A genetic algorithm with sequential niching for discovering small-disjunct rules

Deborah R. Carvalho
Pontificia Universidade Católica do Paraná (PUCPR)
Postgraduate program in applied computer science
R. Imaculada Conceicao, 1155. Curitiba – PR
80215-901. Brazil

Universidade Tuiuti do Paraná (UTP)
Computer Science Dept.
Av. Comendador Franco, 1860. Curitiba-PR
80215-090 Brazil
deborah@ipnet.com.br

Alex A. Freitas
Pontificia Universidade Católica do Paraná (PUCPR)
Postgraduate program in applied computer science
R. Imaculada Conceicao, 1155. Curitiba – PR
80215-901. Brazil
alex@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex
Tel./Fax: (55) (41) 330-1669

Abstract

This work addresses the well-known classification task of data mining. In this context, small disjuncts are classification rules covering a small number of examples. One approach for coping with small disjuncts, proposed in our previous work, consists of using a decision-tree/genetic algorithm method. The basic idea is that examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm (C4.5), while examples belonging to small disjuncts are classified by a genetic algorithm (GA) designed for discovering small-disjunct rules. In this paper we follow this basic idea, but we propose a new GA which consists of several major modifications to the original GA used for coping with small disjuncts. The performance of the new GA is extensively evaluated by comparing it with two versions of C4.5, across several data sets, and with several different sizes of small disjuncts.

1 INTRODUCTION

This paper addresses the well-known classification task of data mining (Hand, 1997). In this task, the discovered knowledge is often expressed as a set of rules of the form: IF <conditions> THEN <prediction (class)>.

This knowledge representation has the advantage of being intuitively comprehensible for the user, and it is the kind of knowledge representation used in this paper.

From a logical viewpoint, typically the discovered rules are in disjunctive normal form, where each rule represents a disjunct and each rule condition represents a conjunct. A small disjunct can be defined as a rule which covers a small number of training examples (Holte et al., 1989).

In general rule induction algorithms have a bias that favors the discovery of large disjuncts, rather than small disjuncts. This preference is due to the belief that it is better to capture generalizations rather than specializations in the training set, since the latter are unlikely to be valid in the test set (Danyluk & Provost, 1993).

Hence, at first glance, small disjuncts are not important, since they tend to be error prone. However, small disjuncts are actually quite important in data mining. The main reason is that, even though each small disjunct covers a small number of examples, the set of all small disjuncts can cover a large number of examples. For instance (Danyluk & Provost, 1993) report a real-world application where small disjuncts cover roughly 50% of the training examples. In such cases we need to discover accurate small-disjunct rules in order to achieve a good classification accuracy rate.

One approach for coping with small disjuncts, proposed in our previou work (Carvalho & Freitas 2000a, 2000b), consists of using a decision-tree/genetic algorithm method. The basic idea is that examples belonging to large disjuncts are classified by rules produced by a decision-tree algorithm (C4.5), while examples belonging to small disjuncts are classified by a genetic algorithm (GA) designed for discovering small-disjunct rules.

In this paper we follow this basic idea, but we propose a new GA which consists of several major modifications to the original GA proposed for coping with small disjuncts.

The rest of this paper is organized as follows. Section 2 describes the hybrid decision tree/genetic algorithm method proposed in our previous work. This section assumes that the reader is familiar with decision trees, a well-known kind of data mining algorithm. Section 3 describes the new GA proposed in this paper for discovering small-disjunct rules. In that section we explain the motivation for the design of this new GA and describe in detail the major modifications that it introduces, by comparison with original GA used in our hybrid method. Section 4 reports the results of extensive experiments evaluating the performance of the proposed method. Finally, section 5 concludes the paper.

2 THE BASIC HYBRID DECISION-TREE / GENETIC-ALGORITHM METHOD FOR RULE DISCOVERY

We have previously proposed a hybrid method for rule discovery that combines decision trees and genetic algorithms (GAs) (Carvalho & Freitas, 2000a; 2000b). The basic idea is to use a decision-tree algorithm to classify examples belonging to large disjuncts and use a GA to discover rules classifying examples belonging to small disjuncts. Decision-tree algorithms have a bias towards generality that is well suited for large disjuncts, but not for small disjuncts. On the other hand, GAs are robust, flexible algorithms which tend to cope well with attribute interactions (Dhar et al, 2000), (Freitas, 2001; 2002), and can be more easily tailored for coping with small disjuncts.

