2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Simplifying Decision Trees Learned by Genetic Programming

Alma Lilia Garcia-Almanza and Edward P.K. Tsang

Abstract— **This work is motivated by financial forecasting using Genetic Programming. This paper presents a method to post-process decision trees. The processing procedure is based on the analysis and evaluation of the components of each tree, followed by pruning. The idea behind this approach is to identify and eliminate rules that cause misclassification. As a result we expect to keep and generate rules that enhance the classification. This method was tested on decision trees generated by a genetic program whose aim was to discover classification rules in financial stock markets. From experimental results we can conclude that our method is able to improve the accuracy and precision of the classification.**

I. Introduction

Decision trees have been widely used in machine learning for classification and prediction. However, the overfitting and complexity of resulting trees have disclosed the necessity of pruning procedures. Several classifiers have incorporated pruning methods, for example: Classifier CART implemented *minimal cost-complexity pruning* [1], ID3 incorporated *reduced error pruning* and *pessimistic pruning* [2], while classifier C4.5 integrated *Error-based pruning* [3]. Breiman [1] and Quinlan [2] have asserted that tree simplification can benefit almost all decision trees when removing parts that do not contribute to classification accuracy. They argued that resultant trees are less complex and more understandable. Furthermore, this simplification helps to control overfitting.

Decision trees generated by Genetic Programming (GP) [4] tend to grow [5], [6], [7], [8]. However, this growth is not necessarily proportional to the quality of the resultant solution. We presume that decision tree simplification can be beneficial to trees produced by GP. Code growth has been controlled introducing a variety of methods. These are grouped in three main types: parsimony pressure, operator modification and code modification [9]. Parsimony pressure tries to evolve small solutions penalizing large individuals, for instance, to establish a maximum depth allowed [4], tarpeian method [10] or the implementation of Minimum Description Length principle (MDL) in the fitness function [11]. Operator modification is represented by no-destructive crossover [12]. Code modification involves changing the structure of the code during or after the evolution, e.g. the pruning method implemented by Eggermont *et al*. [13]. According to Soule, code modification methods have not been explored in depth because it involves the use of more computational resources [12]. However, researches of other machine learning classifiers [1], [3] prefer pruning techniques instead of stopping criterions, they have pointed out that pruning is slower but more reliable because it produces more exploration.

We propose a new approach called *Scenario Method* (SM) to analyse decision trees produced by GP. The aim of this approach is to produce smaller trees with higher prediction accuracy. Analysis is done by identifying relevant parts that contribute to the classification task as well as to separate those fragments that deteriorate their performance or cause overfitting. From results we affirm that this method is able to produce compact trees improving their *accuracy*[1] and their *precision*[2]. This work is illustrated with a dataset composed by closing prices from the London Financial Market. The remainder of this paper is organized as follows: Section II contains an overview of the problem that illustrates our method, Section III presents the SM procedure, while Section IV describes the experiments to test our approach. Section V presents the experiment results. Finally, Section VI summaries the conclusions.

II. Problem Description

To illustrate SM, it was applied to a discovery classification rule problem. The idea is to classify a financial stock dataset in order to predict future movements in the stock price. This problem has been addressed previously by [15], [16], [17]. Every case in the dataset is composed by a *signal* and a set of attributes or *independent variables*. The signal indicates the opportunities for *buying* or *not buying* and *selling* or *not selling*. The signal is calculated looking ahead in a future horizon of n units of time trying to detect an increase or decrease of at least $r\%$ [3]. The independent variables are composed by financial indicators derived from financial technical analysis. Technical analysis is used in financial markets to analyse the price behaviour of stocks. This is mainly based on historical prices and volume trends [18].

III. Scenario Method

The main goal of SM is to simplify decision trees by means of rule selection. This procedure involves dividing the problem using *class division*. The next step is to analyse the tree in order to identify its rules *(Rule extraction)*. Every rule is evaluated *(Rule evaluation)* to select those rules that contribute positively to the classification task. Finally, rules

Alma Lilia Garcia-Almanza is with the Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK (phone: (44) 01206-873975; email: algarc@essex.ac.uk).

Edward P.K. Tsang is with the Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK (phone: (44) 01206-872774; email: edward@essex.ac.uk).

[1]Accuracy is the proportion of the total number of predictions that were correct [14]

[2]Precision is the proportion of the predicted positive cases that were correct [14]

[3]The gain or loss of an investment over a specified period of time

0-7803-9487-9/06/$20.00/©2006 IEEE

TABLE I

DISCRIMINATOR GRAMMAR.

G	→	<Root>
<Root>	→	"If-then-else", <Conjunction> | <Condition>,"Class","No Class"
<Conditional>	→	<Operation>, <Variable>, <Threshold> | <Variable>
<Conjunction>	→	"and"|"or",<Conjunction>|<Conditional>, <Conjunction>|<Conditional>
<Operation>	→	"<", ">"
<Variable>	→	Variable_1 | Variable_2 | ... Variable_n
<Threshold>	→	Real number

with poor performance will be removed from the tree *(Tree pruning)*. The above procedures will be explained in detail in the following sections.

