12

Multi-objective Learning Classifier Systems

Ester Bernadó-Mansilla[1], Xavier Llorà[2], and Ivan Traus[3]

[1] Department of Computer Engineering, Enginyeria i Arquitectura La Salle,
Universitat Ramon Llull. Quatre Camins, 2. 08022 Barcelona, Spain
esterb@salleurl.edu

[2] Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign,
104 S. Mathews Ave., Urbana, IL 61801, USA
xllora@illigal.ge.uiuc.edu

[3] Conducive Corp., 55 Broad Street, 3rd Floor, New York, NY 10004, USA
itraus@conducivecorp.com

Summary. Learning concept descriptions from data is a complex multiobjective task. The model induced by the learner should be *accurate* so that it can represent precisely the data instances, *complete*, which means it can be generalizable to new instances, and *minimum*, or easily readable. Learning Classifier Systems (LCSs) are a family of learners whose primary search mechanism is a genetic algorithm. Along the intense history of the field, the efforts of the community have been centered on the design of LCSs that solved these goals efficiently, resulting in the proposal of multiple systems. This paper revises the main LCS approaches and focuses on the analysis of the different mechanisms designed to fulfill the learning goals. Some of these mechanisms include implicit multiobjective learning mechanisms, while others use explicit multiobjective evolutionary algorithms. The paper analyses the advantages of using multiobjective evolutionary algorithms, especially in Pittsburgh LCSs, such as controlling the so-called *bloat* effect, and offering the human expert a set of concept description alternatives.

12.1 A Multiobjective Motivation

Classification is a central task in data mining and machine learning applications. It consists of inducing a model that describes the target concept represented in a dataset. The dataset is formed by a set of instances, where each instance is described by a set of features and an associated class. The model describes the relation between the features and the available classes, hence it can be used to explain the hidden structure of the dataset and to classify newly collected instances whose associated class is unknown.

Classification may be regarded as an inherently multiobjective task. Such a task requires inducing a knowledge representation that represents the target concept completely and consistently. In many domains, the induced model

E. Bernadó-Mansilla et al.: *Multi-objective Learning Classifier Systems*, Studies in Computational Intelligence (SCI) **16**, 261–288 (2006)
www.springerlink.com

should also be easily interpretable, which often means a compact representation, and take few computational resources especially in the exploitation stage but also in the training stage. All these objectives are usually opposed and classification schemes try to balance them heuristically.

Multiobjective learning is also present in learning classifier systems (LCSs). LCSs evolve a model of the target concept by the use of genetic algorithms (GAs). Particularly, genetic algorithms search for a hypothesis or a set of hypotheses representing the target concept. The hypotheses can be represented as rule sets, prototype sets, or decision trees [42, 43, 13]. Two main approaches are defined under the LCS framework. The so-called Michigan approach codifies a population of rules, where each individual of the population represents a single rule; thus, a partial solution. The whole population is needed to represent the whole target concept. The Pittsburgh approach codifies a complete knowledge representation (either a ruleset, instance set, or induction tree) in each individual. Thus, the best individual evolved is the solution to the classification problem. In both cases, genetic algorithms are used as search algorithms, seeking to optimize the multiobjective learning goals. In many cases [63, 23, 4], the multiobjective goals are dealt implicitly by the interaction of several components. In other cases [10, 45], the multiobjective goals are directly codified in the fitness function.

The later case benefits from the fact that the search is performed by a genetic algorithm and uses experience gained from the field of multiobjective evolutionary algorithms to get a hypothesis optimizing simultaneously the learning goals. One of the advantages associated with this approach is the improvement of the search in the space of possible hypotheses, minimizing the probabilities of falling into local minima. Moreover, the solution obtained is not only a single hypothesis, but a set of compromise solutions representing different hypotheses along the Pareto front. This offers the human expert the possibility of choosing among a set of alternatives. Besides, this allows to explore the possibilities of combining different hypotheses, which opens the area up to the field of ensemble classifiers [17].

This paper revises the main LCS approaches through the prism of multiobjective learning. First, we give a short history of LCS, summarizing early approaches of LCSs which were significant and inspired nowadays competent LCSs. Then, we focus our attention on the most successful systems of both the Michigan and Pittsburgh approaches. We focus on these types of systems separately, since they have different types of difficulties to achieve the learning goals, due to their different architecture. In Michigan LCSs, learning is centered on the evolution of a minimum set of rules where each rule should be accurate and maximally general. We revise how the XCS classifier system [63] achieves implicitly these goals, and compare it with MOLeCS [10], a multiobjective learning classifier. In Pittsburgh LCSs, learning implies a search through a space of rulesets, evolving an accurate, general, and minimum ruleset. The use of variable-size individuals causes *bloat* [60], i.e., an unnecessary growth of individuals without fitness improvement. Evolutionary multiobjective algorithms are useful to limit this effect while achieving the learning goals. Finally, we discuss open issues on multiobjective learning and directions for further research.

12.2 Learning Classifier Systems: A Short Overview

12.2.1 Cognition and Environment Interaction

The origins of the learning classifiers systems field can be placed within the work by Holland on adaptive systems theory and his early proposals on *schemata processors* [32]. This laid the basis for the first practical implementation of a classifier system which was called CS-1 (Cognitive System Level One) [34]. CS-1 was devised as a cognitive system capable of capturing the events from the environment and reacting to them appropriately. Its architecture was composed of a set of receptors collecting messages into a message list. These messages were matched against a set of rules to decide which actions should be sent to the effectors. When reward was received from the environment, an epochal algorithm distributed payoff into the list of active rules that triggered that action. A genetic algorithm was run to discover new rules by means of crossover and mutation.

The system inspired many approaches which were named under the framework of the Michigan approach. The early Michigan approach can be summarized as having the following properties. The system learns incrementally from the interaction with the environment. It usually starts from a random population of rules which is evaluated with a reinforcement learning scheme by means of a credit assignment algorithm. The *bucket brigade* algorithm [34] was a classical credit assignment algorithm, where the quality (strength) of rules is assigned according to the payoff prediction, i.e., the payoff that the rule would receive from the environment if its action was selected. The GA task is to discover new promising rules, while ensuring the co-evolution of a diverse set of rules. The GA uses the strength calculated by the credit assignment algorithm as the fitness of each rule, thus biasing the search towards highly rewarded rules. The maintenance of different rules along the evolution is addressed by a non-generational scheme and the use of niching techniques [29].

The major problems that arise with traditional Michigan LCSs are the achievement of accurate generalizations and the co-evolution and maintenance of different niches in the population. There is a delicate balance between competition of individuals, which compete to occupy its place in the population, and the co-operation needed to achieve jointly a set of rules. This balance is difficult to maintain for Michigan LCSs. The Boole classifier system [62] tried to balance this equilibrium by the use of fitness sharing and a GA applied to the niches defined by the action sets (sets of rules with similar conditions and the same action). COGIN [31] avoided competition among rules by restricting new classifiers to cover instances not covered by other classifiers of the population.

Despite the known difficulties, Michigan approaches succeeded in many applications. NEWBOOLE [16], which inherits from Boole, was tested in medical datasets and its performance found competitive with respect to CN2 [22] and backpropagation neural networks [57]. Additionally, NEWBOOLE obtained a compact and interpretable ruleset. Further work on NEWBOOLE derived into EpiCS [35], which was adapted to the requirements of epidemiologic databases with mechanisms for risk assessment, and unequal distribution of classes, among others. Results are also competitive when compared with other classifier schemes.

