Memetic Computing manuscript No.
(will be inserted by the editor)

Jaume Bacardit · Edmund K. Burke · Natalio Krasnogor

Improving the scalability of rule-based evolutionary learning

Received: date / Accepted: date

Abstract Evolutionary learning techniques are comparable in accuracy with other learning methods such as Bayesian Learning, SVM, etc. These techniques often produce more interpretable knowledge than e.g. SVM; however, efficiency is a significant drawback. This paper presents a new representation motivated by our observations that Bioinformatics and Systems Biology often give rise to very large-scale datasets that are noisy, ambiguous and usually described by a large number of attributes. The crucial observation is that, in the most successful rules obtained for such datasets, only a few key attributes (from the large number of available ones) are expressed in a rule, hence automatically discovering these few key attributes and only keeping track of them contributes to a substantial speed up by avoiding useless match operations with irrelevant attributes. Thus, in effective terms this procedure is performing a *fine-grained* feature selection at a rule-wise level, as the key attributes may be different for each learned rule. The representation we propose has been tested within the BioHEL machine learning system, and the experiments performed show that not only the representation has competent learning performance, but that it also manages to reduce considerably the system run-time. That is, the proposed representation is up to 2-3 times faster than state-of-the-art evolutionary learning representations designed specifically for efficiency purposes.

Work supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant GR/T07534/01.

J. Bacardit · E.K. Burke · N. Krasnogor
ASAP research group, School of Computer Science, Jubilee Campus, Nottingham, NG8 1BB
E-mail: {jaume.bacardit,natalio.krasnogor}@nottingham.ac.uk

J. Bacardit
Multidisciplinary Centre for Integrative Biology, School of Biosciences, Sutton Bonington, LE12 5RD, University of Nottingham, UK

Keywords Evolutionary Algorithms · Evolutionary Learning · Learning Classifier Systems · Rule Induction · Bioinformatics · Protein Structure Prediction

1 Introduction

Learning Classifier Systems (LCS) (Wilson, 1995; De Jong and Spears, 1991) and Genetics-Based Machine Learning (GBML) (Freitas, 2002), collectively named evolutionary learning (EL) methods for the rest of this paper are very powerful and flexible machine learning tools that have been applied to a very large variety of real-world datasets, small (Bernadó et al, 2001; Bacardit and Butz, 2007) and large (Bacardit et al, 2006; Llorà et al, 2008). However, their computational cost in general is quite high, when compared to other kinds of machine learning techniques (Bacardit, 2004).

In recent years we have extensively applied LCS to data mine large-scale Bioinformatics domains (Bacardit et al, 2006, 2007b; Stout et al, 2008). First we used the GAssist system (Bacardit, 2004) belonging to the Pittsburgh approach of LCS (De Jong and Spears, 1991). This system was designed for data mining purposes, containing components that generate very compact and interpretable solutions as well as some efficiency enhancement techniques. Later on, in order to have better scalability, we created a new system, BioHEL (Bioinformatics-oriented hierarchical evolutionary learning) (Bacardit et al, 2007b). BioHEL applies an iterative rule learning (IRL) approach (Venturini, 1993), and was explicitly designed to handle large-scale datasets. This system allowed us to improve (Bacardit et al, 2007b; Stout et al, 2008) upon our original results (Bacardit et al, 2006) but, more importantly, it enabled us to tackle larger and more complex datasets. Still, its computational cost is high, and this is the main issue that we are addressing in this paper.

In recent years many techniques have been developed to reduce the computational cost of EL methods by using mechanisms such as windowing techniques (Bacardit, 2004), precomputing the possible matches of the individuals (Giráldez et al, 2005) or using the vectorial instructions (SSE, Altivec) available in modern day computers (Llorà et al, 2008).

In this paper we take a different approach where instead of trying to speed up the match operations for a given representation, we modify the representation itself to eliminate irrelevant computations. Previous large scale experiments using LCS (Bacardit et al, 2006) support the observation that in real world datasets containing a large number of attributes, only a small fraction of these are expressed in a given rule. On certain domains this fraction (as will be shown later) can be as small as 5% or less of the available attributes.

In the proposed representation a rule, instead of coding all the domain attributes, keeps only a list containing the expressed ones. Two operators can add attributes to this list (specializing operator) or remove attributes from this list (generalizing operator) with a given probability, and the crossover operator has been adapted to recombine these lists too. As a consequence of keeping only a subset of attributes, match operations only evaluate this subset (ignoring all the non-expressed attributes), thus avoiding hundreds of irrelevant computations. Furthermore, as the representation only holds relevant attributes (if the system is able to discover them), the exploration operators will always recombine/mutate data that matters, potentially leading to

(1) a better learning process and (2) more compact and, hence, interpretable solutions.

We have tested this representation integrated within BioHEL. We have compared our representation against the representation for continuous attributes of the NAX system (Llorà et al, 2008), integrated inside BioHEL. We have decided to compare our representation against this one because it was also explicitly designed for efficiency purposes, using SSE vectorial instructions to boost the match operations.

Our evaluation process consists of two stages. In the first stage, a range of real-world domains of relatively small sizes (UCI data sets) and with a wide set of characteristics is used as test bed for the new representation. By doing this, we can (1) verify that the new representation can, indeed, learn as well as the state-of-the-art and (2) perform a sensitivity analysis on some of its parameters. In the second stage of the evaluation process, and to demonstrate the full potential of the new representation, we recur to much larger problems arising from the bioinformatics domain for which hundreds of attributes are needed. Afterwards, the performance of the best configuration of our representation is compared to two standard machine learning techniques. We also propose a simple model of the efficiency gain of this representation against the NAX representation.

The rest of the paper is structured as follows: First, section 2 summarizes some related work. Next, section 3 describes the framework of the paper: the BioHEL system and the NAX rule representation for continuous attributes. Section 4 contains the full description of our attribute list knowledge representation After the definition of the new representation, section 5 describes the experimental design, section 6 shows and analyzes the results of our experiments. Finally, section 7 provides some conclusions and details of further work.

2 Related work

This paper combines two different areas of research within an EL context: representations for continuous attributes and efficiency enhancement techniques. This related work section mainly comments on both topics.

In relation to the first topic, rule representations for real-valued attributes, many different approaches exist. Some methods use some kind of discretization (Giráldez et al, 2003; Divina et al, 2003; Bacardit, 2004) to convert the continuous attributes to categorical ones, which allows the usage of categorical knowledge representations (De Jong and Spears, 1991; Wilson, 1995). A different alternative is to use rules based on fuzzy logic. A good review of these systems is (Cordón et al, 2001).

Another approach, in which the work presented in this paper can be placed, is to use rules that define a hyper-rectangle in the feature space, having for each dimension an interval with a lower bound and an upper bound. Many different EL methods use such kind of representation (Corcoran and Sen, 1994; Wilson, 1999; Stone and Bull, 2003; Llorà et al, 2008). One of these methods (Wilson, 1999) defines the intervals associated to each attribute using two variables: one specifying the center of the interval and the other (spread) specifying the size of the interval. The intervals defined using this approach always remain semantically correct after any possible crossover

or mutation operation. The other methods (Corcoran and Sen, 1994; Stone and Bull, 2003; Llorà et al, 2008) define the intervals also using two variables: a lower bound and an upper bound. In this case, crossover or mutation operations may result in intervals where the variable associated to the lower bound has higher value than the variable associated to the upper bound. To overcome this potential ambiguity, several strategies have been proposed, such as ignoring the attribute (thus, making it irrelevant) (Corcoran and Sen, 1994; Llorà et al, 2008) or assuming that there is no fixed order to specify the bounds: the lowest value is always the lower bound and the highest value is always the upper bound (Stone and Bull, 2003).