A. Class division

To divide the classification problem, a population per each class will be evolved independently (example of a class is to *Buy*). For this purpose decision trees are generated and evolved using *Discriminator Grammar* (DG). This grammar[4] produces trees which classify or not a single class. Table I shows the discriminator grammar and Figure 1 illustrates a decision tree that was created using DG. At this point we introduce the concept of *Conditional Node*, which refers to any node with syntax <Conditional> in discriminator grammar. Class division has previously been applied by other researches such as Teller [20], who evolved an individual program per class using Parallel Architecture Discovery and Orchestration (PADO).

B. Rule extraction

This process analyses the decision trees in order to delimit their rules. Let *T* be a tree with syntax DG. *T* is composed by rules, so it can be expressed as the union of its rules such as $T = (R_1 \cup R_2 \cup \cdots R_n)$ where R_i is a rule and *n* is the total number of rules in *T*. In order to satisfy the tree *T* at least one rule must be satisfied. A rule R_k is a minimum set of conditions that satisfy the tree *T*. The rule R_k can be expressed as the intersection of conditional nodes such as $R_k = (n_{k1} \cap n_{k2} \cap \cdots n_{kt})$ where n_{ki} is a condition and *kt* is the number of conditions in rule R_k . To satisfy rule R_k every condition in this rule has to be satisfied. Let us define *Rule Map* as a matrix that lists the rules of a tree *T* and the conditions that composed every rule. The k^{th}-row in the rule map contains the conditions of rule R_k. Given that the size of rules could be different, the size of the matrix will be *N* x *L*, where *N* is the number of rules in the tree, and *L* is the size of the largest rule. When the length of a rule is smaller than *L* the empty spaces will be fulfilled using 0. Figure 1 shows an example of a decision tree and its rule map. The rule map is used to control the interactions between rules.

[4]The term grammar refers to a representation concerned with the syntactic components and the rules that specify it [19]

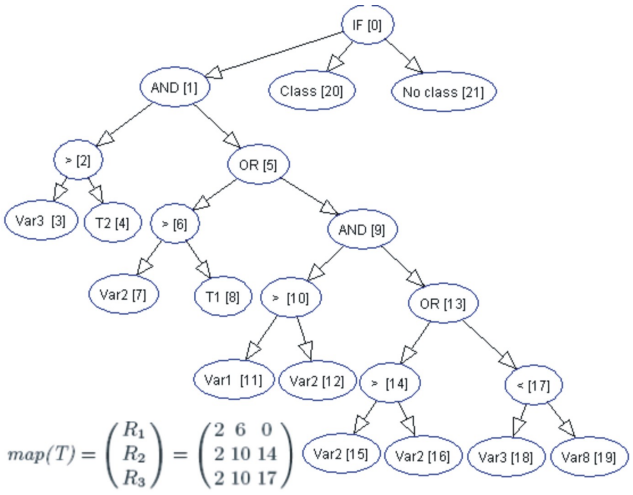

Fig. 1. Example of an individual generated by DG and its Rule Map.

TABLE II

CONFUSION MATRIX.

Predicted/Actual	Positive		Negative	
Positive	TP	True Positive	FP	False Positive
Negative	FN	False Negative	TN	True Negative

C. Rule evaluation

This section explains the procedure to evaluate the performance of each rule in the tree *T*. Every rule $R_i \in T$ is compared against the training dataset and the result is registered in a *Confusion Matrix* [5]. Thus there will be a confusion matrix M_i for each rule $R_i \in T$. Table II displays a confusion matrix for the classification of two classes.

It is important to keep in mind that this problem is exposed to imbalanced classes because the number of opportunities for investors may be infrequent. According to Kubat *et al.* [21], when imbalanced classes take place, it is not reliable to measure the performance of a classifier only using the equation of accuracy. This work is not only focused on the accuracy improvement but also in the precision improvement, because every decision involves an investment. To illustrate this, suppose that every time a true positive is succeeded we gain 1 unit, but every time a false positive is predicted we loss the same amount. Thus the number of true positives has to be bigger than the number of false positives; otherwise the final result will be a negative balance. Taking into account the previous consideration, let (1) be the equation to measure the rule contribution.

$$Ev(R_i) = \begin{cases} \left(\frac{TP_i}{P^T}\right)\left(\frac{TP_i - FP_i}{TP_i + FP_i}\right) & if\ TP_i + FP_i > 0 \\ 0 & Otherwise \end{cases} \quad (1)$$

[5]A confusion matrix consists of information about actual and predicted classifications done by a classifier system. It is an information summary of the performance of the classifier, it shows the accuracy of predicted class as well as errors and omissions[14].

Where the terms in (1) are the components in the confusion matrix and $P^T = TP_i + FN_i$, it is the total number of positive cases in the dataset. Notice that P^T is a constant number for all R_i because it depends on the dataset. The first parenthesis of equation (1) encloses the *recall*[6], it encourages the increment of true positive cases. The second parenthesis encloses an expression similar to *precision*. However, it severely penalizes the false positive cases. Equation (1) could be useful to perform classification in risky problems because it strongly discards the false positive cases.

D. Rule selection

Once the rules in tree *T* have been evaluated the next step is to perform a rule selection based on hypothetical scenarios for the union of rules. The rule with the highest evaluation is taken as a starting point, let us call it R_B. To disclose the potential of the remaining rules $R_\eta \in T$, the *Best* and the *Worst scenario* for $R_{B\eta} = (R_B \cup R_\eta)$ are calculated. Where $R_{B\eta}$ is the union of rules R_B (the best rule of *T*) and R_η. The best scenario is calculated assuming that the true positives cases in R_B and R_η are not overlapped and the false positive cases maximally overlap each other. Thus the best scenario for true positive and false positive cases are calculated as follows: ${TP_{B\eta}}^+ = TP_B + TP_\eta$ and ${FP_{B\eta}}^+ = max(FP_B, FP_\eta)$. Superscripts are used to indicate the scenario, it could be worst (-) or best (+). Subscripts are used to denote rules. The worst scenario is calculated in exactly the opposite way, it assumes that true positive cases maximally overlap. And false positive cases are not overlapped so the total number of false positive cases is the sum of false positives in both rules. Finally the worst scenario for true positive and false positive cases is ${TP_{B\eta}}^- = max(TP_B, TP_\eta)$ and ${FP_{B\eta}}^- = FP_B + FP_\eta$. Once the best and the worst scenarios were calculated, Equation (1) is applied to them as follows:

$$Ev({R_{B\eta}}^+) = \frac{TP_B+TP_\eta}{P^T} \cdot \frac{TP_B+TP_\eta-max(FP_B,FP_\eta)}{TP_B+TP_\eta+max(FP_B,FP_\eta)}$$

$$Ev({R_{B\eta}}^-) = \frac{max(TP_B,TP_\eta)}{P^T} \cdot \frac{max(TP_B,TP_\eta)-FP_B-FP_\eta}{max(TP_B,TP_\eta)+FP_B+FP_\eta}$$

Let us define the *Potential of Improvement* (PI) as the capacity of a rule for improving the tree *T*. PI is calculated using the distance between $Ev(R_B)$ and $Ev({R_{B\eta}}^+)$ as Figure 2 shows. The potential of improvement is calculated as follows:

$$PI(R_\eta) = \begin{cases} \frac{Ev({R_{B\eta}}^+)-Ev(R_B)}{Ev({R_{B\eta}}^+)-Ev({R_{B\eta}}^-)} & if\ \ Ev(R_B) \leq Ev({R_{B\eta}}^+) \\ 0 & Otherwise \end{cases}$$

[6]The recall (true positive rate) is the proportion of positive cases that were correctly identified [14].

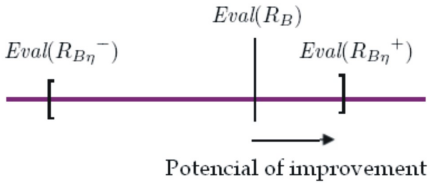

Fig. 2. Interval of the worst and the best scenario of $(R_B \cup R_\eta)$.

TABLE III
EXAMPLE

	R_η (TP_η, FP_η)	$R_B \cap R_\eta$	$R_B \cup R_\eta$ $(TP_{B\eta}, FP_{B\eta})$
$R_{\eta=1}$	$(1, 15)$	$(0, 15)$	$(41, 20)$
$R_{\eta=2}$	$(10, 2)$	$(10, 1)$	$(40, 21)$

Once the potential of improvement is calculated, it is necessary to decide whether or not the rule R_η is beneficial for the tree. A threshold from 0% to 100% is used to determine the level of pruning, it will be defined as *Pruning Threshold* (PT). If ($PI(R_\eta) <$ PT) the rule R_η will be pruned. When PT is close to 0 the level of pruning is low, but when PT is close to 100 a hard pruning is performed. The experiments included in this paper used different pruning thresholds (PT = 0%,10%,...90%) to find out the effect of this parameter.

At this point it is worth addressing an important question: why is the SM preferred rather than the direct evaluation of the combined rule $R_{B\eta}$? One of the reasons is that the direct evaluation of the new rule consumes more computational resources. Another reason is that SM avoids overfitting because it discloses the individual performance of the rule. To illustrate the last point we present an example where it is shown that the direct evaluation of $R_{B\eta}$ and the comparison against R_B does not give a good performance estimation of R_η. Let us express the evaluation of R_B (Best rule) as follows: $R_B = (TP_B, FP_B) = (40, 20)$. Now we add the rules R_1 and R_2 from Table III. The performance of R_1 = (1,15), this shows that it can produce more misclassifications than accurated results. However, direct evaluation suggests that R_1 is able to improve the tree. In contrast, SM discards this rule if we apply a low pruning threshold (greater than 7%). