MIT Press (1992)
12. Altenberg, L.: Emergent phenomena in genetic programming. Proceedings of the Third Annual Conference on Evolutionary Programming (1994) 233–241
13. Blickle, T., Thiele, L.: Genetic programming and redundancy. Genetic Algorithms within the Framework of Evolutionary Computation: Proceedings of the KI-94 Workshop (1994) 33–38

14. Blickle, T.: Evolving compact solutions in genetic programming: A case study. Parallel Problem Solving from Nature, PPSN IV (1996) 564–573
15. Angeline, P.J.: Subtree crossover causes bloat. Genetic Programming 98 (1998) 745–752
16. Langdon, W.B., Poli, R.: Fitness causes bloat: Mutation. Genetic Programming: First European Conference (1998) 37–48
17. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation **6** (1998) 293–309
18. Langdon, W.B.: Quadratic bloat in genetic programming. Proceedings of the Genetic and Evolutionary Computation Conference 2000 (2000) 451–458
19. Podgorelec, V., Kokol, P.: Fighting program bloat with the fractal complexity measure. Genetic Programming: Third European Conference (2000) 326–337
20. Bleuler, S., Brack, M., Thiele, L., Zitzler, E.: Multiobjective genetic programming: Reducing bloat using SPEA2. In: Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, IEEE Press (2001) 536–543
21. Banzhaf, W., Langdon, W.B.: Some Considerations on the Reason for Bloat. Genetic Programming and Evolvable Hardware **3** (2002) 81–91
22. Soule, T.: Exons and code growth in genetic programming. In Lutton, E., Foster, J.A., Miller, J., Ryan, C., Tettamanzi, A.G.B., eds.: Proceedings of the 4th European Conference on Genetic Programming, EuroGP 2002. Volume 2278 of LNCS., Kinsale, Ireland, Springer-Verlag (2002) 143–152
23. Garrell, J.M., Golobardes, E., Bernadó, E., Llorà, X.: Automatic Diagnosis with Genetic Algorithms and Case-Based Reasoning. AIENG **13** (1999) 367–372
24. Bassett, J.K., De Jong, K.A.: Evolving Behaviors for Cooperating Agents. In: Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems, Springer-Verlag Berlin Heidelberg, LNAI 1932 (2000)
25. Bacardit, J., Garrell, J.M.: Métodos de generalización para sistemas clasificadores de Pittsburgh. In: Primer Congreso Espaol de Algoritmos Evolutivos y Bioinspirados (AEB'02). (2002) 486–493
26. Pareto, V.: Cours d'Economie Politique, volume I and II. F. Rouge, Lausanne (1896)
27. Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary computation and convergence to a pareto front. In Koza, J.R., ed.: Late Breaking Papers at the Genetic Programming 1998 Conference, Madison, WI, Omni Press (1998) 221–228
28. Coello-Coello, C.A.: An updated survey of GA-Based Multiobjective Optimization Techniques. Technical report lania-rd-09-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México (December, 1998)
29. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary Computation **8** (2000) 125–147
30. Tackett, W.A.: Recombination, selection, and the genetic construction of computer programs. Unpublished doctoral dissertation, University of Southern California (1994)
31. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich (1999)
32. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8** (2000) 173–195
33. Zitzler, E.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report 103, Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich (May, 2001)
34. Nordin, P., Banzhaf, W.: Complexity Compression and Evolution. In: Proceedings of the Sixth International Conference on Genetic Algorithms. (1995)

35. Bernadó, E., Mekaouche, A., Garrell, J.M.: 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. (1999) 175–184
36. Gómez-Skarmeta, A.F., Jiménez, F., Ibáez, J.: Pareto-optimality in fuzzy modeling. In: 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98). (1998) 694–700
37. Jiménez, F., Gómez-Skarmeta, A.F., Roubos, H., Robert, B.: Accurate, transparent, and compact fuzzy models for function approximation and dynamic modelling through multi-objective evolutionary optimization. In: First International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag. Lecture Notes in Computer Science No. 1993 (2001) 653–667
38. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
39. Llorà, X., Garrell, J.M.: Automatic Classification and Artificial Life Models. In: Proceedings of Learning00 Workshop, IEEE and Univesidad Carlos III (2000)
40. Traus, I., Bernadó, E.: Sistema Classificador Pittsburgh basat en Estratègies Evolutives. Technical Report TR-ISRG-2002/0001, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, European Union (2002)
41. Llorà, X., Garrell, J.M.: Evolving Partially-Defined Instances with Evolutionary Algorithms. In: Proceedings of the 18th International Conference on Machine Learning (ICML'2001), Morgan Kaufmann Publishers (2001) 337–344
42. Llorà, X., Garrell, J.M.: Knowledge-Independent Data Mining with Fine-Grained Parallel Evolutionary Algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), Morgan Kaufmann Publishers (2001) 461–468
43. Oei, C.K., Goldberg, D.E., Chang, S.J.: Tournament selection, niching, and the preservation of diversity. IlliGAL Report No. 91011, University of Illinois at Urbana-Champaign, Urbana, IL (1991)
44. Bäck, T.: Generalized convergence models for tournament- and (μ, λ)-selection. Proceedings of the Sixth International Conference on Genetic Algorithms (1995) 2–8
45. Miller, B.L., Goldberg, D.E.: Genetic algorithms, tournament selection, and the effects of noise. Complex Systems **9** (1995) 193–212
46. Schwefel, H.P.: Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Technical report, Diplomarbeit, Technische Universität Berlin (1965)
47. Schwefel, H.P.: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. In: Interdisciplinary Systems Research. Volume 26., Birkhäuser. Basel (1977)
48. Bäck, T.: Evolutionary algorithms in theory and practice. Oxford University Press, New York (1996)
49. Wilson, S.W.: Get real! XCS with continuous-valued intpus. In Booker, L., Forrest, S., Mitchell, M., Riolo, R.L., eds.: Festschrift in Honor of John H. Holland, Center for the Study of Complex Systems (1999) 11–121
50. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. submitted to EC (1994)
51. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation **2** (1995) 221–248
52. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology (2000)
53. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms. (1987) 41–49

54. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group (1984)
55. Merz, C.J., Murphy, P.M.: UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998)
56. Bernadó, E., Llorà, X., Garrell, J.M.: XCS and GALE: a Comparative Study of Two Learning Classifier Systems with Six Other Learning Algorithms on Classification Tasks. In: Proceedings of the 4th International Workshop on Learning Classifier Systems (IWLCS-2001), *to appear*, Springer-Verlag (2001)
57. Aha, D., Kibler, D.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
58. Quinlan, R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
59. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
60. Frank, E., Witten, I.H.: Generating Accurate Rule Sets Without Global Optimization. In Shavlik, J., ed.: Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann (1998) 144–151
61. Witten, I.H., Eibe, F.: Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2000)
62. Llorà, X., Goldberg, D.E.: Minimal Achievable Error in the LED problem. IlliGAL Report No. 2002015, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2002)
63. Llorà, X., Goldberg, D.E.: Bounding the effect of noise in Multiobjective Learning Classifier Systems. Evolutionary Computation (2003) In press