Finally, there are some EL approaches that do not use rules as their knowledge representation. Some examples of alternative knowledge representations are decision trees (either axis-parallel or oblique). These methods rely on either generating a full tree by means of genetic programming operators (Llorà and Garrell, 2001), using an heuristic method to generate the tree then and using a Genetic Algorithm and an Evolution Strategy to optimize the test performed at each node (Cantu-Paz and Kamath, 2003). Another alternative is the generation of a set of synthetic instances used as the core of a *k-NN* classifier (Llorà and Garrell, 2001).

In relation to methods to alleviate the run-time of EL methods, many alternatives also exist. Some method apply various kinds of windowing techniques (Bacardit et al, 2004). That is, they use only a subset of the training examples to perform the fitness computations of the individuals. The strategy to choose the training subset, the frequency of changing them, etc. define each of them. (Freitas, 2002) contains a taxonomy of such methods. A different approach consists in "precomputing" the possible classifications of the system (Giráldez et al, 2005). The authors exploit the fact that they deal with either nominal or discretized attributes to generate tree structure to efficiently know which examples belong to each value of each attribute (the one corresponding to the rule being evaluated). Furthermore, the matches of a given rule are the intersection of all these subsets of examples. Finally, a recent efficiently enhancement approach is the use of SSE/Altivec/etc. vectorial instructions to perform the match operations (Llorà and Sastry, 2006; Llorà et al, 2008), with the objective of performing various match operations (at the attribute level) at the same time. In (Llorà and Sastry, 2006) the use of SSE operations allows to perform up to 64 attribute matches for binary attributes at the same time, and in (Llorà et al, 2008), four match operations for continuous attributes are performed at the same time.

As we mentioned before, the maintenance of a relevant attribute list effectively represents a feature selection (FS) strategy (FS) (Guyon and Elisseeff, 2003) strategy. There are various works on the use of evolutionary computation for FS (Vafaie and De Jong, 1992; Yang and Honavar, 1998; Ruiz, 2007). Most of these works define a binary chromosome with one bit associated to each attribute of the domain, which decides whether the feature is selected or not. Fitness functions of these methods try to reduce the subset of selected features while attempting to preserver a high accuracy. The approach in this paper is slightly different for two reasons. (1) feature selection and learning are performed in a concurrent and integrated manner, and (2) our feature

selection is performed at a *rule-wise* level. That is, different rules can use different sets of relevant attributes. Standard FS methods filter the dataset keeping only the same subset of attributes for the whole solution (e.g. rule set). Thus, we could say that our representation applies a *fine-grained* FS process.

Another approach from which we draw some inspiration is the Evolutionary Concept Learner (ECL) (Divina and Marchiori, 2005) that uses an evolutionary approach to do inductive logic programming. In this case, rules are Horn Clauses having a list of terms, each of them associated to an attribute. Terms can be added or removed to/from the list using mutation operators, which is similar to our approach. ECL, however, does not have a crossover operator nor was designed to deal efficiently with very large scale datasets.

3 Framework

In this section we describe the proposed framework. In particular we give details about BioHEL (Bacardit and Krasnogor, 2006a; Bacardit et al, 2007b; Stout et al, 2008) and the performance boosting wrapper method based on ensembles (Bacardit and Krasnogor, 2006b) that we have used. As we compare our representation against NAX (Llorà et al, 2008), we describe it in some detail too.

3.1 The BioHEL system

BioHEL has recently been applied (Bacardit et al, 2007b; Stout et al, 2008) to large-scale Bioinformatics datasets. For a full description of the system, as well as the justification for its design issues, please see (Bacardit and Krasnogor, 2006a). BioHEL (Bioinformatics-oriented Hierarchical Evolutionary Learning) follows the Iterative Rule Learning or Separate-and-Conquer approach (Fürnkranz, 1999), which was first used in EL by Venturini (Venturini, 1993). BioHEL is strongly influenced by the GAssist (Bacardit, 2004) Pittsburgh LCS. Several of BioHEL features have been inherited from GAssist. The system applies an almost standard generational GA, which evolves individuals that are classification rules.

The learning process creates a rule set by iteratively learning one rule at a time using a GA. After each rule is obtained, the training examples that are covered by this rule are removed from the training set, thus the GA is forced to explore other areas of the search space to learn other rules. The rules are inserted into a rule set with an explicit default rule that covers the majority class of the domain. The evolved rules will cover all the other classes. Therefore, the iterative search for new rules stops when it is impossible to find any rule where the associated class is not the majority class of the matched examples. When this happens, all remained examples are assigned to the default rule.

Moreover, several repetitions of the GA with the same set of instances (and different initial random seeds) are performed. Only the best rule from all these repetitions will be inserted to the rule set and the examples covered by this rule removed from the training set. Figure 1 contains the pseudo-code of the general workflow of BioHEL.

Fig. 1 General workflow of BioHEL

```
Procedure BioHEL general workflow
Input : TrainingSet
RuleSet = ∅
stop = false
Do
    BestRule = null
    For repetition=1 to NumRepetitionsRuleLearning
        CandidateRule = RunGA(TrainingSet)
        If CandidateRule is better than BestRule
            BestRule = CandidateRule
        EndIf
    EndFor
    Matched = Examples from TrainingSet matched by BestRule
    If class of BestRule is the majority class in Matched
        Remove Matched from TrainingSet
        Add BestRule to RuleSet
    Else
        stop = true
    EndIf
While stop is false
Output : RuleSet
```

To reduce the run-time of the system, BioHEL uses a windowing scheme called ILAS (incremental learning with alternating strata) (Bacardit et al, 2004). This mechanism divides the training set into several non-overlapping subsets and chooses a different subset at each GA iteration for the fitness computations of the individuals.

The explicit default rule mechanism of GAssist (Bacardit et al, 2007a) has also been implemented in BioHEL. This mechanism selects one of the classes of the domain as default class, and takes it out from the evolutionary pool. Thus, the evolved rules will cover all the classes of the domain but the default class. Many policies exist in GAssist to decide which class is used as default class. From those, three policies haven been implemented: using the majority class, using the minority class or having no explicit default rule.

The fitness function of BioHEL is based on the Minimum Description Length (MDL) principle (Rissanen, 1978). The MDL principle is a metric applied to a theory (a rule in our case) which balances its complexity and accuracy. The fitness function is defined as follows:

$$Fitness = TL \cdot W + EL \tag{1}$$

where TL stands for theory length (the complexity of the solution) and EL stands for exceptions length (the accuracy of the solution). BioHEL seeks to minimize it.

W is a weight that adjusts the relation between TL and EL. BioHEL uses the automatic weight adjustment heuristic proposed for GAssist (Bacardit, 2004). The parameters of this heuristic are adjusted as follows: Initial TL ratio: 0.25, weight relaxation factor: 0.90, max iterations without improvement: 10.

TL is defined as follows:

$$TL(R) = \frac{\sum_{i=1}^{i=NA}(1 - Size(R_i)/Size(D_i))}{NA} \quad (2)$$

where R is a rule, NA is the number of attributes of the domain, $Size(R_i)$ is the width of the interval associated to the attribute i of R and $Size(D_i)$ is the width of attribute i's domain. TL is normalized between 0 and 1. It has been designed like this to simplify the tuning of W. TL promotes rules with as few expressed attributes as possible, and where the few expressed attributes have a size as large as possible. Thus promoting, when possible, general rules.

The design of EL has to take into account a suitable trade off between accuracy and coverage thus covering as many examples as possible without sacrificing accuracy. To this end, our proposed coverage measure is initially biased towards covering a certain minimum of examples (named Coverage Breakpoint - CB) and once this CB has been reached the bias is reduced. EL is defined as follows:

$$EL(R) = 2 - ACC(R) - COV(R) \quad (3)$$

$$ACC(R) = \frac{corr(R)}{matched(R)} \quad (4)$$

$$COV = \begin{cases} CR \cdot \frac{RC}{CB} & \text{If } RC < CB \\ CR + (1 - CR) \cdot \frac{RC - CB}{1 - CB} & \text{If } RC \geq CB \end{cases} \quad (5)$$

$$RC = \frac{matched(R)}{|T|} \quad (6)$$

COV is the adjusted coverage metric that promotes the coverage of (at least) a minimum number of examples, while RC is the raw class-wise coverage metric: the percentage of examples from the class predicted by this rule that is actually matched by the rule. ACC is the accuracy of the rule, $corr(R)$ is the number of examples correctly classified by R, $matched(R)$ is the number of examples matched by R, CR (Coverage Ratio) is the weight given in the coverage formula to achieving the minimum coverage, CB is the coverage breakpoint and $|T|$ is the total number of training examples.

3.2 The rule representation of the NAX system

The NAX system (Llorà et al, 2008), as described in the related work section, uses rules that encode an hyperrectangle. The hyperrectangle is defined by specifying an interval within the domain of each attribute of the problem. Formally, a rule encodes a predicate such as:

$$\text{If } l_1 \leq a_1 \leq u_1 \wedge l_2 \leq a_2 \leq u_2 \wedge \ldots \wedge l_n \leq a_n \leq u_n \text{ then predict class } c1 \quad (7)$$

Where l_i and u_i are, respectively, the lower and upper bounds of the interval for attribute i, a_i is the value of attribute i for the instance being predicted and $c1$ is the class of the domain that this rule is predicting.

Encoded in each rule predicate in the NAX system are two vectors whose size is the number of attributes in the domain. One vector stores all the lower bounds for each attribute and the other vector stores the upper bounds. Within a given rule, if $u_i < l_i$ implies that attribute i has been deemed to be irrelevant.

Match operations in NAX are boosted by using SSE2 vectorial instructions. These instructions are able to perform four attribute match operations in parallel. Implementation details are reported in (Llorà et al, 2008).

The initialization of a new rule in NAX works as follows (Llorà, 2008): a parameter specifies the expected value of expressed attributes in a rule. Given this parameter and the number of attributes of the domain a probability of expressing attributes is computed and then, for each attribute in a rule, this probability is used to decide whether the attribute is expressed or not. Afterwards, the lower and upper bounds of the expressed attributes are initialized.

Finally, NAX uses one-point crossover and a mutation operator based on generating new random boundary elements.

3.3 Simple ensemble for consensus prediction

As to improve accuracy and leveraging the fact that BioHEL is a stochastic algorithm we generate several rules sets by running it multiple times with different random seeds. We then join these rule-sets by considering them as an ensemble based on a simple majority vote, thus combining their predictions. This approach, similar to Bagging (Breiman, 1996), has shown to be able to improve system performance for the GAssist (Bacardit, 2004) LCS across a broad range of datasets (Bacardit and Krasnogor, 2006b). In the rest of the paper, all reported accuracy measures will be the result of a such an ensemble prediction.

4 Boosting the efficiency of match operations: the attribute list knowledge representation

This section contains a detailed explanation of the proposed rule representation designed to deal with large scale continuous problems. This description includes a definition of the representation, initialization strategy, exploration operators and, finally, match process.

4.1 Representation definition

Each rule will be represented by four elements, as shown in the example in figure 2: (1) an integer containing the number of expressed attributes, (2) a vector specifying which attributes are expressed, (3) a vector specifying, for each expressed attribute, the lower and upper bound of its associated interval and (4) the class associated to the rule.

4.2 Initialization strategy

The initialization of the rules in any kind of EL system is crucial for a proper learning process, and it becomes critical when dealing with problem domains that contain hundreds of attributes. During initialization, two factors have to be balanced: on the one hand, rules have to be sufficiently general as to be able to match some examples. On the other hand, rules cannot be overgeneral

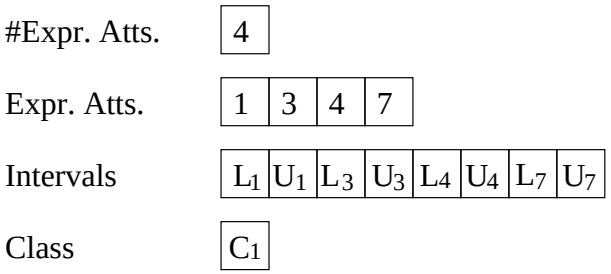


Fig. 2 Example of a rule in the attribute list knowledge representation with four expressed attributes: 1, 3, 4, and 7. l_n = lower bound of attribute n, u_n= upper bound of attribute n, $c1$=Class 1 of the domain

as this will decrease the accuracy of the classifier and slow down the learning process. This issue has been studied in depth for Michigan LCS (Butz, 2006).

For continuous problems, a rule representation for continuous attributes would need to take into account two specific issues: (1) how many and which attributes would be relevant within a rule and (2) how would intervals be defined for the relevant attributes. As suggested by the NAX representation we address the first issue by having a parameter that specifies the expected value of the number of relevant attributes in a rule. The expected value is transformed into a probability and then, using this probability, the relevant attributes are chosen. The answer for the second question means having to decide the size of the interval and its position within the domain of the attribute. The size of the interval is randomly initialized with uniform distribution to be between 25% and 75% of the domain size. Moreover, we use a covering mechanism similar to (Wilson, 1999), that picks a random training example and initializes the interval to be centered at the value of that instance for the attribute being initialized. In case the interval would overlap with lower or upper bound of the domain, it is shifted to place it fully within the attribute domain. The policy used to pick the training examples for the rule initialization is the class-wise sampling without replacement policy from (Bacardit, 2005).

4.3 Exploration operators

The operators used to explore the search space of this representation are divided in two parts: (1) the traditional crossover and mutation operators that edit the intervals for the expressed attributes and (2) the operators that edit the list of expressed attributes.

The crossover operator that we have designed for this representation could be defined as a "virtual one point crossover". In a standard hyperrectangle representation, a one point crossover operator would randomly select a cut point within the chromosome laying in the same position (attribute) for both parents and exchange all the genes after the cut point between parents. In our representation we do the following (exemplified in figure 3):

- We take the first parent and we randomly choose one of its expressed attributes

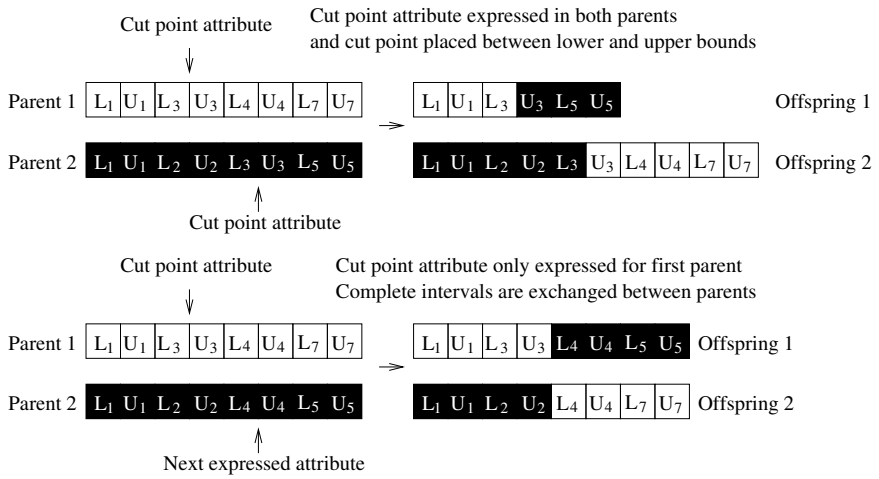


Fig. 3 Example of the crossover operator of the attribute list knowledge representation for two cases: when the cut point attribute is expressed in both parents or only in the first parent

- We check if such attribute is expressed for the other parent
- If the attribute is expressed for both parents then
  - We select a cut point either before the lower bound, between the two bounds or after the upper bound, and we exchange the genes accordingly between the parents
  - We exchange between parents the intervals for the rest of attributes in the list
- If the attribute is not expressed in the second parent then
  - We find the next attribute (in the order in which the attributes are defined in the dataset) that is expressed in the second parent
  - We exchange between parents the complete intervals (without mixing bounds) after the cut point: Parent 1 gives all the intervals after the cut point attribute and parent 2 gives the intervals starting at the next expressed attribute after the cut point attribute
- One offspring randomly picks the class value from one parent and the other offspring from the other parent.

The mutation operator selects one bound with uniform probability and adds or subtracts a randomly generated offset to the bound, of size (picked with uniform distribution) between 0 and 50% of the attribute domain. If the mutation affects the class value of the rule, a different class value is assigned to the rule, picked at random.

Both crossover and mutation operators can create intervals where the lower bound is higher than the upper bound. In this case we "repair" the rules by swapping the bounds.

To add or remove attributes to/from the list of expressed attributes we have two operators, called *specializing* and *generalizing* respectively. These operators are applied to the offspring population after mutation given a cer-

tain individual-wise probability. When the generalizing operator is applied to an individual, we randomly selects with uniform probability one of the expressed attributes and we remove it and its interval from the rule. When the specializing operator is applied, one of the attributes from the domain not expressed in the rule is randomly selected and it is added to the list. An interval for the attribute is randomly initialized using the same policy explained above for the initialization stage.

4.4 Match process

The match process of a rule with this representation is simple:

- For each attribute in the list of expressed attributes:
  - Check if the instance's value for the attribute lays between the bounds
  - If not, the match process returns false, if true, continue with the next attribute
- If all of the instance's values were within the bounds of the rule intervals for each attribute, match returns true

5 Experimental design

This section contains the description of the experimental design of the results reported in this paper. The motivation for these experiments is two-fold: first to study its behavior and verify if this representation can learn, at least, as well as the state-of-the-art in EL research and second to test the representation in large datasets with hundreds of attributes to show all of its power. The designed experiments and datasets will cover both objectives.

5.1 Experiments

In order for this representation to be able to learn properly it has to achieve two objectives: (1) that the relevant attributes are identified and (2) that the correct intervals for these attributes are generated. The first issue is the most critical one. If the representation is not able to identify which attributes should be included in the list, the search through the intervals space is futile. Thus the application of the generalizing and specializing operators has to be performed at the appropriate rate. If the probability of application of these operators is too low, the representation will learn slower than it should, and potentially converging towards a local optima. If the probability is too high we may destroy good solutions. To analyze how sensitive is the representation to these parameters we have tested several values of them: 0.05, 0.10, 0.15, 0.20 and 0.25. That is, a probability of adding and removing attributes ranging from 5% to 25% of the individuals in the offspring population. The representation will be compared against the rule representation of the NAX system, described in section 2.

5.2 Datasets

Two kinds of datasets have been used in the paper. First, we have used a variety of real-world datasets from the well-known UCI repository (Blake et al, 1998). Afterwards, we tested the representation in a couple of large scale bioinformatics datasets with high number of attributes and instances.

5.2.1 UCI datasets

In order to determine whether this representation can learn properly, a test suite with diverse characteristics in terms of number of attributes, number of instances, classes, etc. is needed. To this end, we have selected 16 UCI datasets (summarized in table 1). For simplicity, we have selected datasets that only have continuous attributes, as this initial version of the representation is designed for this kind of features.

Table 1 Features of the datasets used in this paper. #Inst. = Number of Instances, #Attr. = Number of attributes, #Cla. = Number of classes, Dev.cla. = Deviation of class distribution

Dataset Properties				
Code	#Inst.	#Attr.	#Cla.	Dev.cla.
bal	625	4	3	18.03%
bpa	345	6	2	7.97%
gls	214	9	6	12.69%
h-s	270	13	2	5.56%
ion	351	34	2	14.10%
irs	150	4	3	—
pen	10992	16	10	0.43%
sat	6435	36	6	6.78%
seg	2310	19	7	0.25%
son	208	60	2	3.37%
thy	215	5	3	25.78%
wav	5000	40	3	0.48%
wbcd	699	9	2	15.52%
wdbc	569	30	2	12.74%
wine	178	13	3	5.28%
wpbc	198	33	2	26.26%

For all these datasets BioHEL used its standard configuration (Bacardit et al, 2007b), summarized in table 2. We applied a stratified ten fold cross-validation to partition these datasets into training and test sets. The process was repeated three times. Therefore all the accuracies reported later in the paper for these datasets is the average of the accuracy obtained in the 30 test sets. Moreover, these training and test partitions are the same that were used in (Bacardit, 2004), so the results obtained in this paper can be directly compared to all the machine learning methods that were tested in that previous work. The expected value of number of expressed attributes in the initialization of both the NAX and our representation has been set to 15 (or the number of attributes in the domain, in case of domains with less than 15 attributes) for all the experiments in this paper. This parameter was tuned with some previous experiments (not reported).

5.2.2 Bioinformatics datasets

The two bioinformatics datasets are part of our research in protein structure prediction (PSP). Proteins are heteropolymer molecules constructed as a chain of residues or amino acids of 20 different types. This string of amino acids is known as the primary sequence. In its native state, the chain folds

Table 2 Parameters of BioHEL for the experiments with the UCI repository datasets

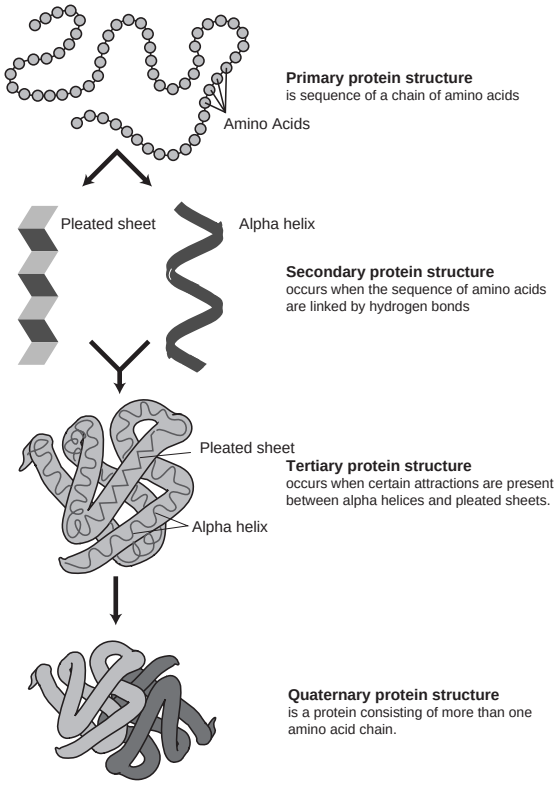
Parameter	Value
General parameters	
Crossover prob.	0.6
Selection algorithm	tournament
Tournament size	4
Population size	500
Individual-wise mutation prob.	0.6
Default class policy	disabled
Iterations	100
Number of strata of ILAS	2
Repetitions of rule learning process	2
Expected value of #expressed att. in init.	15
Rule sets per ensemble	10
MDL-based fitness function	
Iteration of activation	10
Initial theory length ratio	0.01
Weight relax factor	0.90
Coverage breakpoint	0.10
Coverage ratio	0.90

to create a 3D structure. It is thought that this folding process has several steps. The first step, called secondary structure, consists of local structures such as alpha helix or beta sheets. These local structures can group in several conformations or domains forming a tertiary structure. Secondary and tertiary structure may form concomitantly. The final 3D structure of a protein consists of one or more domains. All this process is represented in figure 4. It is very costly to empirically determine this complex 3D structure and, therefore, it needs to be predicted. Even after many decades of research PSP is still an open problem, with many different ways to address it. One of the possible solutions is to concentrate in predicting some structural features of the amino acids in a protein, and use these predictions, among other aims, to constrain the search space of the 3D PSP problem.

The two datasets we use in this paper predict two of these structural features. The first of them is the secondary structure (SS) of the protein residues. That is, the local structural motif that a protein residue can be part of. These local motifs are here categorized in three classes: alpha, beta and coil. We have used a protein dataset called CB513 (Cuff and Barton, 1999) of 513 proteins and 83823 residues (instances). Each instance is characterized with 300 continuous attributes that collects statistical information derived from position-specific scoring matrices (PSSM) (Wood and Hirst, 2005) computed from the target residue (the one for which we are predicting the secondary structure) and its nearest neighbours in the chain.

The next dataset predicts the coordination number (CN) of protein residues. In the native state each residue will have a set of spatial nearest neighbours. The number of nearest neighbours within a certain distance of a given residue is its coordination number. We have used the coordination number definition and dataset from (Bacardit et al, 2006). The dataset contains 257560 instances and 180 continuous attributes, again using the PSSM

Fig. 4 Graphical representation of protein folding. Top: residues in the unfolded chain are represented by a chain of circles. Next, residues begin to form contacts. Short range contacts lead to formation of helical and pleated sheet structures. Finally the overall folded structure is formed. (Illustration Courtesy of National Human Genome Research Institute)



representation. This specific dataset predicts whether the coordination number of a residue is higher or lower than the mid-point of the CN domain. Thus, it is a dataset with two classes.

Both datasets were partitioned into training and test sets using the stratified ten-fold cross-validation procedure. The same configuration of BioHEL detailed in the previous subsection has been used except for the following three parameters: default class policy set to major; number of strata of the ILAS windowing scheme set to 20; coverage breakpoint is 0.0025 for the SS dataset and 0.001 for the CN one.

6 Results

6.1 UCI datasets

Table 3 shows the accuracy results of BioHEL using both the NAX representation and the attribute list knowledge representation. We used a Friedman statistical test for multiple comparisons as suggested in (Demšar, 2006) to

test whether the differences showed by the different compared methods (NAX vs attribute list with various parameter sets) was significant or not. The result of the test was that the performance differences between the compared methods were not significant. That is, our proposed representation is able to achieve, for at least the 16 UCI datasets, a similar performance to that of the state of the art, namely NAX.

Table 3 Accuracy of BioHEL using the NAX and the attribute list knowledge representation on the UCI datasets

Dataset	NAX	Prob. of Generalize and Specialize in Att. List KR				
		0.05	0.10	0.15	0.20	0.25
bal	88.0±3.2	87.4±3.9	88.2±3.6	88.7±3.4	87.7±4.4	88.2±4.1
bpa	68.9±7.2	69.5±7.5	70.0±7.2	69.7±6.3	68.6±8.4	69.3±7.9
gls	74.0±10.4	75.0±8.2	74.4±8.5	75.4±9.4	76.2±7.9	77.8±7.5
h-s	78.9±8.7	78.5±6.4	79.3±9.1	79.0±8.7	77.9±7.9	77.9±7.4
ion	93.1±4.7	93.0±3.8	92.7±4.5	92.5±4.2	92.0±4.3	91.6±4.0
irs	94.4±4.6	93.6±4.7	93.1±5.0	93.6±4.7	93.8±4.5	94.0±4.7
pen	95.2±0.8	94.8±1.0	94.8±1.0	94.6±0.9	94.0±1.1	94.0±1.2
sat	88.5±1.3	88.5±1.2	88.2±1.0	88.5±1.2	88.5±1.0	88.4±1.1
seg	97.1±0.8	97.0±1.0	97.0±0.8	97.0±0.9	97.2±0.8	97.1±0.8
son	81.3±9.5	82.3±9.0	82.7±9.2	81.4±11.6	82.9±7.2	83.7±7.3
thy	94.3±5.3	93.0±4.6	94.1±4.2	93.2±5.1	93.0±4.2	93.5±5.2
wav	84.3±1.7	84.7±1.4	84.6±1.5	85.1±1.6	84.6±1.6	84.7±1.7
wbcd	95.3±2.6	95.5±2.3	95.4±2.6	95.5±2.1	95.5±2.4	95.4±2.5
wdbc	95.9±2.7	95.8±2.6	96.4±2.9	96.2±2.5	96.1±2.9	96.0±2.5
wine	93.0±6.5	92.3±6.5	92.4±6.8	91.5±5.5	92.3±6.3	92.4±6.9
wpbc	78.5±7.5	78.4±7.8	78.5±8.1	77.3±6.5	77.4±7.9	78.4±7.5

Table 3 also provides some insight on the dependence of the proposed representation to the generalizing/specializing probabilities. Our aim in testing various values for these probabilities is to check how sensitive was the representation to these parameters, and to determine if a value too high would be harmful. The accuracy results reported in table 3 suggest that a high probability of specializing and generalizing is not harmful, as all tested configurations obtained statistically similar results.

Thus, we turn next to evaluate the run-time differences among the studied representations/parameter values, in order to determine how efficient is the new representation. Table 4 shows the average run-time of the BioHEL runs for each dataset and tested configuration. All experiments reported in the paper were performed using Opteron processors running at 2.2GHz, Linux operating system and the C++ implementation of BioHEL. A batch version of BioHEL has been used, without any form of parallelism.

We can observe some interesting results. First of all, for most datasets an increasing probability of generalizing and specializing the attribute list manages to reduce the run-time of BioHEL. To evaluate rigorously these run-time differences we have applied again the Friedman test for multiple comparisons. The test indicated that there are significant performance differences between the tested BioHEL configurations with a p-value of $2.7e^{-}5$. the Holm post-hoc test (Demšar, 2006) (with 95% confidence level) indicated two different groups of methods in relation to run-time. The two configurations of the attribute list knowledge representation with a probability of 0.20

and 0.25 obtained similar performance, and both of them were significantly better than all the other configurations of our representation as well as the NAX representation.

Even if our representation (with high probability of generalizing & specializing) is significantly better than NAX in general, in some datasets (the ones with low number of attributes) NAX is faster than our representation. On the other hand, in datasets with high number of attributes (such as sat or son) the proposed representation is substantially faster than the NAX representation.

Table 4 Average run-time of BioHEL's training process (in seconds) using the NAX and the attribute list knowledge representation on the UCI datasets

Dataset	NAX	Prob. of Generalize and Specialize in Att. List KR				
		0.05	0.10	0.15	0.20	0.25
bal	9.4±0.5	12.5±0.6	12.4±0.7	12.4±0.6	11.9±0.6	11.9±0.6
bpa	6.0±0.3	7.3±0.4	7.4±0.4	7.4±0.4	7.3±0.4	7.4±0.4
gls	4.5±0.3	5.5±0.4	5.4±0.4	5.4±0.4	5.2±0.3	5.3±0.3
h-s	4.4±0.3	5.0±0.4	4.8±0.3	4.7±0.3	4.5±0.3	4.5±0.3
ion	3.7±0.5	2.5±0.3	2.5±0.3	2.4±0.3	2.4±0.3	2.4±0.3
irs	1.3±0.2	1.6±0.2	1.6±0.2	1.6±0.2	1.6±0.2	1.6±0.2
pen	174.0±13.7	179.2±15.0	167.1±13.6	160.4±13.5	152.0±12.3	146.4±11.8
sat	110.3±6.1	81.3±3.9	79.6±3.6	78.9±3.5	75.4±3.5	76.2±3.4
seg	24.9±1.5	20.4±1.3	19.0±1.2	18.0±1.1	17.3±1.0	16.9±1.0
son	4.4±0.3	2.5±0.2	2.5±0.2	2.4±0.2	2.4±0.2	2.4±0.2
thy	1.6±0.2	2.0±0.2	1.9±0.2	2.0±0.2	1.9±0.2	2.0±0.2
wav	105.1±1.5	80.1±1.3	80.4±1.3	80.4±1.3	79.6±1.3	78.9±1.3
wbcd	3.6±0.4	3.8±0.5	3.4±0.5	3.1±0.4	3.0±0.4	2.9±0.4
wdbc	4.9±0.3	3.4±0.3	3.3±0.3	3.3±0.3	3.2±0.2	3.2±0.3
wine	1.2±0.1	1.3±0.1	1.2±0.1	1.2±0.1	1.2±0.1	1.2±0.1
wpbc	3.7±0.3	3.0±0.3	3.0±0.3	2.9±0.3	2.8±0.3	2.9±0.3

6.2 Bioinformatics datasets

We turn now to the second stage of our evaluation process. For which we rely on much larger and harder (as shown in our previous work (Bacardit et al, 2006)) datasets. Here we aim at showing the full power of the representation, in what pertains to both increased performance (accuracy), and reduced run-time (efficiency). Table 5 shows, for each of the two tested datasets, the obtained accuracy, average rule-set size, average number of expressed attributes per rule and run-time (reported in hours). In the introduction section we said that this representation could potentially help explore better the search space because it only needs to recombine relevant attributes. In these datasets, specially in the secondary structure dataset, we can see that this is, indeed, the case. The attribute list knowledge representation manages to obtain an accuracy 1% higher than the NAX representation. In the coordination number dataset our representation also obtains higher accuracy, although the difference is minor.

We can clearly see the difference in the learning capacity of both representations in figure 5. This figure plots how the training accuracy of BioHEL for the SS dataset increases with each new rule being learned, considering that all examples not matched by the learned rules will be classified using

Table 5 Results of the experiments on the bioinformatics datasets

Dataset	Result	NAX	Prob. of Generalize and Specialize in Att. List KR				
			0.05	0.10	0.10	0.20	0.25
SS	Acc.	72.4±1.0	73.3±0.8	73.4±0.9	73.3±0.8	73.3±0.8	73.2±0.7
	#rules	268.7±13.6	290.9±10.4	281.6±10.3	271.4±10.3	263.4±7.8	253.3±9.1
	#exp. att.	13.1±3.0	14.6±3.2	14.4±3.2	14.1±3.2	13.7±3.2	13.4±3.2
	run-time (h)	16.1±0.9	6.4±0.4	6.0±0.6	5.9±0.6	5.7±0.4	5.6±0.4
CN	Acc.	80.9±0.4	81.1±0.4	81.1±0.4	81.1±0.4	81.0±0.4	81.0±0.4
	#rules	263.2±12.6	284.7±12.5	275.1±13.3	265.5±13.4	255.5±11.2	245.1±11.8
	#exp. att.	14.3±2.9	16.3±3.0	16.1±3.1	15.7±3.1	15.2±3.1	14.8±3.1
	run-time (h)	45.7±2.5	30.9±2.1	29.8±2.3	28.9±2.3	28.1±1.8	26.7±2.0

the domain majority class. In the figure we can see how the performance difference between the NAX and our representation increases with every rule being learned, being our representation able to learn better. We would like to remind the reader that what we have reported in table 5 is the accuracy obtained by an *ensemble* of rule sets. This is the reason that the test accuracy is higher than the training accuracy showed in figure 5.

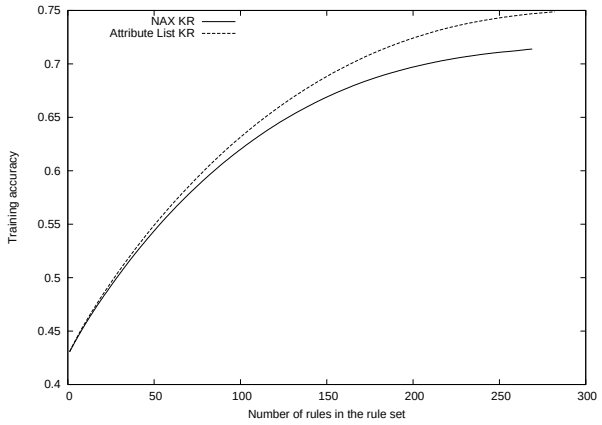


Fig. 5 Evolution of the training accuracy of BioHEL for the SS dataset when adding new rules to the rule set

Higher probabilities of generalizing and specializing show the same behavior observed in the experiments with the UCI datasets: Higher run-time reduction and, in this case, also generation of more compact solutions, with smaller rule sets and expressed attributes per rule. The obtained run-time reductions are quite important. In the SS dataset we have managed to reduce the average run-time from more than 16 hours to less than 6 hours. Our representation is almost three times faster than NAX. On the CN dataset our representation is up to 1.7 times faster than NAX. The run time of our representation is up to 19 hours shorter.

6.3 Comparison to other machine learning techniques

In order to place in a wider context the performance levels of our representation, we have picked up a couple of standard machine learning technique such: the well-known C4.5 (Quinlan, 1993), a rule induction system and Naive Bayes (NB) (John and Langley, 1995), a Bayesian learning algorithm and trained them on all the datasets used in the paper. We have used the implementations of both systems included in the Weka (Witten and Frank, 2000) machine learning package using the default parameters. Table 6 contains the test accuracy of both C4.5 and NB and includes also for comparison the performance of our representation using a probability of generalize and specialize of 0.25 (the best configuration so far).

The results of this comparison indicate that BioHEL using the new representation was the best method 11 times. Naive Bayes was the best method in 6 datasets and C4.5 was the best method only two times. 9 of the 11 datasets where BioHEL was the best method are precisely the 9 datasets with larger number of attributes, the ones for which the attribute list rule representation has been designed to tackle. Moreover, the performance difference is specially large in the two bioinformatics datasets. The results in table 6 were again analyzed using the Friedman test for multiple comparisons, that determined that the performance differences between the methods was significant with a p-value of 0.03795. The post-hoc Holm test revealed that the control method (BioHEL) was significantly better than both C4.5 and Naive Bayes with a confidence level of 95%.

Table 6 Accuracy of C4.5, Naive Bayes and BioHEL using the attribute list knowledge representation all the tested datasets. Best method for each dataset is marked in bold

Dataset	C4.5	NB	BioHEL
bal	77.7±2.9	**90.3±1.7**	88.2±4.1
bpa	65.7±7.9	55.7±8.8	**69.3±7.9**
gls	68.8±9.6	48.9±9.7	**77.8±7.5**
h-s	79.8±6.5	**84.0±8.0**	77.9±7.4
ion	89.0±5.9	83.2±8.1	**91.6±4.0**
irs	94.2±5.4	**95.8±5.3**	94.0±4.7
pen	**96.5±0.7**	85.8±1.2	94.0±1.2
sat	86.5±1.4	79.6±1.5	**88.4±1.1**
seg	**97.1±1.1**	80.3±1.8	**97.1±0.8**
son	73.2±8.6	68.8±10.8	**83.7±7.3**
thy	92.7±5.9	**97.0±3.9**	93.5±5.2
wav	75.1±1.5	80.0±1.8	**84.7±1.7**
wbcd	94.3±2.8	**96.0±2.4**	95.4±2.5
wdbc	93.3±3.3	93.3±3.0	**96.0±2.5**
wine	92.2±6.4	**96.6±4.1**	92.4±6.9
wpbc	71.1±8.6	68.2±9.3	**78.4±7.5**
SS	55.1±0.7	67.7±0.7	**73.2±0.7**
CN	73.1±0.4	75.7±0.6	**81.0±0.4**

6.4 Modelling the efficiency of the representation

As we have seen in the results presented earlier in this section, our representation is faster than the NAX representation in many datasets, specially in

the ones with larger number of attributes, but it is also slower than NAX sometimes. Thus, we would like to investigate further into the relative efficiency gains between the two systems to determine if we can model this gain, by determining in which circumstances the proposed representation is faster or slower than NAX.

The answer to the first question is straightforward: Our representation is faster than NAX in the match process, where it can exploit the fact that it only holds information from the relevant attributes of the domain. The complexity of the match process for NAX is $O(N)$ where N is the number of attributes in the domain[1], while our representation has a complexity of $O(E)$, where E is the number of expressed attributes in the domain. The performance advantage of our representation depends on how large is the difference between N and E.

The brief answer to the second question is that our representation has more overhead than NAX in all tasks related to managing the list of expressed attributes. This task is performed by three operators: crossover, generalizing and specializing. These three operators have a complexity of $O(E)$. The crossover operator has to look through the list of attributes of the second parent to identify the appropriate cut point, as explained in section 4. The generalizing and specializing operators have to compact or expand the lists held by of the representation (attributes and intervals) when an expressed attributes has been selected to be removed or a non expressed attribute has been selected to be expressed. Thus, we can say that the higher the number of expressed attributes, the higher the overhead of the representation.

From the simple analysis we made of gain and overhead of our representation we can see that a model of the relative performance of our representation against NAX should only take into account two variables: the number of attributes and the number of expressed attributes of each domain. We created a simple model, specified in equation 8, and fitted it[2] using run-time data that we collected from the experiments reported in previous sections. For each dataset, the speedup is computed as the run-time of NAX divided by the run-time of our representation using a probability of generalizing/specializing of 0.25, as this setting was in average the fastest setting of our representation. E is extracted from the rule sets resulting of each run of BioHEL, again from the setting that uses a probability of generalizing/specializing of 0.25, and averaged across all runs. The p-value of applying an F-statistic to the fitting of this model was less than $2.2e^{-16}$ ($a = 0.55282$, $b = -0.06518$). We can observe in the fitted value for a and b a reflection of the analysis we made previously in this section: the gain of our representation increases with the number of attributes in the domain and decreases with the number of expressed attributes. Figure 6 plots the fitted model overlapped to the empirical data. We also fitted models to compare the speedup of NAX and the other tested probabilities of the generalizing and specializing operators. The parameters of these other models were very similar to the ones for the 0.25

[1] Although NAX can deem some attributes irrelevant, it still need to check if these are relevant or not during the match process

[2] We have used the lm function of R to fit the model

probability. We will not plot these models as they overlap quite substantially with the one shown in figure 6.

$$Speedup = a \cdot \sqrt[3]{N} + b \cdot E \tag{8}$$

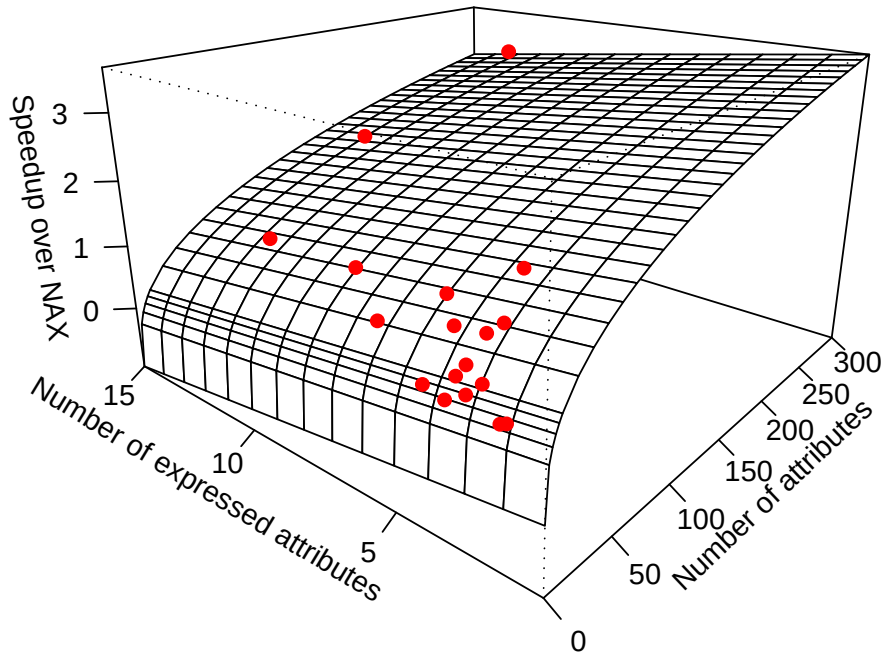


Fig. 6 Speedup between the NAX and the attribute list knowledge representation in relation to the total number of attributes and the number of expressed attributes of each domain

7 Conclusions and further work

In this paper we introduce a new representation intended to improve the efficiency of Evolutionary Learning (EL) methods on problems with high number of continuous attributes. The representation keeps a list of relevant attributes, and only evolve intervals to cover them, ignoring irrelevant ones. In this way the representation is much faster as it does not need to perform match operations on irrelevant attributes. Moreover, the exploration takes place only on the attributes that have shown to be relevant to the problem, potentially leading to a much better learning process.

We tested the representation integrated in BioHEL, a recent EL method applying the Iterative Rule Learning approach, and we compared its performance against the knowledge representation of the NAX system, also designed for efficiency purposes. Our test suite included many standard datasets

from the UCI repository, to verify if the representation was able to learn properly, and also two large-scale bioinformatics datasets with high number of attributes.

Our results show that the representation has competent performance when compared to NAX (even learning better in some datasets) and also manages to substantially reduce the run-time of the system. In the large datasets this reduction is very considerable, having run-times many hours shorter than the NAX representation. These characteristics of the representation will allow us to tackle much larger datasets that before we could not afford to solve using EL techniques. The comparison against two standard machine learning techniques revealed that our system significantly performed them, specially in datasets with a high number of attributes.

In future work, we would test this representation integrated within both Michigan and Pittsburgh approaches of LCS. We hope to investigate whether further tuning of the representation, by e.g. testing other settings of its parameters, leads to better and faster learning process. It would also be interesting to study alternative initialization policies, alternative policies of application of the specialize and generalize operators as e.g. local search methods, as well as evaluating the interactions between this representation and other efficiency enhancement techniques. Finally, we would like to study how can we extend the representation to also deal with nominal attributes, specially having in mind the GABIL (De Jong and Spears, 1991) representation, used previously in bioinformatics datasets (Bacardit et al, 2006, 2007b; Stout et al, 2008).

Acknowledgments

We would like to thank Michael Stout and Jonathan D. Hirst for their collaboration. We are grateful for the use of the University of Nottingham's High Performance Computer. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grants GR/T07534/01 and EP/E017215/1.

References

Bacardit J (2004) Pittsburgh genetics-based machine learning in the data mining era: Representations, generalization, and run-time. PhD thesis, Ramon Llull University, Barcelona, Spain

Bacardit J (2005) Analysis of the initialization stage of a pittsburgh approach learning classifier system. In: GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference, ACM Press, vol 2, pp 1843–1850

Bacardit J, Butz MV (2007) Data mining in learning classifier systems: Comparing xcs with gassist. In: Advances at the frontier of Learning Classifier Systems, Springer-Verlag, pp 282–290, DOI http://dx.doi.org/10.1007/978-3-540-71231-2_19

Bacardit J, Krasnogor N (2006a) Biohel: Bioinformatics-oriented hierarchical evolutionary learning. Nottingham eprints, University of Nottingham

Bacardit J, Krasnogor N (2006b) Empirical evaluation of ensemble techniques for a pittsburgh learning classifier system. In: Ninth International Workshop on Learning Classifier Systems (IWLCS 2006), Springer, Lecture Notes in Artificial Intelligenge, URL `http://www.asap.cs.nott.ac.uk/publications/pdf/iwlcs2006.pdf`, to appear

Bacardit J, Goldberg D, Butz M, Llorà X, Garrell JM (2004) Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In: Parallel Problem Solving from Nature - PPSN 2004, Springer-Verlag, LNCS 3242, pp 1021–1031

Bacardit J, Stout M, Krasnogor N, Hirst JD, Blazewicz J (2006) Coordination number prediction using learning classifier systems: performance and interpretability. In: GECCO '06:

Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press, pp 247–254

Bacardit J, Goldberg DE, Butz MV (2007a) Improving the performance of a pittsburgh learning classifier system using a default rule. In: Learning Classifier Systems, Revised Selected Papers of the International Workshop on Learning Classifier Systems 2003-2005, Springer-Verlag, LNCS 4399, pp 291–307

Bacardit J, Stout M, Hirst JD, Sastry K, Llorà X, Krasnogor N (2007b) Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press, New York, NY, USA, pp 346–353, DOI http://doi.acm.org/10.1145/1276958.1277033

Bernadó E, Llorà X, Garrell JM (2001) XCS and GALE: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In: Fourth International Workshop on Learning Classifier Systems - IWLCS-2001, pp 337–341

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

Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140

Butz MV (2006) Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design, Studies in Fuzziness and Soft Computing, vol 109. Springer

Cantu-Paz E, Kamath C (2003) Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(1):54–68

Corcoran AL, Sen S (1994) Using real-valued genetic algorithms to evolve rule sets for classification. In: Proceedings of the IEEE Conference on Evolutionary Computation, IEEE Press, pp 120–124, URL citeseer.nj.nec.com/corcoran94using.html

Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific

Cuff JA, Barton GJ (1999) Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins 34:508–519

De Jong KA, Spears WM (1991) Learning concept classification rules using genetic algorithms. In: Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp 651–656

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Divina F, Marchiori E (2005) Handling continuous attributes in an evolutionary inductive learner. IEEE Transactions on Evolutionary Computation 9(1):31–43

Divina F, Keijzer M, Marchiori E (2003) A method for handling numerical attributes in GA-based inductive concept learners. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Springer-Verlag, pp 898–908

Freitas AA (2002) Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag

Fürnkranz J (1999) Separate-and-conquer rule learning. Artificial Intelligence Review 13(1):3–54, URL citeseer.ist.psu.edu/26490.html

Giráldez R, Aguilar-Ruiz J, Riquelme J (2003) Natural coding: A more efficient representation for evolutionary learning. In: GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference, Springer-Verlag, pp 979–990

Giráldez R, Aguilar-Ruiz JS, Santos JCR (2005) Knowledge-based fast evaluation for evolutionary learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C 35(2):254–261

Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182, URL http://portal.acm.org/citation.cfm?id=944968

John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, pp 338–345, URL citeseer.ist.psu.edu/john95estimating.html

Llorà X (2008) Personal communication

Llorà X, Garrell JM (2001) Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Third Genetic and Evolutionary Computation Conference, Morgan Kaufmann, pp 461–468

Llorà X, Sastry K (2006) Fast rule matching for learning classifier systems via vector instructions. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press, New York, NY, USA, pp 1513–1520, DOI http://doi.acm.org/10.1145/1143997.1144244

Llorà X, Priya A, Bhargava R (2008) Observer-invariant histopathology using genetics-based machine learning. Natural Computing, Special issue on Learning Classifier Systems p in press

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann
Rissanen J (1978) Modeling by shortest data description. Automatica vol. 14:465–471
Ruiz R (2007) New heuristics in feature selection for high dimensional data. AI Commun 20(2):129–131
Stone C, Bull L (2003) For real! XCS with continuous-valued inputs. Evolutionary Computation Journal 11(3):298–336
Stout M, Bacardit J, Hirst JD, Krasnogor N (2008) Prediction of recursive convex hull class assignments for protein residues. Bioinformatics 24(7):916–923
Vafaie H, De Jong KA (1992) Genetic algorithms as a tool for feature selection in machine learning. In: Proceeding of the 4th International Conference on Tools with Artificial Intelligence, pp 200–203
Venturini G (1993) Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In: Brazdil PB (ed) Machine Learning: ECML-93 - Proc. of the European Conference on Machine Learning, Springer-Verlag, Berlin, Heidelberg, pp 280–296
Wilson SW (1995) Classifier fitness based on accuracy. Evolutionary Computation 3(2):149–175
Wilson SW (1999) Get real! XCS with continuous-valued inputs. In: Booker L, Forrest S, Mitchell M, Riolo RL (eds) Festschrift in Honor of John H. Holland, Center for the Study of Complex Systems, pp 111–121, URL `citeseer.nj.nec.com/233869.html`
Witten IH, Frank E (2000) Data Mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann
Wood MJ, Hirst JD (2005) Protein secondary structure prediction with dihedral angles. Proteins 59:476 – 481
Yang J, Honavar VG (1998) Feature subset selection using a genetic algorithm. IEEE Intelligent Systems 13(2):44–49, DOI http://dx.doi.org/10.1109/5254.671091