Constructive Induction and Genetic Algorithms for Learning Concepts with Complex Interaction

Leila Shila Shafti
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco
Madrid E–28049, Spain
leila.shafti@uam.es

Eduardo Pérez Pérez
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco
Madrid E–28049, Spain
eduardo.perez@uam.es

ABSTRACT

Constructive Induction is the process of transforming the original representation of hard concepts with complex interaction into a representation that highlights regularities. Most Constructive Induction methods apply a greedy strategy to find interacting attributes and then construct functions over them. This approach fails when complex interaction exists among attributes and the search space has high variation. In this paper, we illustrate the importance of applying Genetic Algorithms as a global search strategy for these methods and present MFE2/GA[1], while comparing it with other GA-based Constructive Induction methods. We empirically analyze our Genetic Algorithm's operators and compare MFE2/GA with greedy-based methods. We also performed experiments to evaluate the presented method when concept has attributes participating in more than one complex interaction. In experiments that are conducted, MFE2/GA successfully finds interacting attributes and constructs functions to represent interactions. Results show the advantage of using Genetic Algorithms for Constructive Induction when compared with greedy-based methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Induction*; H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms

Algorithms, Design, Experimentation

Keywords

Attribute interaction, constructive induction, feature construction, feature selection, genetic algorithms, shared attributes

[1]MFE2/GA is a modification to our previous method introduced in[27]

GECCO'05, June 25–29, 2005, Washington, DC, USA.

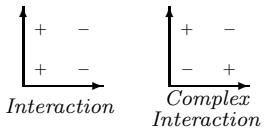

Figure 1: Interaction and complex interaction in instance space

1. INTRODUCTION

Similarity-Based Learning methods (SBL) learn concepts by discovering similarities. They assume that cases belonging to the same class are located close to each other in the instance space, defined by original attributes. They achieve high accuracy when the data representation of the concept is good enough to maintain the closeness of instances of the same class. For this reason most machine learning methods attain high accuracy on many artificial and real-world domains such as those provided in Irvine database [3, 10].

Hard concepts with complex interaction are difficult to be learned by an SBL. *Interaction* means the relation between one attribute and the target concept depends on another attribute. When the dependency is not constant for all values of the other attribute, the interaction is *complex* [20]. Figure 1 shows an example of interaction and complex interaction between two attributes in instance space. The complex interaction has been seen in real-world domains such as protein secondary structure [22]. Due to interaction, each class is scattered through the space, and therefore, regularities are hidden to the learner. The interaction problem arises when shortage of domain knowledge exists and only low-level primitive attributes are available to represent data.

Constructive induction (CI) methods have been introduced to ease the attribute interaction problem. Their goal is to automatically transform the original representation space of hard concepts into a new one, where the regularity is more apparent [1, 5]. This goal is achieved by constructing *new* features from the given attribute set, to abstract the interaction among several attributes into a new one. The new feature outlines the interaction and makes it apparent to the learner.

Most CI methods apply a greedy search to find new features. Due to the attributes interaction, the search space for

constructing new features has more variation, and therefore, a greedy method may find a local optimal solution. Recent works on problems with interaction [6, 27] show that a global search strategy such as Genetic Algorithms (GA) is more likely to be successful in searching through the intractable and complicated search space [16]. Moreover, GA provides the ability to construct and evaluate several features as one single individual. Evaluating a combination of features is essential for the success of a CI method when complex interactions among several subsets of attributes exist.

This paper highlights the importance of applying GA as a global search strategy for CI methods, and introduces a new GA-based CI method called MFE2/GA. Unlike other GA-based CI methods, MFE2/GA benefits from the use of a non-algebraic form of representing new features. As we showed in [26] this form of representation reduces the difficulty of constructing complex features.

Next section focuses on the shortcoming of greedy-based CI methods when high interaction exists among attributes. In Sect. 3 some of recently introduced GA-based CI methods are reviewed. Sect. 4 presents MFE2/GA, our new GA-based CI method. Experiments in Sect. 5 show that the new method successfully finds and highlights interactions among attributes and outperforms greedy-based CI methods.

2. THE IMPORTANCE OF GA FOR CI

As mentioned in previous section, the goal of CI is to abstract the interaction among attributes into new features to highlight regularities. This goal is usually achieved by constructing functions defined over interacting attributes. Therefore, a CI method needs to find subset of interacting attributes and define a function over this subset. For this purpose some CI methods such as Fringe, SymFringe [18, 19] and DCFringe [30] make use of hypotheses previously generated by a similarity based learner. These hypothesis-driven methods strongly depend on the generated hypothesis. When the concept is complex, because of the high interaction among attributes an SBL cannot generate a useful hypothesis, and consequently, the new features constructed using such hypothesis will be meaningless.

Other CI methods such as GALA [12], LFC [24], and MRP [21] apply a greedy strategy for selecting attributes and constructing new features. Attributes are considered one by one as candidates for inclusion in the current feature under construction. When high interaction exists among attributes, each attribute by itself does not give enough information about the concept and therefore, may be considered as an irrelevant attribute by a greedy CI method [6, 7, 20]. In order to see the interaction among attributes and construct a feature that highlights this interaction, several attributes are needed to be evaluated and selected at a time. So we need to search the space of all possible combination of attributes. However, this space grows exponentially with the number of attributes and has many local optima; thus, it becomes more complex to be explored. A CI method needs a search strategy that eventually finds the global optimal solution in such a complex search space. A greedy search approach may be suitable only when the variation of the search space is not high, otherwise it will find the local optimum.

Another problem of the greedy CI methods is that because of the greedy strategy of generating new features, construction of each feature depends on the feature previously constructed. Since the search space of concepts with complex interaction has high variation and several local optima the greedy strategy of constructing features may lead to a local optimal solution.

As an alternative to greedy methods, a global search such as evolutionary algorithms can be used for complex concepts. Evolutionary algorithms are theoretically and empirically proved to provide a robust search in complex spaces [8, 9]. Genetic Algorithms (GA) have been applied successfully to a variety of learning problems as well as to other tasks such as solving optimization problems. The reason for their success is that these methods search in intractable search space by retaining a balance between the exploitation of the best solution and the exploration of the search space [16].

In addition to the global search property, GA provides the ability to evaluate several attributes or constructed features as one single individual. This is important for a CI method when complex interaction exists, since each attribute or feature alone can be evaluated as an irrelevant one.

Therefore, if a proper representation language, GA operators and fitness function are provided, a GA-based CI method has the potential to generate useful features.

3. RELATED WORKS

Recently, new CI methods based on genetics search strategy have been introduced. Their success in achieving higher accuracy proves the advantage of GA over greedy searches. In this section some of these methods are reviewed and evaluated. These methods are preprocessing procedures. When the process is finished the new features are added to or used instead of the original set of attributes, and then data represented by the new set is given to a learner.

Most GA-based CI methods such as GCI [2], GPCI [11], Gabret [29], GAP [28] and the GA approach of Otero [17], consider individuals as algebraic expressions represented by Koza's parse trees [13] with operators in internal nodes and attributes in leaves. Among these, GPCI and Otero's approach have a set of predefined operators while the others apply domain specific operators. Unlike these methods, Larsen [14] and Ritthoff [25] do not use parse trees for constructing features. Larson applies the X-of-N representation of functions introduced by Zheng for a greedy CI [31]; so, this method also uses the fix predefined X-of-N operator. In Ritthoff's method each individual is a subset of attributes and functions represented by algebraic expressions. Genetic operators are not applied over segments of function. Functions are constructed arbitrarily by a new unary operator from the set of attributes in the individual and a set of predefined operators.

All these methods apply an *algebraic representation* [26] for constructing features, using some simple *predefined operators* or *domain specific operators*. Simple predefined operators make the method applicable to a wide range of problems. However, a complex feature is required to capture and encapsulate the interaction using simple operators. Domain specific operators reduce the complexity of feature construction. Nevertheless, specifying these operators properly cannot be performed without any prior information about the target concept. In Sect. 4 we see how *non-algebraic representation* in contrast to algebraic representation could be successfully applied to CI.

Features are evaluated differently by each of these methods. Each individual in GPCI, Otero's and Larsen's meth-

ods, and GCI represents a new feature or function. The first three methods use an Entropy-based fitness function [23] evaluating features individually. GCI uses a learner for evaluating the fitness of the function. Therefore, the fitness is measured considering the function in combination with original attributes. Gabret also uses a learner for fitness evaluation, but applies a different strategy. It consists of two independently selectable genetic-based modules, feature selection and feature construction. In the feature selection module, individuals are bit-strings representing subsets of attributes in current set. In feature construction module, individuals are parse trees representing sets of features and attributes. Therefore, in both modules a set of attributes and functions is evaluated as a whole. GAP applies a similar strategy. Ritthoff also uses a learner for measuring fitness, and evaluates a set of attributes and constructed features as a whole.

Evaluating a *combination* of features as an individual is useful for a CI, when high interaction exists as, each attribute or constructed feature alone may not be evaluated correctly. Also, using a learner as fitness function has the advantage of evaluating constructed functions in combination with other attributes. These methods add the new feature to the list of attributes and update data, then apply a learner like C4.5 [23] to measure the accuracy. However, this kind of evaluation, called *hypothesis-driven* evaluation, relies on the performance of the learner. Therefore, if the learner assigns incorrectly a low or high fitness value to individuals, it will guide the algorithm to a local solution. Moreover, performing a learning system for evaluating each individual increases the execution time of GA. A *data-driven* evaluation formula like entropy is preferred. This approach only depends on data and, therefore, is more reliable than a hypothesis-driven approach for guiding GA. In addition, computation time of fitness function is very important for GA performance, since this function is called for every individual during many generations. Experiments in [26] shows the advantage of a data-driven fitness function over a hypothesis-driven one.

It is important to note that GPCI is different from the other methods as it applies a divide and conquer strategy [19] to produce more than one new feature. After each performance of GA, the best individual, as the new feature, is used for splitting data. Then a new and independent GA is performed for each division. At the end of the procedure, one feature for each splitting of data exists which are added to original set of attributes. In fact GPCI uses GA for constructing each single feature but the whole process of constructing a set of features is greedy. The construction of each feature depends on the feature previously constructed. If the previous feature is not good enough it may misguide the whole process toward a local optimal solution.

Finally, all these methods modify the simple GA to be applied to a particular problem of learning. These modifications push simple GA away from their theoretical bases. These methods have many factors, which are difficult to be analyzed by schema theorem that will help the method to approach to the solution. The most important factor is the representation of the constructed features.

4. MFE2/GA

MFE2/GA (Multi-Feature Extraction using GA) aims to construct new features to highlight interactions. As explained in Sect. 2 the search space for finding relevant attributes and constructing functions is large and with high variation when complex interactions exist in concept. Therefore, a global search strategy such as GA is needed to successfully find the optimal solution.

We applied GA to search through the space of different combination of attributes subsets and functions defined over them. GA receives training data with original attributes set and finds subsets of interacting attributes and features representing the interaction. When GA finishes, the new features are added to the original set of attributes and the new representation of data is given to a standard learner for learning. The current version of the method assumes that all continuous attributes have been converted to nominal attributes before running the system.

4.1 Representation Language

For constructing new features, two tasks are needed to be performed: finding the subsets of interacting attributes and generating a function defined over each subset. Our individuals are sets of subsets of primitive attributes such as $Ind = \langle S_1, S_2, \ldots, S_k \rangle$ where $S_i \subset S$, $S_i \neq \emptyset$, and S is the set of original attributes.

Subsets in individuals are represented by bit-strings of length n, where n is the number of original attributes; each bit showing the presence or absence of the attribute in the subset. Therefore, each individual is a bit-string of length $k.n$ $(k > 0)$ such as $Ind = \langle b_1, \ldots, b_n : b'_1, \ldots, b'_n : b''_1, \ldots, b''_n : \ldots \rangle$. Since each individual has different number of subsets, the length of individuals is variable. To avoid unnecessary growth of individuals, the number of subsets in individuals is made limited so that $k \leq 5$.

Each subset in individual is associated with a function that is extracted from data. Thus each individual is actually representing a set of functions such as $\{F_1, F_2, \ldots, F_k\}$, where F_i is a function defined over S_i. It is important to note that during mutation and crossover if a subset is changed, the associated function is also changed since a new F'_i is extracted for the new subset S'_i in the offspring.

The function F_i created for any given subset in an individual uses a non-algebraic form of representation. By contrast to algebraic representation, non-algebraic form refers to a representation language that does not apply any form of algebraic operators. For example, for a Boolean attribute set $\{X_1, X_2\}$ an algebraic feature like $(X_1 \wedge \overline{X}_2) \vee (\overline{X}_1 \wedge X_2)$ can be represented by a non-algebraic feature such as $\langle 0110 \rangle$, where the j^{th} element in $\langle 0110 \rangle$ represents the outcome of the function for j^{th} combination of attributes X_1 and X_2 according to the following truth table:

X_1	X_2	f
0	0	0
0	1	1
1	0	1
1	1	0

As discussed in Sect. 3, most GA-based CI methods apply algebraic form of representing functions. We showed in [26] that a complex algebraic expression is required to capture and encapsulate the interaction into a feature, while non-algebraic representation reduces the difficulty of constructing complex features. For this reason, and in spite of their use of GA search strategy, some CI methods fail when a high-order complex interaction exists among attributes.

The function F_i for any given subset $S_i = \{X_{i1}, \ldots, X_{im}\}$

is defined by assigning Boolean class labels to all the tuples in the Cartesian product $X_{i1} \times \ldots \times X_{im}$. The class assigned to each tuple t depends on the training samples that *match* the tuple, that is, the training samples whose values for attributes in S_i are equal to the corresponding values in tuple t. More precisely, the class assigned depends on the class labels of all those training samples matching the tuple, as discussed case by case next:

Case 1. If there are no training samples matching t, a class label is assigned to $F_i(t)$ stochastically, according to the class distribution in the training data.

Case 2. If all training samples matching t belong to the same class, this is the class assigned to $F_i(t)$.

Case 3. If there is a mixture of classes in the samples matching t, the class assigned to $F_i(t)$ depends on the numbers of tuples labeled by Case 2 as positive and negative, p_2 and n_2 respectively. If $p_2 > n_2$, the negative class is assigned; and otherwise, the positive class is assigned.

As discussed in the next section, the GA's fitness evaluation is applied to individuals composed of several F_i, each defined over a subset S_i. Thus, each individual encapsulates several interactions into features, and this allows GA to simultaneously construct and evaluate features, which turns out to be essential when several high-order interactions exist in data.

4.2 Fitness Function

We mentioned in Sect. 3 that some GA-based CI methods apply a hypothesis-driven fitness function, and therefore, their performance relies on learner they apply. Moreover, the computation time of a proper data-driven fitness function is less than a hypothesis-driven one. Therefore, we applied a data-driven fitness function.

When features are extracted, for each $Ind = \langle S_1, \ldots, S_k \rangle$, data are projected onto the set of new features $\{F_1, \ldots, F_k\}$ and the goodness of the individual is evaluated by the following formula:

$$Fitness(Ind) = \frac{min(|\pi^+ - \pi^-|, |\pi^- - \pi^+|)k + \|\pi^+ \cap \pi^-\|(k+1)}{r(k+1)} + \frac{\sum |S_i|}{k|S|} \qquad (1)$$

where π^+ is set of positive tuples and π^- is set of negative tuples obtained by projecting data into $\{F_1, \ldots, F_k\}$, r is the total number of tuples in training data, the single bars $|z|$ denote the number of attributes (or tuples) in subset (or relation) z, and the double bars $\|p\|$ denote the number of examples in training data that match with the tuples in relation p. The objective of GA is to minimize the value of $Fitness(Ind)$. The first term in this formula estimates how good is the set of new features for classifying data. It is divided by $k+1$ to favor individuals with larger number of subsets. The aim is to prefer several simple features to few complex features. The complexity of features is evaluated in the last term by measuring the fraction of attributes participating in constructing features.

To reduce over-fitting, we use 90% of training data for generating functions and all training data for fitness evaluation. Aside from that, the empirical evaluation of the system will be based on unseen data (see Sect. 5).

Mutation in Attributes Level (Mutation Type-1)
Parent1 = ⟨1001001**0**:01**0**10100:0001**0**111⟩
Child1 = ⟨1101001**1**:01**1**10100:0000**1**111⟩

Mutation in Subsets Level (Mutation Type-2)
Parent2 = $\langle S_1, S_2, S_3, S_4, S_5 \rangle$
Child2 = $\langle S_1, S_2', S_3, S_4, S_5 \rangle$

Crossover in Attributes Level (Crossover Type-1)
Parent3 = ⟨1001|0010:010|10100:00010111⟩
Parent4 = ⟨0010|1011:10010001:111|01100⟩
Child3 = ⟨10011011:10010001:11110100:00010111⟩
Child4 = ⟨00100010:01001100⟩

Crossover in Subsets Level (Crossover Type-2)
Parent5 = $\langle S_{11}, S_{12} \rangle$ Mask1= ⟨10⟩
Parent6 = $\langle S_{21}, S_{22}, S_{23}, S_{24}, S_{25} \rangle$ Mask2= ⟨01101⟩
Child5 = $\langle S_{11}, S_{22}, S_{23}, S_{25} \rangle$
Child6 = $\langle S_{12}, S_{21}, S_{24} \rangle$

Figure 2: GA operators

4.3 GA Operators

Our objective is to generate different subsets of attributes with their associated functions and combine them to eventually find the group of functions defined over subsets of interacting attributes. Operators are applied in two levels: *attributes level* to generate different subsets and features; and, *subsets level* to make different combination of subsets and features as illustrated by examples in Fig. 2.

Mutation in attributes level, Mutation Type-1, considers the individual as a bit-string of size $k.n$ where k is the number of subsets and n is the number of original attributes. The traditional mutation is applied over the bit-string to flip bits of the string. This operation aims to introduce a new subset by a tiny change in the previously generated subset in individual.

The Mutation Type-2, that is mutation in subsets level, considers an individual as a sequence of subsets and replaces one subset by a new generated subset. Therefore, this operator introduces more variability into the population and gives more diversity comparing to Mutation Type-1.

Similarly, two types of crossover operators are used in this method. Crossover Type-1, that is crossover in attributes level, exchanges segments of individuals considering them as bit-strings. It applies the classical two-point crossover. The two crossing points in the first parent are selected randomly. On the second parent, the crossing points are selected randomly, subject to the restriction that they must have the same distance from the subsets boundary in bit-string representation as they had in the first parent [4]. This operator may change the length of the individual. But we impose the limitation of $k \leq 5$; otherwise, new crossing points are selected until the produced offspring have $k \leq 5$ subsets. Depending on where the crossing points are situated this operator may generate new subsets from subsets of parents and/or recombine subsets.

Crossover in subsets level, Crossover Type-2, aims to generate different combinations of subsets by exchanging subsets of parents. It considers individuals as sequence of subsets and performs uniform crossover. Two crossover masks are generated randomly to define the cutting points. This operator may change the length of the individuals and therefore has the restriction of $k \leq 5$, same as above. Crossover Type-2 only recombines subsets and does not generate any new subset.

It is important to note that GA operators are performed over individuals representing combinations of subsets of attributes. Thus, changing an individual implies modifying attributes subsets and their combination. Furthermore, since each attributes subset determines a function extracted from data (as explained in Sect. 4.1), a modified subset of attributes means a new constructed feature. In the same way a modified combination of subsets of attributes means a new combination of functions. Therefore, the application of GA operators to MFE2/GA's population produces the evolution of construed features and their combination.

Another important note is that the bit-string representation of individuals in this method provides the facility to apply classical GA operators, and therefore, maintains theoretical bases of GA.

5. EXPERIMENTS

We empirically analyze the effect of our main operators, that are crossover in attributes level and in subsets level (see Sect. 4.3), on performance of GA. Also MFE2/GA is compared and evaluated with greedy-based method, MRP [21]. Furthermore, the performance of the presented method is evaluated over hard concepts with attributes participating in more than one complex interaction. For implementing GA, we used PGAPack Library [15] with default parameters, except those indicated in Table 1. The stopping rule is to reach the maximum number of iterations limit or the maximum number of iterations in which no change in the best evaluation is allowed. To reduce over-fitting we used 90% of training data for constructing functions and all training data for evaluating the constructed feature using Formula 1 as fitness function.

5.1 Evaluating Crossover Operators

The aim is to evaluate the importance of our two crossover operators for converging GA to optimal solution. For each problem, MFE2/GA was run; once with crossover type-1, once with crossover type-2 and once more with both crossover operators. When both crossovers are permitted, each time that individuals are selected for reproduction, the type of crossover operator is selected for application by flipping a coin.

Synthetic problems were used for these experiments. All problems are concepts with complex interaction that can be represented by several smaller interactions. See Appendix A for definition of these concepts.

Figures 3 and 4 show the average results of 20 independent runs of GA for each concept. For each run, 5% of shuffled data were used for training and the rest for final evaluation. When GA was finished the new features were added to original set of attributes and data were updated. Then the 5% training data were used for learning by C4.5 and the accuracy was measured over 95% unseen data. The horizontal axis in figures represents concepts. In Figures 3 the vertical axis shows the average accuracy of running learner C4.5 on updated data. This axis in Figure 4 shows the average number of generations needed in order to achieve the result for each concept.

Table 1: GA's modified parameters

GA Parameter	New Value
Population Size	100
Mutation Probability	0.01
Num. of Strings to be Replaced	90
Max Iteration	350
Max No Change Iteration	100

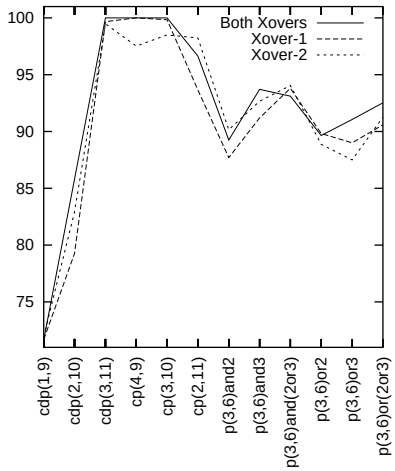

Figure 3: Comparing the use of crossover operators - Accuracy

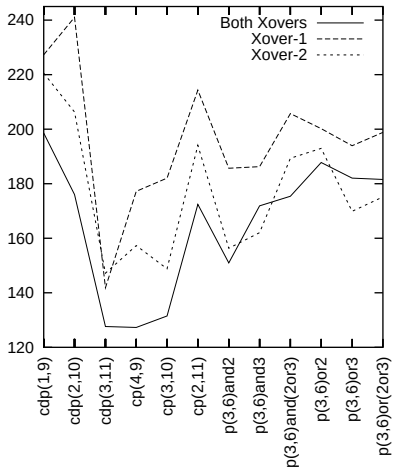

Figure 4: Comparing the use of crossover operators - Number of generations

Figure 3 shows that using crossover in attributes level or in subsets level alone caused GA to fail sometimes; and therefore, gave a lower average accuracy. The application of both crossover operators helped GA to converge to optimal solution and allowed a higher accuracy. Therefore, both crossover operators are required to achieve better accuracy.

It can be seen from Figure 4 that, in addition to resulting better accuracy, using two operators together helped GA to terminate earlier.

Results in Figures 3 and 4 support our claims that two crossover operators defined in Sect. 4.3 complement each other to accelerate the convergence of MFE2/GA to the optimal solution.

Table 2: The average accuracies over complex concepts

Concept	Num. of Relevant Atts	MRP	MFE2/GA + C4.5
cp(4,9)	6	99.0 (1.6)	**100 (0.0)**
cp(3,10)	8	89.9 (1.2)	**100 (0.0)**
cp(2,11)	10	91.7 (5.7)	**96.7 (6.1)**
cdp(3,11)	6	97.3 (3.9)	**100 (0.0)**
cdp(2,10)	9	**92.0 (6.5)**	85.8 (8.6)
cdp(1,9)	12	**81.3 (3.0)**	71.7 (3.8)
P(3,6)∧(2)	10	90.5 (4.5)	93.1 (5.7)
P(3,6)∧(3)	10	87.6 (5.4)	**93.8 (5.2)**
P(3,6)∧(3or2)	10	79.0 (2.3)	**89.3 (5.7)**
P(3,6)∨(2)	10	**97.1 (2.9)**	89.7 (5.5)
P(3,6)∨(3)	10	95.9 (4.4)	91.1 (7.6)
P(3,6)∨(2or3)	10	80.4 (6.0)	**92.5 (5.7)**

5.2 Comparing with a Greedy Method

In this section, MFE2/GA is compared with MRP [21] which is a greedy-based CI method. MRP was selected because of its similarity to MFE2/GA in using non-algebraic representation of new features. MRP represents features by sets of tuples using relational projection. However it applies a greedy hill climbing method for selecting attributes and constructing features. Nevertheless, MRP has shown good performance on concepts with complex interactions when compared to non GA-based CI methods such as Fringe, Grove, Greedy3 [19] and LFC [24].

Experiments were run over synthetic problems with more than one interaction (see Appendix A). These problems were used as prototypes to exemplify complex interaction in real-world hard problems. Therefore, similar results are expected for real-world problems, where the main difficulty is complex interaction. All concepts have complex interaction among attributes that can be represented by several smaller interactions.

For each concept, MFE2/GA was run 20 times independently, using 5% of shuffled data for training and the rest for final evaluation. When MFE2/GA was finished, its performance was evaluated by the accuracy of C4.5 on modified data after adding constructed features, using the same 5% data as training data and 95% unseen data as test data. Table 2 gives a summary of MFE2/GA's accuracy over 20 runs and its comparison with MRP. Numbers between parentheses indicate standard deviation. Bold means that the difference is statistically significant ($\alpha = 0.02$).

It can be seen that in most cases, MFE2/GA significantly outperforms MRP. When number of interacting attributes grows, the concept becomes more complex to be learned. This additional complexity, along with MRP's greedy strategy, makes this method to result in low accuracy. On the other hand, MFE2/GA uses a global search strategy to successfully break down the interaction over relevant attributes into two or more interactions over smaller subsets of attributes; and therefore, it gives better accuracy than MRP in most concepts of Table 2.

The synthetic concepts $cdp(i, j)$ illustrates well the different behaviors of MRP and MFE2/GA. Each of the concepts $cdp(1, 9)$, $cdp(2, 10)$ and $cdp(3, 11)$ involves three parity relations combined by simple interactions (conjunction and disjunction of parity). These concepts differ in the degree of parity involved (4, 3, and 2, respectively). More importantly, they also differ in the ratio of relevant attributes (12/12, 9/12, and 6/12, respectively). This affects MFE2/GA in a higher degree than it affects MRP due to differences between both systems' biases. In particular, considering $cdp(1, 9)$, MRP's focus on learning one single best relation guides learning toward $Parity(a_1, \ldots, a_4)$. As stated above, the interactions that combine $Parity(a_1, \ldots, a_4)$ with the other parity features in this concept are simple (conjunction and disjunction). So MRP easily finds its way toward learning $Parity(a_1, \ldots, a_4)$. Had it used only this single feature to classify unseen data, it would have obtained even higher accuracy than it does (up to 87%). Perhaps, due to overfitting, MRP's heuristic function does not allow the system to reach such theoretically best possible performance on this concept. However, MRP's bias gets it closer to the goal than MFE2/GA. MFE2/GA's bias is, in sense, opposite to MRP's. It focuses on learning multiple features at once to evaluate them in combination. This higher flexibility in searching a large and complex feature space makes the system more dependent on data quality (since features are extracted from training data). Therefore, MFE2/GA's more flexible search gets trapped in a local optimum, over-fitting data; whereas MRP is favored by its strong bias for *one best* relation, which in this case, does indeed exists and it is easy to find.

It is important to note that in all experiments MFE2/GA generated close approximations to represent the complex interaction by several sub-interactions. Indeed, in more than 80% of experiments MFE2/GA successfully found the exact subsets of interacting attributes and corresponding functions representing sub-interactions.

We have not done a detail study of MFE2/GA's execution time, since it is implemented as a prototype system, without any optimizing aspects and intended mainly to evaluate a new way of integrating GA into CI. However, as an indication, we report here MFE2/GA's overall running time for these experiments. The total real time to run 20 experiments for 12 concepts took 146 minutes on a Pentium 4, 2.26 GHz, with 384 MB of RAM; which is in average 36 seconds for each MFE2/GA run.

The use of synthetic concepts allowed an in-depth analysis of the system behavior before moving on to try to solve real-world problems with difficulties similar to those exemplified by these synthetic concepts. More studies are needed to evaluate how MFE2/GA will scale up for large concepts and real-word problems.

5.3 Concepts with Shared Attributed

The main advantage of MFE2/GA with non-algebraic representation of features is its capability of dealing with complex concepts when there are attributes participating in more than one interaction. However, despite using non-algebraic representation of features, MRP's greedy search strategy prevents the construction of useful features when there are shared attributes in several interactions. This limitation of MRP and other similar greedy-based methods is anticipated in [20].

A number of experiments were carried out to further evaluate this property of MFE2/GA using concepts with attributes participating in more than one interaction (see Appendix A). We compared MFE2/GA with C4.5 and C4.5-

Table 3: The average accuracies over complex concepts with relations on shared attributes

Concept	Num. of Relevant Atts	C4.5 Original Atts	C4.5Rules Original Atts	C4.5 Relevant Atts	C4.5Rules Relevant Atts	MFE2/GA + C4.5
$(P_4 \wedge P_4)$-of-6	6	72.2 (3.6)	70.9 (2.9)	87.9 (5.5)	94.4 (4.8)	**99.1 (1.6)**
$(P_6 \wedge P_6)$-of-8	8	72.2 (3.5)	69.0 (3.5)	73.2 (2.4)	69.7 (2.4)	**92.8 (5.7)**
$(P_3 \wedge P_3 \wedge P_3)$-of-6	6	87.4 (1.5)	85.1 (3.2)	91.7 (3.6)	96.4 (2.8)	**99.8 (0.7)**
$(P_4 \wedge P_4 \wedge P_4)$-of-6	6	87.5 (0.1)	84.2 (2.0)	90.6 (2.5)	95.8 (3.9)	**99.6 (0.7)**
$(P_4 \wedge P_4 \wedge P_4)$-of-8	8	87.5 (0.1)	84.2 (3.0)	85.4 (1.8)	85.4 (1.8)	**96.4 (1.8)**
$(P_6 \wedge P_6 \wedge P_6)$-of-8	8	86.5 (2.2)	83.9 (2.7)	87.0 (1.6)	86.2 (1.9)	**92.2 (3.3)**

Rules [23]. Table 3 shows the average accuracy after 20 independent runs over 5% shuffled data. Columns 3 and 4 show the result of C4.5 and C4.5Rules on original set of data. The results of C4.5 and C4.5Rules are also shown in columns 5 and 6 after forcing them to use only relevant attributes.

As it can be seen, in all cases MFE2/GA achieved significantly higher accuracy comparing to the others. This demonstrates the ability of this method to construct useful features when there are attributes participating in more than one interaction.

The results in column 6 show that if the system can be informed which attributes are relevant, C4.5Rules can slightly improve performance in most cases. However, MFE2/GA by discovering interacting attributes and constructing new functions, considerably facilitates learning these concepts. Due to the complex interactions of these concepts, selection of interacting attributes alone does not facilitates learning. This implies that a preprocessing feature selection alone and without feature construction cannot help a learner when a high interaction exists among attributes.

6. CONCLUSION

In this paper the problem of greedy-based CI methods was discussed. It was explained that the search space for finding interacting attributes and constructing functions is large and has high variation when complex interaction exists in concept. Therefore, a global search such as GA is more likely to be successful.

It was also argued that CI requires constructing and evaluating several features together, since each feature by itself may not be evaluated correctly due to the complex interaction in data. GA allows simultaneously constructing and evaluating several features represented as a single individual, which turns out to be essential for concepts with high complex interaction.

Considering these, MFE2/GA, a GA-based CI method, was presented. The simple bit-string representation of individuals as subsets of attributes, along with non-algebraic representation of corresponding functions defined over such attribute subset, allowed the application of classical GA operators. A new data-driven fitness function was used, which was faster than a hypothesis-driven fitness function. Two types of mutation and crossover operators along with the data-driven fitness function in this method showed a good behavior in converging GA to optimal solution. Representing several subsets of attributes and corresponding functions as one individual made MFE2/GA capable of dealing with concepts with attributes participating in more than one interaction. Experiments showed the good performance of this method when compared with greedy-based methods on concepts where CI was important due to complex interactions.

7. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Science and Technology, under grant number TIC2002-1948.

8. REFERENCES

[1] D. W. Aha. Incremental constructive induction: An instance-based approach. In *Proc. of the Eighth International Workshop on Machine Learning*, pages 117–121, Evanston, Illinois, 1991. Morgan Kaufmann.

[2] H. Bensusan and I. Kuscu. Constructive induction using genetic programming. In T. Fogarty and G. Venturini, editors, *Proc. of Evolutionary Computing and Machine Learning Workshop (ICML'96)*, Bari, Italy, July 1996.

[3] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[4] K. A. De Jong, W. M. Spears, and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.

[5] T. G. Dietterich and R. S. Michalski. Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence*, 16(3):257–294, July 1981.

[6] A. A. Freitas. Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review*, 16(3):177–199, November 2001.

[7] A. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., 2002.

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

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[10] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[11] Y. Hu. A genetic programming approach to constructive induction. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Proc. of the Third Annual Genetic Programming Conference*, pages 146–151, University of Wisconsin, Madison, Wisconsin, USA, July 1998. Morgan Kauffman.

[12] Y. Hu and D. F. Kibler. Generation of attributes for learning algorithms. In *Proc. of the Thirteenth National Conference on Artificial Intelligence*, pages 806–811. AAAI, The MIT Press, August 1996.

[13] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[14] O. Larsen, A. A. Freitas, and J. C. Nievola. Constructing X-of-N attributes with a genetic algorithm. In *Proc. of the Genetic and Evolutionary Computation Conference*, page 1268, San Francisco, July 2002. Morgan Kaufmann.

[15] D. Levine. Users guide to the PGAPack parallel genetic algorithm library. Technical Report 18, Argonne National Laboratory, 1996.

[16] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag New York, Inc., 1999.

[17] F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola. Genetic programming for attribute construction in data mining. In C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, editors, *Proc. of the Sixth European Conference in Genetic Programming*, volume 2610 of *Lecture Notes in Computer Science*, pages 384–393. Springer-Verlag, April 2003.

[18] G. Pagallo. *Adaptive Decision Tree Algorithms for Learning from Examples*. PhD thesis, University of California at Santa Cruz, 1990.

[19] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[20] E. Pérez. *Learning Despite Complex Interaction: An Approach Based on Relational Operators*. PhD thesis, niversity of Illinois, Urbana-Champaign, 1997.

[21] E. Pérez and L. A. Rendell. Using multidimensional projection to find relations. In *Proc. of the Twelfth International Conference on Machine Learning*, pages 447–455, Tahoe City, California, July 1995. Morgan Kaufmann.

[22] N. Qian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202(4):865–884, August 1988.

[23] R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[24] H. Ragavan and L. A. Rendell. Lookahead feature construction for learning hard concepts. In *Proc. of the Tenth International Conference on Machine Learning*, pages 252–259, University of Massachusetts, Amherst, MA, USA, June 1993. Morgan Kaufmann.

[25] O. Ritthoff, R. Klinkenberg, S. Fischer, and I. Mierswa. A hybrid approach to feature selection and generation using an evolutionary algorithm. In *UK Workshop on Computational Intelligence*, Birmingham, U.K., September 2002.

[26] L. S. Shafti and E. Pérez. Genetic approach to constructive induction based on non-algebraic feature representation. In M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *Proc. of the Fifth International Symposium on Intelligent Data Analysis*, Lecture Notes in Computer Science, pages 599–610. Springer-Verlag, August 2003.

[27] L. S. Shafti and E. Pérez. Machine learning by multi-feature extraction using genetic algorithms. In C. Lemaître, C. A. Reyes, and J. A. González, editors, *Proc. of the Ninth Ibero-American Conference on Artificial Intelligence - Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 246–255. Springer-Verlag, November 2004.

[28] M. G. Smith and L. Bull. Feature construction and selection using genetic programming and a genetic algorithm. In C. Ryan, T. Soule, M. Keijzer, E. P. K. Tsang, R. Poli, and E. Costa, editors, *Proc. of the sixth European Conference in Genetic Programming*, volume 2610 of *Lecture Notes in Computer Science*, pages 229–237, Essex, UK, April 2003. Springer-Verlag.

[29] H. Vafaie and K. DeJong. Feature space transformation using genetic algorithms. *IEEE Intelligent Systems*, 13(2):57–65, March-April 1998.

[30] D. S. Yang, L. A. Rendell, and G. Blix. A scheme for feature construction and a comparison of empirical methods. In *Proc. of the Twelfth International Joint Conference on Artificial Intelligence*, pages 699–704, Sydney, Australia, August 1991. Morgan Kaufmann.

[31] Z. Zheng. Constructing X-of-N attributes for decision tree learning. *Machine Learning*, 40(1):35–75, 2000.

APPENDIX

A. CONCEPTS DEFINITIONS

All concepts used for experiments are synthetic concepts defined over twelve Boolean attributes $a_1, \ldots, a_{12}$. The concepts used in Sec. 5.1 and 5.2 are as follows:

$cp(i,j)$	$=\text{Parity}(a_i,\ldots,a_6) \wedge \text{Parity}(a_7,\ldots,a_j)$
$cdp(i,j)$	$=\text{Parity}(a_i,\ldots,a_4) \wedge$ $(\text{Parity}(a_{\frac{i+j}{2}},\ldots,a_8) \vee \text{Parity}(a_j,\ldots,a_{12}))$
$P(3,6) \wedge (l)$	$=\text{Parity}(a_3,\ldots,a_6) \wedge$ exactly l attributes in $\{a_7,\ldots,a_{12}\}$ are true
$P(3,6) \vee (l)$	$=\text{Parity}(a_3,\ldots,a_6) \vee$ exactly l attributes in $\{a_7,\ldots,a_{12}\}$ are true

All above concepts have complex interaction among relevant attributes that can be represented by several smaller interactions.

We also defined the following particular complex concepts for experiments in Sec. 5.3:

$(P_4 \wedge P_4)$-of-6	$=\text{Parity}(a_1,\ldots,a_4) \wedge$ $\text{Parity}(a_3,\ldots,a_6)$
$(P_6 \wedge P_6)$-of-8	$=\text{Parity}(a_1,\ldots,a_6) \wedge$ $\text{Parity}(a_3,\ldots,a_8)$
$(P_3 \wedge P_3 \wedge P_3)$-of-6	$=\text{Parity}(a_1,\ldots,a_3) \wedge$ $\text{Parity}(a_3,\ldots,a_5) \wedge \text{Parity}(a_4,\ldots,a_6)$
$(P_4 \wedge P_4 \wedge P_4)$-of-6	$=\text{Parity}(a_1,\ldots,a_4) \wedge$ $\text{Parity}(a_2,\ldots,a_5) \wedge \text{Parity}(a_3,\ldots,a_6)$
$(P_4 \wedge P_4 \wedge P_4)$-of-8	$=\text{Parity}(a_1,\ldots,a_4) \wedge$ $\text{Parity}(a_3,\ldots,a_6) \wedge \text{Parity}(a_5,\ldots,a_8)$
$(P_6 \wedge P_6 \wedge P_6)$-of-8	$=\text{Parity}(a_1,\ldots,a_6) \wedge$ $\text{Parity}(a_2,\ldots,a_7) \wedge \text{Parity}(a_3,\ldots,a_8)$

The important characteristic of these concepts is that there are attributes participating in more than one interaction.