12.2.2 Genetic Algorithm based Concept Learners

Coexisting with the early developments of Michigan LCSs, a parallel line of LCSs was also under research named as the Pittsburgh approach. It emerged with LS-1 classifier system [58], which inspired a main classifier scheme called GABL [23].

The Pittsburgh approach can be characterized as a "classical" genetic algorithm, whose individuals codify a set of rules (in fact, later approaches also introduced other types of codifications). Each individual represents a whole ruleset; thus, a complete solution to the learning problem. The main learning cycle usually operates as in a generational genetic algorithm. The fitness of each individual is evaluated as the performance of its ruleset as a whole, usually under a supervised learning scheme. Fitness usually considers the classification accuracy of the ruleset against the training set of samples.

The first approaches such as GABL considered only classification accuracy on each individual's fitness. Individuals were codified as fixed length rulesets, where each rule was codified as a binary string. Posterior versions introduced incremental learning (GABIL) and the use of two specific operators, *adding alternatives* and *dropping condition*, to bias the search towards general rules and minimal rulesets respectively.

In Pittsburgh LCSs each individual codifies a complete ruleset, so there is no need for cooperation among individuals as in the Michigan approach. This way, the operation of the GA is simpler; the GA converges to a single solution and niching algorithms are not needed. However, since Pittsburgh LCSs search in the space of possible rulesets, the search space is large and usually takes more computational resources than the Michigan approach. Another difficulty is that few control can be applied at the rule level; e.g., rule's generalization, which may encourage the formation minimal rulesets, can hardly be tuned. The two additional GABIL's operators were designed with this purpose. GIL [36] was another proposal that included a large set of operators acting at different levels, whose purpose was also gaining control over the type of rules evolved, but at the expense of an increased parameterization.

The use of variable sized individuals allowed increased flexibility in Pittsburgh LCSs, but caused excessive growth of individuals, with the inclusion of useless rules inside the individuals or the evolution of overspecific rules. Some solutions added parsimony pressures in the fitness function, as in [27] while others addressed these issues from a multiobjective problem perspective [45]. Some recent developments are GALE [39], and GAssist[4]. The later will be analyzed as an example of implicit multiobjective learner. We will also study in detail some multiobjective evolutionary learners, such as MOLS-GA and MOLS-ES [45].

12.2.3 Hybrid Approaches

Due to the known different difficulties of the Michigan and the Pittsburgh approaches, several hybrid systems were proposed such as REGAL and GA-MINER. REGAL [28] uses a distributed architecture, composed of several local GAs evolving separately a single rule covering a partial set of training examples, and a global GA that forms the complete ruleset. The key point of the system relies on the distribution of examples and the combination of the partial solutions.

GA-MINER [25] is a system designed for pattern discovery in large databases. Each individual in the population has a single rule, but its evaluation is done independently from other rules in the population; so it does not need any credit assignment algorithm. The formation of rulesets is performed using an incremental update strategy based on heuristics. The main goal of GA-MINER is to find some interesting patterns in the dataset, instead of fully covering the database as expected in a classification task. Under this framework, the evolution of a complete ruleset is not necessary and in fact, this is not guaranteed by the system.

12.2.4 Accuracy-based Approaches

XCS [63, 64] represents a major development of Michigan style LCSs. Previous Michigan LCSs had identified difficulties to achieve accurate generalizations and maintain a balanced co-evolution of different niches in the population. This led to suboptimal rulesets that could hardly represent the target concept. XCS differs from traditional LCSs on the definition of rule fitness, which is based on the accuracy of the payoff prediction rather than on the prediction itself, and the use of a niche GA. These aspects have resulted in a strong tendency to evolve accurate and maximally general rules, favoring the achievement of knowledge representations that, besides being complete and accurate, tend to be minimal [37].

Other accuracy-based approaches have been studied recently [11]. Particularly, UCS (sUpervised Classifier System) shares many features with XCS, such as the generalization mechanisms. The main difference is that UCS is designed specifically for supervised learning problems. The experiments showed that UCS is more suitable to classification problems with large spaces, specially those with high number of classes or with highly unequal distribution of examples.

We will study XCS in more detail and compare it with MOLeCS, a multiobjective Michigan LCS.

12.3 Multiobjective Optimization

Prior to the study of the different multiobjective mechanisms of the Michigan and Pittsburgh LCSs, we briefly describe the notation we will use to refer to multiobjective issues.

In a multiobjective optimization problem (MOP) [61] a solution $\mathbf{x} \in \Omega$ is represented as a vector of n decision variables $\mathbf{x} = (x_1, \ldots, x_n)$, where Ω is the decision variable space. We want to optimize k objectives which are defined as $f_i(\mathbf{x})$, with $i = 1 \ldots k$. These objectives are grouped in a vector function denoted as $F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, where $F(\mathbf{x}) \in \Lambda$. F is a function which maps points from the decision variable space Ω to the objective function space Λ:

$$\begin{aligned} F : \Omega &\longmapsto \Lambda \\ \mathbf{x} &\longmapsto \mathbf{y} = F(\mathbf{x}) \end{aligned} \tag{12.1}$$

Without loss of generality, we can define a MOP as the problem of minimizing a set of objectives $F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, subject to some constraints $g_i(\mathbf{x}) \leq 0$, $i = 1, \ldots, m$. These constraints are necessary for problems where there are invalid solutions in Ω.

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 [53] introduced the concept of dominance and Pareto optimum to deal with this issue.

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

$$\mathbf{u} \preceq \mathbf{v} \iff \forall i \in 1, \ldots, k\, ,\, u_i \leq v_i \wedge \exists i \in 1, \ldots, k : u_i < v_i \tag{12.2}$$

The concept of *Pareto optimality* is based on the dominance definition. Thus, a solution $\mathbf{x} \in \Omega$ is Pareto optimal if there is not any other solution $\mathbf{x}' \in \Omega$ whose objective vector $\mathbf{u}' = F(\mathbf{x}')$ dominates $\mathbf{u} = F(\mathbf{x})$. 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 $\mathcal{P}^*$:

$$\mathcal{P}^* := \{\mathbf{x}_1 \mid \nexists\, \mathbf{x}_2 : \mathbf{F}(\mathbf{x}_2) \preceq \mathbf{F}(\mathbf{x}_1)\} \tag{12.3}$$

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

$$\mathcal{PF}^* := \{\mathbf{u} = \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \mid \mathbf{x} \in \mathcal{P}^*\} \tag{12.4}$$

12.4 Multiobjective Learning in Michigan Style LCSs

Michigan style LCSs codify each individual as having a single rule. Thus, learning in Michigan LCSs implies codifying the learning goals at the rule level. The approach is to maximize accuracy and generality in each rule, which consequently could combine into a consistent, complete and minimal ruleset. This section revises two particular Michigan LCSs, representative of two different approaches. The first one, XCS, uses an accuracy-based fitness, while generalization is achieved mainly by a niche GA applied in a frequency basis. The later, MOLeCS, defines a multiobjective fitness which directly guides the search process to optimize these goals.

12.4.1 XCS: Implicit Multiobjective Learning

Description of XCS

XCS [63, 64] represents the target concept in a set of rules. This ruleset is incrementally evaluated by means of interacting with the environment, through a *reinforcement learning scheme*, and is improved by a search mechanism based on a *genetic algorithm*.

Each rule's quality is estimated by a set of parameters. The main parameters are: a) *prediction*, an estimate of the reward that the rule would receive from the environment, b) *accuracy*, the accuracy of the prediction and c) *fitness*, which is based only on accuracy.

The basic training cycle performs as follows. At each time step, an input x is presented to the system. Given x, the system builds a match set [M], which is formed by all the classifiers in the population whose conditions are satisfied by the input example. If no classifiers match, then the covering operator is triggered. It creates new classifiers matching the current input.

During training, XCS explores the consequences of all classes for each input. Therefore, given a match set [M], XCS selects randomly one of the classes proposed in [M] and sends it to the environment. All classifiers proposing that class are classified as belonging to the action set [A]. The environment returns a reward which is used to update the parameters of the classifiers in [A], and then the GA may be triggered. In test mode, XCS proposes the best class from those advocated in [M], and the update and search mechanisms are disabled.

If we run XCS in training, the GA is triggered when the average time since the last occurrence of the GA in the action set exceeds a threshold θ_{GA}. The

GA takes place in the action set, rather than in the whole population. It selects two parents from the current [A] with probability proportional to fitness, and applies crossover and mutation. The resulting offspring are introduced into the population. If the population is full, a classifier is selected for deletion. The deletion probability is proportional to the size of the action sets where the classifier has participated. If the classifier is experienced and poorly fit, its deletion probability is increased by an inverse proportion of its fitness.

XCS also uses subsumption as an additional mechanism to favor the generalization of rules. Whenever a classifier is created by the GA, it is checked for subsumption with its parents before being inserted into the population. If one of the classifier's parents is sufficiently experienced, accurate, and more general than the classifier, then the classifier is discarded, and the parent's numerosity is increased. This process is called *GA subsumption.*

Implicit Multiobjective Learning

XCS receives inputs from the instances available in the training set and receives feedback of its classifications in the form of rewards. The environment is designed to give a maximum reward if the system predicts the correct classification and a minimum reward (usually zero) otherwise. XCS's goal is to maximize the rewards received from the environment and in doing so it tries to get a *complete, consistent and minimal representation* of the target concept. We analyze the role of each component in achieving this compound goal.

The role of covering is to enforce a complete coverage of the input space. Whenever an input is not covered by any classifier, covering creates an initial classifier from where XCS launches its learning and search mechanisms in that part of the space.

The task of the reinforcement component is to evaluate the classifier's parameters from the reward received from the environment. This is done incrementally. For each example, it computes the prediction error of the classifier, gets a measure of accuracy, and then updates fitness based on this computed accuracy.

The search component is based on a genetic algorithm. Basing fitness on accuracy makes the genetic algorithm to search for accurate rules. Generalization is stressed by the application of the GA to the niches defined by the action sets. This is explained by Wilson's generalization hypothesis [63]: given two accurate classifiers with one of them matching a subset of the input states matched by the other, then the more general one will win because it participates in more action sets and thus has more reproductive opportunities. As a consequence, the more general classifier will tend to displace the specific classifier, resulting in compact representations. Subsumption is included to encourage generalization and compactness as well.

The niche GA is also designed to favor the maintenance of a diverse set of rules which jointly represent the target concept. This is achieved through different mechanisms: a) the GA's triggering mechanism, which tries to balance the application of the GA among all the action sets, b) selection, which is applied locally to the action sets, c) crossover, performing a kind of restricted mating, and d) the deletion algorithm, which tends to delete resources from the most numerous action sets. In XCS, the niches are defined by the action sets, which are formed by a set of classifiers matching a common set of input states and a common class.

12.4.2 MOLeCS: Explicit Multiobjective Learning

From Multiobjective Rulesets to Multiobjective Rules

MOLeCS (MultiObjective Learning Classifier System) [10] is a Michigan style LCS that evolves a ruleset describing the target concept by the use of multiobjective evolutionary algorithms.

As a Michigan-style LCS, MOLeCS evolves a population of rules which jointly represent the target concept. Thus, each individual is a rule and the whole ruleset is formed by all individuals in the population. The system's goal is to achieve a complete, consistent and compact ruleset by means of multiobjective evolutionary algorithms. However, these goals cannot be directly defined into the GA search, since the GA search is performed at the rule level, not at the ruleset level. Therefore, these goals are adapted to the mechanisms of a Michigan style LCS by defining two objectives at the rule level: generalization and accuracy. Each rule should maximize simultaneously generalization and accuracy. If fitness was only based on accuracy, the GA search would be biased towards accurate but too specific rules. This would result in an enhancement of the solution set (i.e., need of more rules) and also poor coverage of the feature space. On the contrary, basing fitness on generality would result in low performance, in terms of classification accuracy. The solution was to balance these two characteristics at the same time. The hypothesis was that by guiding the search towards general and accurate rules would result in minimum, complete and accurate set of rules.

Description of MOLeCS

Each individual in MOLeCS codifies a rule (classifier) of type: *condition* → *class*. Each rule r_i is evaluated against the available instances in the training dataset, and two measures are computed:

$$generalization(r_i) = \frac{\#\text{ covered examples }(r_i)}{\#\text{ examples in the training set}} \tag{12.5}$$

$$accuracy(r_i) = \frac{\#\text{ correctly classified examples }(r_i)}{\#\text{ covered examples }(r_i)} \tag{12.6}$$

MOLeCS defines the multiobjective problem as the simultaneous maximization of accuracy and generalization. Thus, the fitness of each rule is assigned according to a multiobjective evaluation strategy. Several MOEA techniques were explored in MOLeCS, being a method based on lexicographic ordering the most successful one. Particularly, the approach taken was that of sorting the population according to the accuracy objective and in case of rules equally accurate, sort them by the generalization objective. The interpretation was that of searching for accurate rules being as general as possible. Once sorted, fitness was assigned according to the ranking. Other strategies based on Pareto dominance and the aggregating approach were also considered but found less successful.

Once the fitness assignment phase is performed, the GA proceeds to the selection and recombination stages. In each iteration, G individuals are selected with stochastic universal sampling (SUS) [7]. Then, they undergo crossover with probability p_c and mutation with probability p_m per allele. The resulting offspring are introduced into the parental population. The replacement method was designed to achieve a complete coverage of the feature space.

In fact, it was argued that promoting general classifiers was not sufficient to reach a complete ruleset. The genetic algorithm could tend, due to the genetic drift [30], to a single general and accurate classifier and usually one classifier does not suffice to represent the whole target concept. Therefore, the system should enforce the *co-evolution* of a set of diverse fit rules by niching mechanisms. Niching in MOLeCS is performed in the replacement stage. Particularly, deterministic crowding is applied, where the child replaces the most similar parent only if it has greater fitness.

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 class. As explained before, in case of equally fit rules, the most accurate rule is chosen.

12.4.3 Results

XCS's generalization hypothesis [64] explains that the accuracy-based fitness coupled with the niche GA favor the evolution of compact rulesets consisting of the most general accurate rules. Additionally, the optimally hypothesis [37] argues that XCS tends to evolve minimum rulesets. These hypotheses are supported by theoretical studies on the pressures induced by the interaction of XCS's components [21, 19]. XCS has also demonstrated to be highly competitive with other machine learning schemes such as induction trees, and nearest neighbors, among others, in real world classification problems [65, 13]. Recent studies are investigating the domain of competence of XCS in real world domains, i.e., to what kind of problems XCS is suited and poorly suited to [12].

MOLeCS was tested in artificial classification problems, of type of multiplexer and parity problems, often used in the LCS community, as well as in real world datasets. The multiobjective evolutionary algorithms let the system evolve accurate and maximally general rules which together formed compact representations. A comparison with a single-objective approach maximizing only each rule's accuracy demonstrated that the multiobjective optimization was necessary to overcome the tendency of evolving too many specific rules which would lead the system towards suboptimal solutions partly representing the target concept. Optimizing simultaneously generality and accuracy was a better approach than a single optimization approach. However, giving the same importance to these objectives with techniques such as Pareto ranking caused the system evolve overgeneral rules, preventing other maximally general rules from being explored and maintained in the population. This consequently resulted in poor accurate rulesets. The best approach taken was that of introducing the decision preferences a priori, i.e., during the search process, so that the algorithm could find accurate rules as general as possible. This led to the evolution of nearly optimal rulesets, with results highly competitive with respect to other LCSs such as XCS in real world datasets.

The main difficulty of MOLeCS was identified within the niching mechanisms. MOLeCS presented difficulties to stabilize a niche population and obtain a diverse set of rules which jointly represented a complete ruleset. Niching mechanisms such as deterministic crowding were only useful for limited datasets which could be easily represented by small rulesets. Switching towards implicit niching mechanisms such as those of XCS, would include an extra generalization pressure, as it is explained by Wilson's generalization hypothesis. Having both an explicit pressure towards generalization by means of a multiobjective approach and an implicit generalization pressure produced by the niche GA could break the delicate equilibrium of evolutionary pressures inside LCSs and lead to overgeneral rules. However, this remains an unexplored area that would benefit from further research.

12.5 Multiobjective Learning in Pittsburgh Style LCSs

The previous section presented how Michigan classifier systems evolve a population of rules that classify a particular. However, the Pittsburgh style classifier systems evolve a population of individuals, each of them a variable-length ruleset that represents a complete solution to the problem. Such an approach greatly simplifies the evolutionary process. For instance, the fitness of each candidate individual can be computed using the accuracy of the ruleset classifying the dataset. Then, the evaluated individuals undergo the traditional selection and recombination mechanisms.

Such simplicity, however, comes with a price to pay, the tradeoff among accuracy and generalization. As in Michigan approaches, Pittsburgh classifier systems should provide compact and accurate individuals. The rest of this section revises the implications of such a goal on Pittsburgh classifier systems and presents some systems and results that exploit multiobjective optimization ideas to achieve it.

12.5.1 A Generalization Race?

Evolution of variable size individuals causes *bloat* [60]. The term, named within the field of genetic programming (GP), refers to the code growth of individuals without any fitness improvement.

Langdon [38] attributes bloat to two possible reasons, labeled as "fitness causes bloat" and "natural code is protective". The former refers to the fact that selection does not distinguish between individuals with the same fitness but different size. Thus, for each individual with a given fitness, there is a large set (possibly infinite) of individuals with the same fitness and larger codes. A search guided by fitness becomes a random walk among individuals with different sizes and equivalent fitness. The second cause of bloat considers that neutral code that does not influence fitness tends to be protective, i.e., reduces the probability that the genetic operators disrupt useful code. Thus, they do not improve fitness but their maintenance in the population is favored by the genetic operators.

In Pittsburgh LCSs, bloat may arise in two different forms: (1) the addition of useless rules, or (2) the evolution of over-specific rules. Without limitation on the number of rules in the ruleset, the search space becomes too large, and solutions are far from being optimal. Contributions to address this issue have been worked out from both the GP field and the LCSs field.

Some of the approaches taken in the GP field consist of imposing a *parsimony pressure* toward compact individuals (see for example, [59]) by varying fitness or through specially tailored operators. Recently, an approach proposed by Bleuler, Brack, Thiele, and Zitzler [15] uses a multiobjective evolutionary algorithm to optimize two objectives: maximize fitness and minimize size. In their approach they use the SPEA2 algorithm [68, 70, 69].

In Pittsburgh classifier systems, some approaches also used a parsimony pressure, while others directly codified a multiobjective evolutionary approach. The next sections detail them.

12.5.2 Implicit Multiobjective Learning

Parsimony Pressure

The classical approach to addressing bloat and multiobjective learning goals was to introduce a parsimony pressure in the fitness function, in such a way that the fitness of larger individuals was decreased [9, 3, 27]. For example, in [27] 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 threshold value appropriately.

Bacardit and Garrell [3] define a similar fitness function, as well as a set of operators for the deletion of introns[4] [49] (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 rulesets tend to have more generalization capabilities [14, 3, 49].

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 [49], or even in a total population failure (population collapses with individuals of minimal size). Soule and Foster [59] showed that the effect of the 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. The first step toward introducing multiobjective pressures in Pittsburgh classifier systems is to use a linear combination of the accuracy of a given individual and its generality –understood as inversely proportional to its size. The next section revises an approach of this type, which is based on the *minimum description length* principle [5].

GAssist

GAssist [5] is a Pittsburgh LCS descendant of *GABIL*. The system applies a near-standard generational *GA* that evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable–length ruleset. Tournament selection is used.

A special fitness function based on the Minimum Description Length (*MDL*) principle [56] is used, balancing the accuracy and the complexity of an individual. The *MDL* formulation used is defined as follows:

$$MDL = W \cdot \mathrm{TL} + \mathrm{EL} \tag{12.7}$$

where TL is for the complexity of the ruleset, which considers the number of rules and the number of relevant attributes in each rule, and EL is a measure of the error of the individual on the training set. W is a constant that adjusts the relation between TL and EL. An adaptive mechanism is used to avoid domain-specific tuning of the constant.

The system also uses a rule deletion operator to further control the bloat effect. The algorithm deletes the rules that do not have been activated by any example, the so-called introns. The authors argue that the growth of

[4] Non-coding segments. In GP literature this concept has also been termed *non-effective code* [8].

introns in the individuals is protective, in the sense that they prevent the crossover operator from disrupting the useful parts of the individual. However, as the code growths, the chances of improving the fitness of the individual by recombination also decrease. Hence, removing part of this useless code through an appropriate balance is beneficial to the search process.

A comparison of GAssist with XCS [2] showed similar performance in terms of accuracy rate. However, the number of rules evolved by GAssist was much smaller that the number of rules of XCS. The comparison of GAssist with XCS also arose a certain difficulty of GAssist in handling multiple classes and huge search spaces. XCS was found to overfit in some datasets, where the generalization pressure due to the niche reproduction hardly applied because of data sparsity. Section 12.5.4 gives more details by means of a comparison of these systems in a selected set of classification problems.

12.5.3 Explicit Multiobjective Learning

By using a multiobjective evolutionary approach, we can explicitly search for a set of rules that minimizes the misclassification error 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 dataset 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 ruleset" to the final user (decision maker), or combine them all using some *bagging* technique [17, 41, 39].

Classification and Multiobjective Optimization

Multiobjective Evolutionary Algorithms can be applied to trade off between two objectives: the accuracy of an individual, and its size.

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

$$minimize\ F(\mathbf{x}) = (f_e(\mathbf{x}), f_s(\mathbf{x})) \tag{12.8}$$

$$f_e(\mathbf{x}) = \frac{miss(\mathbf{x}, \mathcal{D})}{|\mathcal{D}|} \tag{12.9}$$

$$f_s(\mathbf{x}) = size(\mathbf{x}) \tag{12.10}$$

The misclassification error corresponds to the number of instances incorrectly classified divided by the size of the training set. Searching for rulesets minimizing the misclassification error means to search for rulesets covering correctly as many instances as possible. Minimizing the number of rules of the ruleset seeks to search for general and compact rulesets, which moreover avoid the bloat effect. In fact, this is a simple multiobjective approach. Other types of proposals could include more objectives such as measures of generalization of the rulesets, or coverage (number of instances covered by a ruleset).

MOLS-GA

Some approaches have addressed learning rulesets in Pittsburgh LCS architectures from a multiobjective perspective. MOLS-GA and MOLS-ES [45] are two examples. They represent two similar approaches to the multiobjective problem, being different on whether they base the search mechanism on a GA (MOLS-GA) or an evolution stratgey (MOLS-ES). Since they do not offer significant differences on the multiobjective approach itself, we will center our study on MOLS-GA for being more representative of Pittsburgh LCSs.

MOLS-GA codifies a population of individuals, where each individual represents a complete representation of the target concept. The available knowledge representations are rulesets or instance sets [39, 42, 43]. If the problem's attributes are nominal, MOLS-GA uses rulesets, represented by the ternary alphabet (`0`, `1`, `#`) often used in other LCSs [33, 29, 63]. 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.

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 12.1 shows an example of the different equivalence classes that appear in a population at a given iteration. This plot is obtained with the multiplexer (`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.

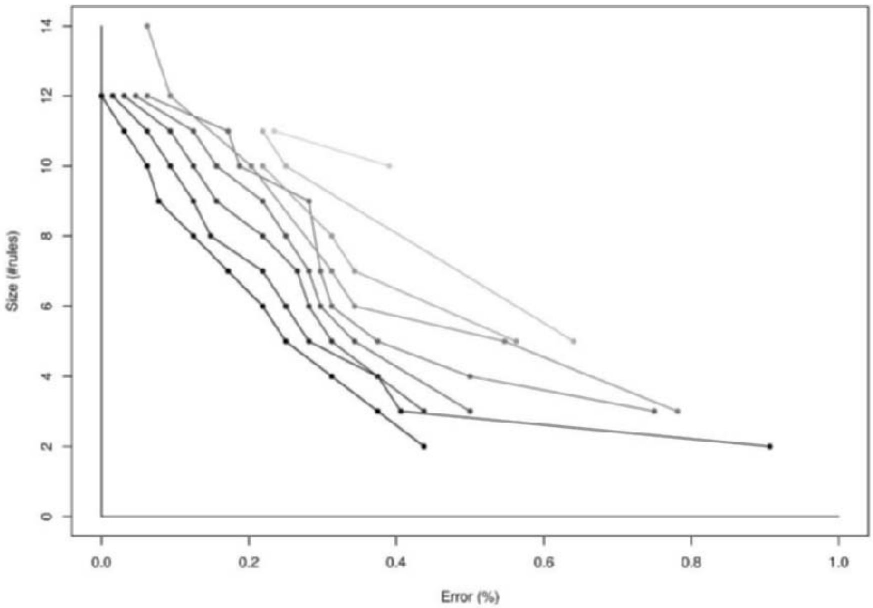


Fig. 12.1. Sorted population fronts at a given iteration of MOLS in the `mux` problem [45].

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:

$$fitness(\mathcal{I}^i) > fitness(\mathcal{I}^{i+1}) \tag{12.11}$$

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:

$$fitness(\mathcal{I}^i_j) = \frac{(n-i)\delta}{\sum_{k\in\mathcal{I}} \phi(d_{\mathcal{I}^i_j \mathcal{I}_k})} \tag{12.12}$$

where $\phi(d_{\mathcal{I}^i_j \mathcal{I}_k})$ is the sharing function [30]. The *sharing* function is computed using the Euclidean distance between the multiobjective vectors. The radius of the *sharing* function σ_{sh} was set to $\sigma_{sh} = 0.1$.

After individuals are evaluated, selection is applied using a tournament selection algorithm [50, 6] with elitism. Elitism is often applied in evolutionary multiobjective optimization algorithms and it usually consists of keeping the solutions of the Pareto front evolved in each generation [61]. 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 solutions with the lowest error, which are important to drive the evolution toward accurate solutions.

After selection, crossover and mutation are applied. Crossover is adapted from two-point crossover to individuals of variable size; cut points can occur anywhere inside the ruleset, but they should be equivalent in both parents so that valid offspring can be obtained. The mutation consists in generating a random new gene value.

What is the Purpose of the Pareto Front?

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

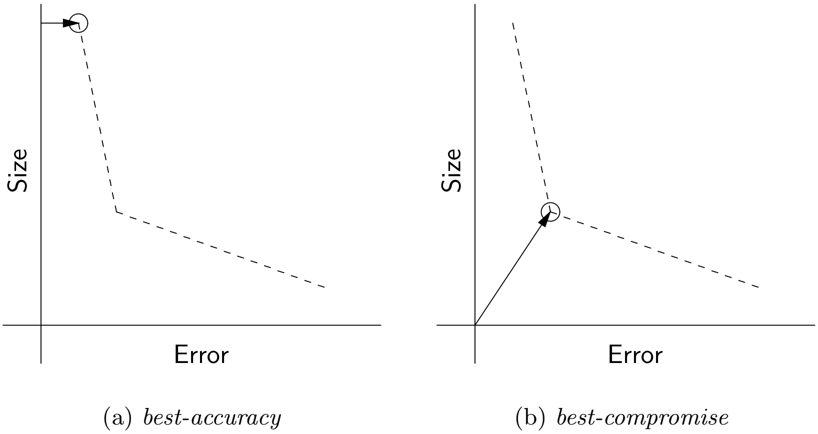
The *decision maker* has several hypotheses among which to choose, all provided by the classification front. The *decision maker* can be a human or an expert system with some knowledge about the problem. A typical approach is to select a solution from the evolved Pareto front. Two strategies may be considered, as shown in figure 12.2. The first one (*best-accuracy*) chooses the solution $\mathbf{x}$ of the front with the best accuracy, that is, the one that minimizes $f_e(\mathbf{x})$. On the other hand, the second one (*best-compromise*) picks the hypothesis of the front that minimizes the objective vector $\mathbf{u} = F(\mathbf{x})$. Thus, the selected solution is the one that balances equally both objectives. In other words, the solution $\mathbf{x}$ that minimizes $|F(\mathbf{x})| = \sqrt{f_e(\mathbf{x})^2 + f_s(\mathbf{x})^2}$.

We could instead benefit from the combination of the multiple hypotheses obtained in the evolved Pareto front. Such a technique is inspired in the *bagging* technique [17, 41], which consists of training several classifiers and combining them all into an ensemble classifier. The bagging technique tends to reduce the deviation among runs, and the combined solution often improves the generalization capability of the individual solutions [39]. An adaptation of such an strategy to the combination of solutions from the Pareto front could give similar benefits as shown elsewhere [40].

12.5.4 Experiments and Results

Through a Pareto Front Glass

Figure 12.1 shows an example of how the sorting of a population of candidate solutions produces a clear visualization of the tradeoff between accuracy and generality.



(a) *best-accuracy* (b) *best-compromise*

Fig. 12.2. Decision maker strategies using (a) the solution with the lowest error (*best accuracy*) and (b) the solution closer to the origin (*best compromise*).

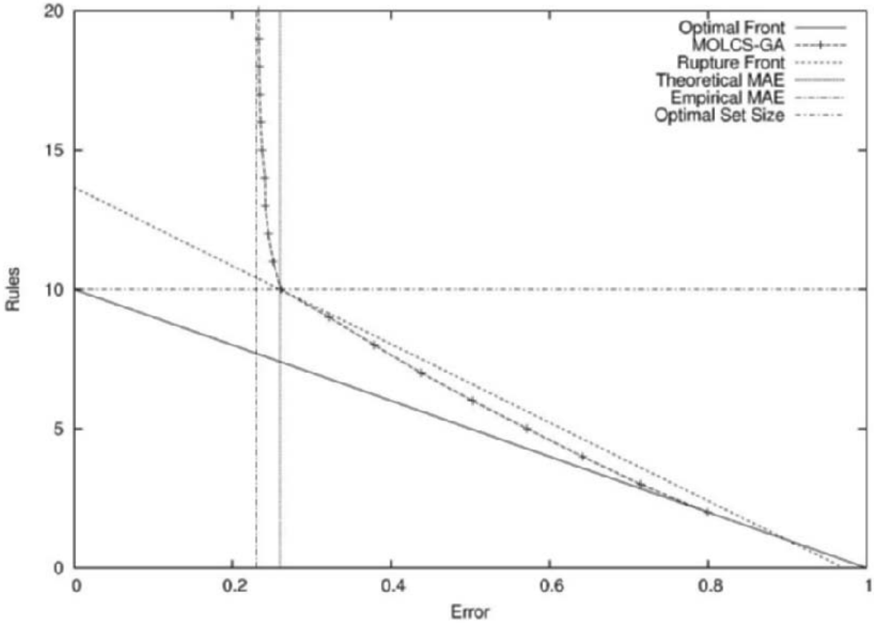


Fig. 12.3. Evolved Pareto front by MOLS-GA in the `LED` problem [44].

Such Pareto fronts arise an interesting property, as shown by Llorà and Goldberg [44]. In the presence of noise in the dataset, the Pareto front presents a clear rupture around the maximally general and accurate solutions. Such rupture point is achieved around the minimal achievable error (MAE) [44] for the dataset at hand. 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 or a solution too specific to the current training dataset. If any of these solutions is tested using new instances not previously shown in the training phase, they would produce 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.

Results

The work by Llorà, Goldberg, Traus, & Bernadó-Mansilla [45] evaluated the performance of different multiobjective LCSs on nine different datasets, which are described in Table 12.1. Two of them are *artificial datasets*. `Mux` is the eleven input multiplexer, widely used by the LCS community [63]. `Led` is the seven-segments problem [18], obtained from the UCI repository [47]. The remaining datasets also belong to the UCI repository: *bupa liver disorders* (`bpa`), *Wisconsin breast cancer* (`bre`), *glass* (`gls`), *ionosphere* (`ion`), *iris* (`irs`), *primary tumor* (`prt`), and *sonar* (`son`). These datasets contain categorical and numeric attributes, as well as binary and n-ary classification tasks.

We summarize some of the results published in [45], where two explicit multiobjective Pittsburgh approaches are compared with other schemes. Particularly, we compare MOLS-GA and MOLS-ES with GAssist and XCS. We also include a comparison with non-evolutionary schemes: IB1 [1], C4.5 [54, 55], and PART [26]. Their algorithms were obtained from the *Weka* package [67] and ran with their default configuration.

The results were obtained from *stratified ten-fold cross-validations runs* [48, 67] using the different learning algorithms on the selected datasets. MOLS-GA used a *best-accuracy* strategy for the test phase, whereas MOLS-ES used *best-compromise*. Table 12.2 shows the classification accuracy obtained by each method. Observe that LCSs in general are highly competitive with the non-evolutionary schemes. Moreover, MOLS-GA and MOLS-ES offer good classification accuracies when compared with GAssist and XCS. There is not a clear winner in all the datasets. Instead, all the methods seem to perform equivalently.

Table 12.3 shows the average and standard deviation of the number of rules obtained by the LCS approaches. Observe that the three Pittsburgh-based methods offer fairly simple rulesets, composed of 3 to 15 rules approximately. On the contrary, XCS gets much larger rulesets. In fact, the ruleset evolved by Pittsburgh LCSs is that obtained by the best individual of the population, while the ruleset obtained by XCS considers all the population. If we consider that GAssist evolves a population of 400 individuals, the number of explored rules is equivalent to that evolved by XCS. Moreover, the ruleset of a Pittsburgh LCSs operates as an activation list; i.e., the rules form a hierarchy going from the most general to the most specific ones. The first rules tend to cover many examples (the general case), while the last rules codify the exceptions to the general rule. XCS does not use such a hierarchy so that the number of necessary rules is higher. Despite these issues, the final XCS's ruleset should be still further processed. Most of the rules are only product of the recent exploration of the algorithm and are not relevant to performance. Other rules overlap partially with others; therefore, they could be compacted. Research on reduction algorithms and further processing of the evolved rules is being conducted by several authors to get a ruleset easily interpretable to human experts [66, 24, 51]. In [51] some reduction algorithms reported a reduction of 97% without a significant degradation in classification accuracy. For example, the `bpa` ruleset could be reduced to 90 rules, which is still higher that Pittsburgh LCSs, but more reasonable to human experts.

MOLS-GA is in average the method which gets smaller rulesets for similar values of classification accuracy. This probably means that MOLS-GA evolves better Pareto fronts than MOLS-ES. Figure 12.4 shows two examples for the `bre` and the `prt` datasets. See that the first one belongs to the case where MOLS-ES gets better performance, which corresponds to a better evolved Pareto front. The second plot corresponds to the more general case where MOLS-GA obtains a better front, resulting in higher performance. MOLS-GA also gets smaller rulesets than GAssist in average, which means that the explicit multiobjective approach is effective to further reduce the size of the ruleset.

Table 12.1. Summary of the datasets 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

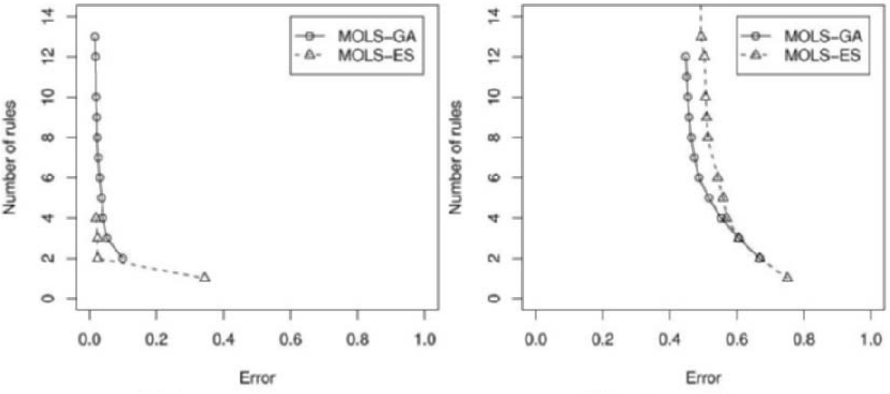
The Pareto fronts presented in figure 12.4 also suggest another interesting analysis of the results. For instance, the front presented in figure 4(b) shows an interesting resemblance to the fronts obtained in the noisy `led` problem

Table 12.2. Comparison of classification accuracy on selected datasets. The table shows the mean and standard deviation of each method on a *stratified ten-fold cross-validation* experiment [45, 2].

id	MOLS-GA	MOLS-ES	GAssist	XCS	C4.5	PART	IB1
bpa	76.5±13.4	68.7±6.7	61.5±8.3	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	95.9±2.5	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	68.2±9.3	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	92.4±5.0	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	95.3±5.7	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	n.a.	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	47.8±8.1	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	77.2±8.9	77.5±3.6	71.5±0.5	73.5±2.2	83.6±9.6

Table 12.3. Comparison of ruleset sizes obtained by different methods on selected datasets. The table shows the mean and standard deviation of each method on a *stratified ten-fold cross-validation* experiment [45, 2].

id	MOLS-GA	MOLS-ES	GAsssist	XCS
bpa	10.1±0.3	15.1±4.0	7.2±1.2	2377.5±125.7
bre	9.4±1.5	8.5±1.6	11.7±2.6	2369.5±114,5
gls	4.2±0.4	12.2±2.7	8.8±1.4	2709±98.9
ion	6.9±0.3	12.8±3.8	2.1±0.3	4360±134.9
irs	2.7±0.1	5.8±1.1	4.3±0.9	730±75.8
led	14.9±0.8	14.7±2.0	n.a.	102.5±10.2
mux	10.0±0.3	12.7±1.9	11.2±0.5	640±45.5
prt	9.3±0.5	12.0±5.0	16.0±3.8	1784±45.8
son	11.6±0.5	13.1±4.0	8.3±1.4	4745±123.2



(a) bre problem (b) prt problem

Fig. 12.4. Pareto fronts achieved in real problems by MOLS-GA and MOLS-ES [45].

(see figure 12.3). This clearly points out to the presence of inconsistencies in the `prt` dataset that bounds the MAE. However, further work on the usage of the MAE measure is still needed, as well as exploring its connections to *probably approximately correct* models in the *computational learning theory* field [48].

12.6 Current Research Trends and Further Directions

Learning classifier systems address the complex multiobjective task either balancing the learning goals implicitly or by means of evolutionary multiobjective algorithms adapted to the particular architectures.

Within the field of Michigan LCSs, the best approach, XCS, does not use explicit multiobjective evolutionary algorithms. The combination of accuracy-based fitness and the niche GA, with the addition of other heuristic components, achieves the appropriate balance to favor the creation and maintenance of maximally general rules. In fact, the equilibrium is delicate and studies on pressures point out how to balance this equilibrium through appropriate parameter settings. Despite this difficult balance, it does not seem feasible to add explicit multiobjective algorithms to further emphasize the generalization of rules, because this probably would favor overgeneral rules. However, this still remains an unexplored research area which may be analyzed to strength generalization in cases where this is scarcely favored by the niche GA, such as in problems with unequal distribution of examples per class (see for example [52]). Explicit multiobjective algorithms are also able to achieve accurate and maximally general rules in other types of Michigan LCSs. The main difficulty is found on the architecture rather than on the multiobjective evaluation itself. The main problem is to stabilize different niches co-operating to form a complete ruleset. More research on niching algorithms is needed, combined with restricted mating methodologies.

A problem common in Michigan LCSs is the difficulty to control the number of rules forming the final ruleset. For binary attributes, favoring generalization of rules is enough to achieve a minimum ruleset. But for continuous attributes and real-valued representations, Michigan LCSs tend to produce high number of rules that overlap partially. Pittsburgh LCSs using some kind of pressure on the number of maximum rules produce rulesets much reduced than those of Michigan LCSs. Although some reduction algorithms have been designed in the Michigan framework to get compact rulesets [20], they still provide larger rulesets. This imposes a limitation to their interpretability by human experts. Stressing generalization and favoring removal of useless rules during training could help evolving smaller rulesets. This would also reduce the computational cost, since the system would maintain less rules during training.

Within the Pittsburgh approach, implicit and explicit multiobjective approaches are able to balance accuracy and generalization. Moreover, the explicit usage of multiobjective techniques provides an output with a clear tradeoff among both objectives. A Pareto front provides a way of sorting the final population of candidate solutions. Such a front may be explored for a particular solution, or just combined to form an ensemble of classifiers, as it is currently being analyzed in [40]. Such research also needs to face the readability of the final solution. Measures of rule interpretability must be defined. Once the proper measure is ready, it can simply be integrated in the evolutionary process as a third objective to be optimized. Many domains, such as medical domains, specify the cost of misclassifying each of the classes, instead of using a single measure of error. Multiobjective algorithms can also be appropriate to balance the different types of misclassification errors and provide a set of alternatives. However, we should analyze whether the addition of more objectives involves a harder search, and what kind of solutions can be evolved.

Another key research area for Pittsburgh classifier systems is the identification of overfitting situations from the evolved Pareto fronts, as pointed out in the chapter. However, such approach assumes that the global pressure toward accuracy and generality will lead the rulesets towards the desired learning goals. This is a clearly top-down approach. Recently, the work of Wilson in 1995 [63] and the major shift in the way in which fitness was computed in the Michigan approach have been revisited by researchers. Accuracy has become a central element in the process of computing the fitness of rules (or classifiers). Initial attempts to apply Wilson's ideas to Pittsburgh-style classifier systems are on their way [46]. When such ideas are combined with estimation of distribution algorithms, a bottom-up approach to Pittsburgh classifiers would start to emerge—as the compact classifier system (CCS) has preliminary shown.

Acknowledgments

The first author acknowledges the support of Enginyeria i Arquitectura La Salle, Ramon Llull University, as well as the support of Ministerio de Ciencia y Tecnología under Grant TIC2002-04036-C05-03 and Generalitat de Catalunya under Grant 2002SGR-00155.

This work was sponsored in part by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

References

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] J. Bacardit and M. V. Butz. Data Mining in Learning Classifier Systems: Comparing XCS with GAssist. In *Seventh International Workshop on Learning Classifier Systems (IWLCS-2004)*. LNAI, Springer (in press), 2004.

[3] J. Bacardit and J. M. Garrell. Métodos de generalización para sistemas clasificadores de Pittsburgh. In *Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, pages 486–493, 2002.

[4] J. Bacardit and J. M. Garrell. Bloat Control and Generalization Pressure using the Minimum Description Length Principle for a Pittsburgh Approach Learning Classifier System. In *Proceedings of the 6th International Workshop on Learning Classifier Systems*. LNAI, Springer (in press), 2003.

[5] J. Bacardit and J. M. Garrell. Evolving Multiple Discretizations with Adaptive Intervals for a Pittsburgh Rule-Based Learning Classifier System. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2724 of *LNCS*, pages 1818–1831. Springer, 2003.

[6] T. Bäck. Generalized convergence models for tournament- and (μ, λ)-selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 2–8, 1995.

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

[8] W. Banzhaf and W. B. Langdon. Some Considerations on the Reason for Bloat. *Genetic Programming and Evolvable Hardware*, 3(1):81–91, 2002.

[9] J. K. Bassett and K. A. De Jong. Evolving Behaviors for Cooperating Agents. In *Foundations of Intelligent Systems: 12th International Symposium*, volume 1932 of *LNAI*, pages 157–165. Springer-Verlag Berlin Heidelberg, 2000.

[10] E. Bernadó-Mansilla and J. M. Garrell. MOLeCS: Using Multiobjective Evolutionary Algorithms for Learning. In *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001*, volume 1993 of *LNCS*, pages 696–710. Springer Verlag, 2001.

[11] E. Bernadó-Mansilla and J. M. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[12] E. Bernadó-Mansilla and T. K. Ho. Domain of Competence of XCS Classifier System in Complexity Measurement Space. *IEEE Transactions on Evolutionary Computation*, 9(1):82–104, 2005.

[13] E. Bernadó-Mansilla, X. Llorà, and J. M. Garrell. XCS and GALE: a Comparative Study of Two Learning Classifier Systems on Data Mining. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *LNAI*, pages 115–132. Springer, 2002.

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

[15] S. Bleuler, M. Brack, L. Thiele, and E. Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 536–543. IEEE Press, 2001.

[16] P. Bonelli, A. Parodi, S. Sen, and S. W. Wilson. NEWBOOLE: A fast GBML System. In *Seventh International Conference on Machine Learning*, pages 153–159. Morgan Kaufmann, 1990.

[17] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[19] M. V. Butz. *Rule-based Evolutionary Online Learning Systems: Learning Bounds, Classification, and Prediction*. PhD thesis, University of Illinois, 2004.

[20] M. V. Butz, P. L. Lanzi, X. Llorà, and D. E. Goldberg. Knowledge Extraction and Problem Structure Identification in XCS. In *Parallel Problem Solving from Nature (PPSN-2004)*, volume 3242 of *LNCS*, pages 1051–1060. Springer, 2004.

[21] M. V. Butz and M. Pelikan. Analyzing the Evolutionary Pressures in XCS. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 935–942. San Francisco, CA: Morgan Kaufmann, 2001.

[22] P Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[23] K. A. De Jong and W. M. Spears. Learning Concept Classification Rules Using Genetic Algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656. Sidney, Australia, 1991.

[24] P. W. Dixon, D. W. Corne, and M. J. Oates. A Preliminary Investigation of Modified XCS as a Generic Data Mining Tool. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *LNCS*, pages 133–150. Springer, 2002.

[25] I. W. Flockhart. GA-MINER: Parallel Data Mining with Hierarchical Genetic Algorithms. Technical Report EPCC-AIKMS-GA-MINER-REPORT 1.0, University of Edinburgh, 1995.

[26] E. Frank and I. H. Witten. Generating Accurate Rule Sets Without Global Optimization. In *Machine Learning: Proceedings of the Fifteenth International Conference*, pages 144–151. Morgan Kaufmann, 1998.

[27] J. M Garrell, E. Golobardes, E. Bernadó-Mansilla, and X. Llorà. Automatic Diagnosis with Genetic Algorithms and Case-Based Reasoning. *Artificial Intelligence in Engineering*, 13:367–372, 1999.

[28] A. Giordana and F. Neri. Search-Intensive Concept Induction. *Evolutionary Computation*, 3(4):375–416, 1995.

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

[30] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[31] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257, 1993.

[32] J. H. Holland. Processing and processors for schemata. In *Associative Information Processing*, pages 127–146, 1971.

[33] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press/ Bradford Books edition, 1975.

[34] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. *Pattern Directed Inference Systems*, pages 313–329, 1978.

[35] J. H. Holmes. Discovering Risk of Disease with a Learning Classifier System. In *Proceedings of the Seventh International Conference of Genetic Algorithms (ICGA97)*, pages 426–433. Morgan Kaufmann, 1997.

[36] C. Janikow. *Inductive Learning of Decision Rules in Attribute-Based Examples: a Knowledge-Intensive Genetic Algorithm Approach.* PhD thesis, University of North Carolina at Chapel Hill, July 1991.

[37] T. Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete and Minimal Representations for Boolean Functions. In *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, 1997.

[38] W. B. Langdon and R. Poli. Fitness causes bloat: Mutation. *Genetic Programming: First European Conference*, pages 37–48, 1998.

[39] X. Llorà. *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.

[40] X. Llorà, J. Bacardit, I. Traus, and E. Bernadó-Mansilla. Where to go once you evolved a bunch of promising hypotheses? In *Learning Classifier Systems, 6th International Workshop, IWLCS 2003*, LNAI. Springer (in press), 2005.

[41] X. Llorà and J. M. Garrell. Automatic Classification and Artificial Life Models. In *Proceedings of Learning00 Workshop.* IEEE and Univesidad Carlos III, 2000.

[42] X. Llorà and J. M. Garrell. Evolving Partially-Defined Instances with Evolutionary Algorithms. In *Proceedings of the 18th International Conference on Machine Learning (ICML'2001)*, pages 337–344. Morgan Kaufmann Publishers, 2001.

[43] X. Llorà and J. M. Garrell. Knowledge-Independent Data Mining with Fine-Grained Parallel Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 461–468. Morgan Kaufmann Publishers, 2001.

[44] X. Llorà and D. E. Goldberg. Bounding the effect of noise in Multiobjective Learning Classifier Systems. *Evolutionary Computation*, 11(3):279–298, 2003.

[45] X. Llorà, D. E. Goldberg, I. Traus, and E. Bernadó-Mansilla. Accuracy, Parsimony, and Generality in Evolutionary Learning Systems via Multiobjective Selection. In *Learning Classifier Systems, 5th International Workshop, IWLCS 2002*, volume 2661 of *LNAI*, pages 118–142. Springer, 2003.

[46] X. Llorà, K. Sastry, and D. E. Goldberg. The Compact Classifier System: Scalability Analysis and First Results. In *Proceedings of the IEEE Conference on Evolutionary Computation.* IEEE press (in press), 2005.

[47] C. J. Merz and P. M. Murphy. 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.

[48] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[49] P. Nordin and W. Banzhaf. Complexity Compression and Evolution. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995.

[50] C. K. Oei, D. E. Goldberg, and S. J. Chang. Tournament selection, niching, and the preservation of diversity. IlliGAL Report No. 91011, University of Illinois at Urbana-Champaign, Urbana, IL, 1991.

[51] A. Orriols Puig and E. Bernadó-Mansilla. Analysis of Reduction Algorithms in XCS Classifier System. In *Recent Advances in Artificial Intelligence Research and Development*, volume 113 of *Frontiers in Artificial Intelligence and Applications*, pages 383–390. IOS Press, 2004.

[52] A. Orriols Puig and E. Bernadó-Mansilla. The Class Imbalance Problem in Learning Classifier Systems: A Preliminary Study. In *Learning Classifier Systems, 7th International Workshop, IWLCS 2005*, LNAI. Springer (in press), 2005.

[53] V. Pareto. *Cours d'Economie Politique, volume I and II.* F. Rouge, Lausanne, 1896.

[54] R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[55] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[56] J. Rissanen. Modeling by shortest data description. *Automatica*, vol. 14:465–471, 1978.

[57] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing, Vol. I, II.* The MIT Press, 1986.

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

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

[60] W. A. Tackett. *Recombination, selection, and the genetic construction of computer programs.* Unpublished doctoral dissertation, University of Southern California, 1994.

[61] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[62] S. W. Wilson. Classifier System Learning of a Boolean Function. Technical Report RIS 27r, The Rowland Institute for Science, 1986.

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

[64] S. W. Wilson. Generalization in the XCS Classifier System. In *Genetic Programming: Proceedings of the Third Annual Conference*, pages 665–674. San Francisco, CA: Morgan Kaufmann, 1998.

[65] S. W. Wilson. Mining Oblique Data with XCS. In *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *LNAI*, pages 158–176. Springer-Verlag Berlin Heidelberg, 2001.

[66] S. W. Wilson. Compact Rulesets from XCSI. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *LNAI*, pages 197–210. Springer, 2002.

[67] I. H. Witten and F. Eibe. *Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 2000.

[68] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, 1999.

[69] E. Zitzler. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report 103, Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, May, 2001.

[70] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.