The method discovers rules in two training phases. In the first phase it runs C4.5, a well-known decision tree induction algorithm (Quinlan, 1993). The induced, pruned tree is transformed into a set of rules (or disjuncts). Each of these rules is considered either as a small disjunct or as a "large" (non-small) disjunct, depending on whether or not its coverage (the number of examples covered by the rule) is smaller than or equal to a given threshold.

The second phase consists of using a GA to discover rules covering the examples belonging to small disjuncts. In the previous version of our method, each run of the GA discovers rules classifying examples belonging to a separated small disjunct. In this paper we introduce a major modification of this phase: all small disjuncts are grouped together into a single training set and given to the GA, so that a single run of the GA discovers rules classifying examples belonging to the total set of small-disjunct examples. This new approach (as well as the motivation for it) will be described in the next section.

Before we move to the next section, however, we review in the following the main characteristics of our previous GA (Carvalho & Freitas 2000a), hereafter called GA-Small (standing for GA with Small training set), in order to make this paper self-contained. Hereafter the new GA introduced in this paper will be called GA-Large-SN (standing for GA with Large training set and with Sequential Niching), since it not only uses a larger training set but also uses a sequential niching method, as will be described later.

In GA-Small each individual represents the antecedent (IF part) of a small-disjunct rule. The consequent (THEN part) of the rule, which specifies the predicted class, is not represented in the genome. Rather, it is fixed for a given GA-Small run, so that all individuals have the same rule consequent during all that run.

Each run of GA-Small discovers a single rule (the best individual of the last generation) predicting a given class for examples belonging to a given small disjunct. Since it is necessary to discover several rules to cover examples of several classes in several different small disjuncts, GA-Small is run several times for a given dataset. More precisely, one needs to run GA-Small $d * c$ times, where d is the number of small disjuncts and c is the number of classes to be predicted. For a given small disjunct, the k-th run of GA-Small, $k = 1,...,c$, discovers a rule predicting the k-th class.

The genome of an individual consists of a conjunction of conditions composing a given rule antecedent. Each condition is an attribute-value pair, as shown in Figure 1. In this figure A_i denotes the i-th attribute and Op_i denotes a logical/relational operator comparing A_i with one or more values V_{ij} belonging to the domain of A_i, as follows. If attribute A_i is categorical (nominal), the operator Op_i is "*in*", which will produce rule conditions such as "A_i *in* $\{V_{i1},...,V_{ik}\}$", where $\{V_{i1},...,V_{ik}\}$ is a subset of the values of the domain of A_i. By contrast, if A_i is continuous (real-valued), the operator Op_i is either "≤" or ">", which will produce rule conditions such as "$A_i \leq V_{ij}$", where V_{ij} is a value belonging to the domain of A_i. Each condition in the genome is associated with a flag, called the active bit B_i, which takes on the value 1 or 0 to indicate whether or not, respectively, the i-th condition is present in the rule antecedent (phenotype). This allows GA-Small to use a fixed-length genome (for the sake of simplicity) to represent a variable-length rule antecedent (phenotype).

A_1 $Op_1\{V_{1j}..\}$	B_1	...	A_i $Op_i\{V_{ij}..\}$	B_i	...	A_n $Op_n\{V_{nj}..\}$	B_n

Figure 1: Structure of the genome of an individual.

For a given GA-Small run, the genome of an individual consists of n genes (conditions), where $n = m - k$, m is the total number of predictor attributes in the dataset and k is the number of ancestor nodes of the decision tree leaf node identifying the small disjunct in question. Hence, the genome of a GA-Small individual contains only the attributes that were *not* used to label any ancestor of the leaf node defining that small disjunct.

To evaluate the quality of an individual GA-Small uses the fitness function:

$$\text{Fitness} = (TP / (TP + FN)) * (TN / (FP + TN)) \qquad (1)$$

where TP, FN, TN and FP – standing for the number of true positives, false negatives, true negatives and false positives – are well-known variables often used to evaluate the performance of classification rules – see e.g. (Hand, 1997). In formula (1) the term (TP / (TP + FN)) is usually called sensitivity (*Se*) or true positive rate, whereas the term (TN / (FP + TN)) is usually called specificity (*Sp*) or true negative rate. These two terms are multiplied to foster the GA to discover rules having both high *Se* and high *Sp*.

GA-Small uses tournament selection, with tournament size of 2 (Michalewicz, 1996). It also uses standard one-point crossover with crossover probability of 80%, and mutation probability of 1%. Furthermore, it uses elitism with an elitist factor of 1 – i.e., the best individual of each generation is passed unaltered into the next generation.

GA-Small also includes an operator especially designed for simplifying candidate rules. The basic idea of this rule-pruning operator is to remove several conditions from a rule to make it shorter. This operator is applied to every individual of the population, right after the individual is formed.

Unlike the usually simple operators of GA, GA-Small's rule-pruning operator is an elaborate procedure based on information theory (Cover & Thomas, 1992). Hence, it can be regarded as a way of incorporating a classification-related heuristic into a GA for rule discovery. The heuristics in question is to favor the removal of rule conditions with low information gain, while keeping the rule conditions with high information gain. In other words, the larger information gain of a rule condition has the smaller probability of removing that condition from the rule – see (Carvalho & Freitas, 2000a; 2000b) for details.

Once all the $d * c$ runs of GA-Small are completed, examples in the test set are classified. For each test example, the system pushes the example down the decision tree until it reaches a leaf node. If that node is a large disjunct, the example is classified by the decision tree algorithm. Otherwise the system tries to classify the example by using one of the c rules discovered by the GA for the corresponding small disjunct. If there is no small-disjunct rule covering the test example it is classified by a default rule, which predicts the majority class among the examples belonging to the current small disjunct. If there are two or more rules discovered by the GA covering the test example, the conflict is solved ty using the rule with the largest fitness (on the training set) to classify that example.

3 THE EXTEND HYBRID DECISION-TREE / GENETIC-ALGORITHM METHOD FOR RULE DISCOVERY

In the previous section we have reviewed GA-Small, the GA algorithm for discovering small disjunct rules previously proposed as part of our hybrid decision-tree/GA method. The two main limitations of that GA are:

(a) Each run of GA-Small has access to a very small training set, consisting of just a few examples belonging to a single leaf node of a decision tree. Intuitively, this makes it difficult to induce reliable classification rules in some cases.

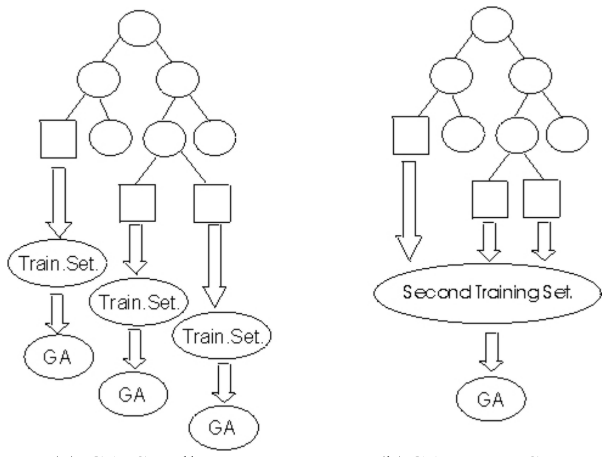
(b) Although each run of the GA is relatively fast (since it uses a small training set), the hybrid method as a whole has to run the GA many times (since the number of GA-Small runs is proportional to the number of small disjuncts and the number of classes). Hence, the hybrid C4.5/GA-Small method turns out to be considerably slower than the use of C4.5 alone.

These two limitations were our motivation to develop a new GA for discovering small disjunct rules. We stress that in this paper we propose just a new GA, without modifying the decision-tree algorithm of the above-mentioned hybrid method.

By comparison with GA-Small, the new GA proposed in this paper – denoted GA-Large-SN, as mentioned above – involves five major modifications. These modifications are described in the detail in the next subsections.

3.1 INCREASING THE CARDINALITY OF THE TRAINING SET

In our new GA-Large-SN, all the examples belonging to all the leaf nodes considered small disjuncts are grouped in a single training set, called the "second training set" (to distinguish it from the original training set used by C4.5 to build the decision tree). This second training set is provided as input data for the GA. This is the most important characteristic of GA-Large-SN, and it is the basis for the other characteristics discussed below.



(a) GA-Small (b)GA-Large-SN

Figure 2: Differences in the training sets of the GAs

This characteristic of GA-Large-SN is illustrated in Figure 2(b), where one can clearly see that all small disjuncts are grouped into a single, relatively large training set. This is in sharp contrast with the approach used by GA-Small (described in section 2), illustrated in Figure 2(a), where one can clearly see that each small disjunct is used as a small training set.

3.2 USING A NICHING METHOD TO FOSTER THE DISCOVERY OF MULTIPLE RULES

As a result of the above-discussed increase in the cardinality of the training set, one needs to discover several rules to cover the examples of each class. (Recall that this was not the case with the approach described in section 2, since in that approach it was assumed that a GA-Small run had to discover a single rule for each class.) Therefore, in our new GA-Large-SN it is essential to use some kind of niching method, in order to foster population diversity and avoid its convergence to a single rule. In this work we use a sequential niching method (Beasley et al., 1993). We chose this kind of method for two reasons. First, its simplicity. Second, and most important, it does not require the specification of additional parameters for its execution, unlike well-known niching methods such as fitness sharing (Goldberg & Richardson, 1987) and crowding (Mahfoud, 1995).

```
BEGIN
/* TrainingSet-2 contains all examples belonging to all small disjuncts */
 RuleSet = ∅;
 build TrainingSet-2;
 WHILE cardinality(TrainingSet-2) > 5
    run the GA;
    add the best rule found by the GA to RuleSet;
    remove from TrainingSet-2 the examples
        correctly covered by that best rule;
 END-WHILE
END-BEGIN
```

Figure 3: GA with sequential niching for discovering small disjunct rules

The pseudo-code of our GA with sequential niching is shown, at a high level of abstraction, in Figure 3. It starts by initializing the set of discovered rules (denoted RuleSet) with the empty set and building the second training set (denoted TrainingSet-2), as explained above. Then it iteratively performs the following loop. First, it runs the GA, using TrainingSet-2 as the training data for the GA. The best rule found by the GA (i.e., the best individual of the last generation) is added to RuleSet. Then the examples correctly covered by that rule are removed from TrainingSet-2, so that in the next iteration of the WHILE loop TrainingSet-2 will have a smaller cardinality. An example is "correctly covered" by a rule if the example's attribute values satisfy all the conditions in the rule antecedent and the example belongs to the same class as predicted by the rule. This process is iteratively performed while the number of examples in TrainingSet-2 is greater than 5. (It is assumed that when the cardinality of TrainingSet-2 is smaller than 5 there are too few examples to allow the discovery of a reliable classification rule.)

It should be noted that the sequential niching method used in this work is a variation of the one proposed by (Beasley et al., 1993). The latter actually requires the specification of a parameter, associated with a distance metric, for modifying the fitness landscape according to the location of solutions found in previous iteractions. In order to implement this parameter, the author uses an Euclidian distance.

By contrast, there is no need for this kind of parameter in our version of sequential niching. In order to avoid that the same search spaced be explored several times, the examples that are correctly covered by the discovered rules are removed from the training set. Hence, the nature of the fitness landscape is automatically updated as rules are discovered along different iterations of the sequential niching method.

3.3 MODIFICATION OF THE METHOD USED TO DETERMINE A RULE'S CONSEQUENT

Each run of GA-Large-SN still discovers a single rule, and a rule's consequent (the class predicted by the rule) is not encoded into the genome, like in the GA-Small described in section 2. However, unlike GA-Small, in GA-Large-SN the consequent of each rule is not fixed upfront for all rules (individuals) in the population. Rather, the consequent of each rule is dynamically chosen as a function of the rule's antecedent. More precisely, a rule's consequent is chosen as the most frequent class in the set of examples covered by that rule's antecedent.

3.4 A NEW HEURISTICS FOR RULE PRUNING

The GA-Large-SN proposed in this paper uses a new heuristic measure for rule pruning. This measure is based on the idea of using the decision tree built by C4.5 to compute a classification accuracy rate for each attribute, according to how accurate were the classifications performed by the decision tree paths in which that attribute occurs. That is, the more accurate were the classifications performed by the decision tree paths in which a given attribute occurs, the higher the accuracy rate associated with that attribute, and the smaller the probability of removing that a condition with attr ibute from a rule. The computation of an accuracy rate for each attribute is performed by the procedure shown in Figure 4.

The computation of the accuracy rate associated with each attribute is performed as follows. For each attribute A_i, the algorithm checks each path of the decision tree built by C4.5 in order to determine whether or not A_i occurs in that path. (The term path is used here to refer to each complete path from the root node to a leaf node of the tree.) For each path p in which A_i occurs, the algorithm computes two counts, namely the number of examples classified by the rule associated with path p, denoted $\#Classif(A_i,p)$, and the number of examples *correctly* classified by the rule associated with path p, denoted $\#CorrClassif(A_i,p)$.

```
BEGIN
  $Count\_of\_Unused\_Attr = 0$;
  FOR each attribute $A_i$, $i$=1,...,$m$
     IF attribute $A_i$ occurs in at least one path in the tree
        THEN compute the accuracy rate of $A_i$, denoted $Acc(A_i)$ (see text);
        ELSE increment $Count\_of\_Unused\_Attr$ by 1;
     END-IF
  END-FOR
  $Min\_Acc$ = the smallest accuracy rate among all attributes
               that occur in at least one path in the tree;
  FOR each of the attributes $A_i$, $i$=1,...,$m$, such that $A_i$
        does not occur in any path in the tree
     $Acc(A_i) = Min\_Acc / Count\_of\_Unused\_Attr$;
  END-FOR
  $Total\_Acc = \sum_{i=1}^{m} Acc(A_i)$ ;
  FOR each attribute $A_i$, $i$=1,...,$m$
     Compute the normalized accuracy rate of $A_i$,
        denoted $Norm\_Acc(A_i)$, as:
        $Norm\_Acc(A_i) = Acc(A_i) / Total\_Acc$ ;
  END-FOR
END-BEGIN
```

Figure 4: Computation of each attribute's accuracy rate, for rule pruning purposes

where Z_i is the number of decision tree paths where attribute A_i occurs. Note that formula (2) is used only for attributes that occur in at least one path of the tree. All the attributes that do not occur in any path of the tree are assigned the same value of $Acc(A_i)$, and this value is determined by the formula:

$$Acc(A_i) = Min\_Acc / Count\_of\_Unused\_Attr , \qquad (3)$$

where Min_Acc and $Count\_of\_Unused\_Attr$ are determined as shown in Figure 4.

Finally, the value of $Acc(A_i)$ for every attribute A_i, i=1,...,m, is normalized by dividing its current value by $Total\_Acc$, which is determined as shown in Figure 3.

Once the normalized value of accuracy rate for each attribute A_i, *denoted* $Norm\_Acc(A_i)$, has been computed by the procedure of Figure 4, it is directly used as a heuristic measure for rule pruning. The basic idea here is the same as the basic idea of the rule pruning procedure mentioned in section 2. In that section, where the heuristic measure was the information gain, it was mentioned that the larger the information gain of a rule condition, the smaller the probability of removing that condition from the rule. In GA-Large-SN, we replace the information gain of a rule condition with $Norm\_Acc(A_i)$, the normalized value of the accuracy rate of the attribute included in the rule condition. Hence, the larger the value of $Norm\_Acc(A_i)$, the smaller the probability of removing the i-th condition from the rule. The remainder of the rule pruning procedure proposed in (Carvalho & Freitas 2000a) remains essentially unaltered.

Note that the accuracy rate-based heuristic measure for rule pruning proposed here effectively exploits information from the decision tree built by C4.5. Hence, it can be considered as a kind of hypothesis-driven measure, since it is based on a hypothesis (in our case, a decision tree) previously constructed by a data mining algorithm.

By contrast, the previously-mentioned information gain-based heuristic measure does not exploit such information. Rather, it is a measure whose value is computed directly from the training data, independent of any data mining algorithm. Hence, it can be considered as a kind of data-driven measure.

3.5 INCREASING THE GENOME LENGTH

Recall that in GA-Small (reviewed in section 2) the genome contained only the attributes which were *not* used to label any ancestor of the leaf node defining the small disjunct being processed by the GA. That approach made sense because GA-Small was using as the training set only the examples belonging to a single leaf node. Clearly, the attributes in the ancestor nodes of that leaf node were not useful to distinguish between classes of examples in the leaf node, since all those examples had the same values for those attributes.

However, the situation is different in the case of the new GA-Large-SN proposed in this paper. Now the training set of the GA consists of all the examples belonging to all the leaf nodes that are considered small disjuncts – i.e., all those examples are effectively mixed into a single training set. Hence, the above notion of "attributes in the ancestor nodes of a single leaf node" is not meaningful any more. Therefore, in GA-Large-SN the genome contains m genes, where m is the number of attributes of the data being mined. I.e., all attributes can occur in the rule represented by an individual, so that in theory a rule can contain at most m conditions in its antecedent. Of course, in practice the number of conditions in a rule will be much smaller than m, due to the use of the above-discussed rule pruning operator.

4 COMPUTATIONAL RESULTS

We have evaluated the performance of GA-Large-SN across eight public-domain data sets of the the well-known data repository of the UCI (University of California at Irvine), available at:

http://www.ics.uci.edu/~mlearn/MLRepository.html.

The examples that had some missing value were removed from these data sets. In the Adult data set we have used the predefined division of the data set into a training and a test set. In the Connect data set we have randomly partitioned the data into a training and a test set with 47290 and 20267 examples, respectively. In the other datasets we have run a well-known 10-fold cross-validation procedure, which essentially works as follows. The data set is randomly partitioned into 10 mutually-exclusive and exhaustive partitions. Then the classification algorithm is run 10 times. In the i-th run, i = 1,...,10, the i-th partition is used as the test set, and the remaining nine partitions are grouped and used as the training set. After the 10 runs are over, the reported accuracy rate is the average accuracy rate over all those 10 runs.

In our experiments we have used a commonplace definition of small disjunct, based on a fixed threshold of the number of examples covered by the disjunct. The definition is: "A decision-tree leaf is considered a small disjunct if and only if the number of examples belonging to that leaf is smaller than or equal to a fixed size *S*."

In order to better evaluate the performance of GA-Large-SN, it is important to compare it against other classification method(s). In particular, we wanted to compare the hybrid system against another method that induces rules or trees (which can be straightforwardly converted to rules). In this case the kind of knowledge representation used by the systems being compared is the same, and the difference in the results will reflect mainly differences in search strategies. Hence, we can compare the evolutionary search strategy of GA-Large-SN against the local, greedy search strategy of a rule induction or decision tree algorithm.

Within this spirit we report the results of experiments comparing our hybrid C4.5/GA-Large-SN system with two other classification methods. The first is C4.5 alone, which is used to classify all examples – i.e., both large-disjunct examples and small-disjunct examples. The second is a "double run" of C4.5, hereafter called "double C4.5" for short. The later is a new way of using C4.5 to cope with small disjuncts, as follows.

The main idea of our "double C4.5" is to build a classifier running twice the algorithm C4.5. The first run considers all examples in the original training set, producing a first decision-tree. Once all the examples belonging to small disjuncts have been identified by this decision tree, the system groups all those examples into a single example subset, creating the "second training set", as described above for GA-Large-SN (see Figure 2(b)). Then C4.5 is run again on this second, reduced training set, producing a second decision tree. In other words, the second run of C4.5 uses as training set exactly the same "second training set" used by GA-Large-SN. This makes the comparison between GA-Large-SN and "double C4.5" very fair.

In order to classify a new example, the rules discovered by both runs of C4.5 are used as follows. First, the system checks whether the new example belongs to a large disjunct of the first decision tree. If so, the class predicted by the corresponding leaf node is assigned to the new example. Otherwise (i.e., the example belongs to one of the small disjuncts of the first decision tree), the new examples are classified by the second decision tree.

The motivation for this more elaborated use of C4.5 was an attempt to create a simple algorithm that was more effective in coping with small disjuncts.

Recall that the hybrid C4.5/GA-Large-SN method has an important parameter, namely the small-disjunct size threshold (*S*). In order to evaluate how robust the method is with respect to this parameter, we have done experiments with four different values of *S*, namely 3, 5, 10 and 15. For each of these four *S* values, we have done ten different experiments, varying the random seed used to generate the initial population of individuals. The results reported below, for each value of *S*, are an arithmetic average of the results over these ten different experiments. Therefore, the total number of experiments is 40 (4 values of *S* * 10 different random seeds). In addition, recall that each of these 40 experiments actually consists of a 10-fold cross-validation run for most data sets (with the exception of the Adult and Connect data sets, where a single division of the data into training and test sets was used).

Each run of GA-Large-SN is relatively fast, so that each of these 40 experiments took a processing time on the order of six minutes for the biggest data set, Connect, and for the largest value of *S* (15), on a Pentium III with 192Mb of RAM.

We now report results comparing the classification accuracy rate (on the test set) of the proposed hybrid C4.5/GA-Large-SN with C4.5 alone (Quinlan, 1993) and with the above-described "double C4.5". We have used C4.5's default parameters. In each GA-Large-SN run the population has 200 individuals, and the GA is run for 50 generations.

Table 1: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for *S* = 3

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	**78.06 (0.3)**	77.86 (0.1) + -
Adult	78.62 (0.3)	81.19 (0.3)	**85.45 (0.1)** + +
Crx	91.79 (2.1)	92.57 (1.2)	**93.69 (1.2)**
Hepatitis	80.78(13.3)	78.95 (6.9)	**89.25 (9.5)**
House-votes	93.62 (3.2)	**97.32 (2.4)**	97.18 (2.5)
Segmentation	**96.86 (1.1)**	76.62 (2.8)	81.46 (1.1) - +
Wave	75.78 (1.9)	68.18 (3.7)	**83.86 (2.0)** + +
Splice	65.68 (1.3)	55.65 (6.0)	**70.62 (8.6)** +

The results are shown in Tables 1, 2, 3 and 4 referring to *S* values of 3, 5, 10 and 15, respectively. In these tables the first column indicates the data sets. The second column shows the accuracy rate on the test set achieved by C4.5 alone, classifying both large-disjunct and small-disjunct examples. The third column reports the accuracy rate for C4.5(2). The fourth column reports the accuracy rate achieved by our hybrid C4.5/GA-Large-SN system, using C4.5 to classify large-disjunct examples and our GA classify small-disjunct examples. The values between brackets are standard deviations. For each data set, the highest accuracy rate among the three classifiers is shown in bold.

In addition, in the fourth column we indicate, for each data set, whether or not the accuracy rate of C4.5/GA-Large-SN is significantly different from the accuracy rates of the other two methods. More precisely, the cases where the accuracy rate of C4.5/GA-Large-SN is significantly better (worse) than the accuracy rate of each of the other two methods is indicated by the "+" ("-") symbol. A difference between two methods is deemed significant when the corresponding accuracy rate intervals (taking into account the standard deviations) do not overlap.

Let us now analyze the results of Tables 1, 2, 3 and 4 starting with Table 1 (where *S* = 3). In this table C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 5 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. In addition, C4.5/GA-Large-SN is significantly better than C4.5(2) in 4 data sets, and the reverse is true in only 1 data set.

Table 2: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for *S* = 5

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	77.09 (0.3)	**77.85 (0.2)** + +
Adult	78.62 (0.3)	79.27 (0.3)	**85.50 (0.2)** + +
Crx	91.79 (2.1)	92.03 (1.0)	**93.06 (1.6)**
Hepatitis	80.78(13.3)	75.67 (17.1)	**89.48 (9.7)**
House-votes	93.62 (3.2)	93.54 (3.9)	**97.44 (2.9)**
Segmentation	**96.86 (1.1)**	74.49 (3.4)	80.41 (1.0) - +
Wave	75.78 (1.9)	65.59 (4.4)	**85.31 (2.4)** + +
Splice	65.68 (1.3)	57.45 (8.7)	**70.44 (7.8)**

In Table 2 (where *S* = 5) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 7 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 4 data sets, and the reverse is not true in any data set.

Table 3: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for *S* = 10

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	76.19 (0.3)	**76.95 (0.1)** + +
Adult	78.62 (0.3)	76.06 (0.3)	**80.04 (0.1)** + +
Crx	**91.79 (2.1)**	90.78 (1.2)	91.66 (1.8)
Hepatitis	80.78(13.3)	82.36 (18.7)	**95.05 (7.2)**
House-votes	93.62 (3.2)	89.16 (8.0)	**97.65 (2.0)**
Segmentation	**96.86 (1.1)**	72.93 (5.5)	78.68 (1.1) -
Wave	75.78 (1.9)	64.93 (3.9)	**83.95 (3.0)** + +
Splice	65.68 (1.3)	61.51 (6.6)	**70.70 (6.3)**

In Table 3 (where *S* = 10) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 6 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 3 data sets, and the reverse is not true in any data set.

In Table 4 (where *S* = 15) C4.5/GA-Large-SN outperforms both C4.5 alone and C4.5(2) in 6 of the 8 data sets. C4.5/GA-Large-SN is significantly better than C4.5 alone in 3 data sets, and the reverse is true in only 1 data set. C4.5/GA-Large-SN is significantly better than C4.5(2) in 3 data sets and the reverse is not true in any data set.

Table 4: Accuracy Rate (%) of C4.5, "double C4.5" (C4.5 (2)) and our hybrid C4.5/GA-Large-SN for *S* = 15

Data set	C4.5	C4.5(2)	C4.5/GA
Connect	72.60 (0.3)	74.95 (0.3)	**76.01 (0.3)** + +
Adult	78.62 (0.3)	74.29 (0.3)	**79.32 (0.2)** + +
Crx	**91.79 (2.1)**	90.02 (0.8)	90.40 (2.4)
Hepatitis	80.78(13.3)	66.16 (19.1)	**82.52 (7.0)**
House-votes	93.62 (3.2)	88.53 (8.4)	**95.91 (2.3)**
Segmentation	**96.86 (1.1)**	73.82 (5.8)	77.11 (1.9) –
Wave	75.78 (1.9)	65.53 (4.0)	**82.65 (3.7)** + +
Splice	65.68 (1.3)	64.35 (4.7)	**70.62 (5.5)**

We can summarize the results of the above four tables as follows.

- C4.5/GA-Large-SN outperformed C4.5 alone in 87.50% of the data sets for *S* = 3 and *S* =5, and in 75% for *S* = 10 and *S* = 15.
- C4.5/GA-Large-SN outperformed C4.5(2) in 75% of the data sets for *S* = 3, and in 100% for *S* = 5, *S* = 10 and *S* = 15.
- Considering the results of the three methods for every data set and every value of *S*, the best accuracy rate was obtained by C4.5/GA in 78.2% of the cases, by C4.5 alone in 15.6% of the cases, and by C4.5(2) in only 6,2% of the cases.

Finally, a brief comment on computational time is appropriate here. We have mentioned, at the beginning of section 3, that one of our motivations for designing GA-Large-SN was to reduce processing time, by comparison with GA-Small. We have done an experiment comparing the processing time of both GA-Large-SN and GA-Small, on the same machine, in the same data set, namely the Connect data set – which is the largest data used in the above-reported experiments with GA-Large-SN. We observed that GA-Large-SN takes about only 12% of the processing time of GA-Small in the Connect data set.

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper we have described a new hybrid decision-tree/GA (C4.5/GA-Large-SN) method. The GA component of this method, called GA-Large-SN, consists of major modifications of the original GA (here called GA-Small) proposed by (Carvalho & Freitas 2000a), as discussed throughout section 3.
We have compared the new hybrid C4.5/GA-Large-SN system with 2 algorithms based on the use of C4.5 alone, namely: (a) the default version of C4.5; (b) a "double run of C4.5", which uses the same training set as GA-Large-SN. This comparison was performed across four different values for a parameter defining the size of a small disjunct.

Overall, the hybrid C4.5/GA-Large-SN obtained considerably better accuracy rates than both above-mentioned versions of C4.5 alone, in all the four definitions of small-disjunct size used in this paper.

In this paper we have focused on comparing the performance of the hybrid C4.5/GA-Large-SN with the performance of C4.5 alone, since C4.5 is a very well-known algorithm that is often used in comparison with other algorithms in the literature. In future research we also intend to compare the predictive accuracy of the rules discovered by C4.5/GA-Large-SN with the predictive accuracy of the rules discovered by the GA alone.

References

D.Beasley, D.R.Bull, R.R.A.Martin, (1993). Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation 1(2),* 101-125. MIT Press

D.R.Carvalho, A.A.Freitas (2000a). A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in Data Mining. *Proc 2000 Genetic and Evolutionary Computation Conf. (Gecco-2000), 1061-1068*. Las Vegas, NV, USA. July.

D.R.Carvalho, A.A.Freitas (2000b). A genetic algorithm-based solution for the problem of small disjuncts. Principles of Data Mining and Knowledge Discovery (*Proc. 4th European Conf., PKDD-2000*. Lyon, France). Lecture Notes in Artificial Intelligence 1910, 345-352. Springer-Verlag.

T.M.Cover, J.A.Thomas (1991) *Elements of Information Theory.* John Wiley&Sons

V.Dhar, D.Chou, F.Provost (2000) Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid With Entropy Reduction. *Data Mining and Knowledge Discovery 4(4)*, 251-280. Oct. 2000.

A.P.Danyluk, F.J.Provost (1993). Small Disjuncts in Action: Learning to Diagnose Erros in the Local Loop of the Telephone Network, *Proc. 10th International Conference Machine Learning*, 81-88.

A.A.Freitas (2001) Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3)*, Nov. 2001, pp. 177-199.

A.A.Freitas (2002) Evolutionary Algorithms. Chapter of forthcoming *Handbook of Data Mining and Knowledge Discovery.* Oxford University Press, 2002.

D.E.Goldberg, J.Richardson (1987) Genetic Algorithms with Sharing for Multimodal Function Optimization. *ICGA-87*. 41-49.

D.J.Hand (1997). *Construction and Assessment of Classification Rules*, John Wiley & Sons.

R.C.Holte, L.E.Acker, B.W.Porter (1989). Concept Learning and the Problem of Small Disjuncts, *Proc. IJCAI – 89*, 813-818.

S.W.Mahfoud (1995). *Niching Methods for Genetic Algorithms*, Illigal Report N. 95001, Thesis Submitted for degree of Doctor of Philosophy in Computer Science. University of Illinois at Urbana-Champaign.

Z.Michalewicz (1996) *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd Ed. Springer-Verlag.

J.R.Quinlan (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publisher.

G.M.Weiss, H.A.Hirsh (2000). Quantitative Study of Small Disjuncts, *Proc. of Seventeenth National Conference on Artificial Intelligence.* Austin, Texas, 665-670.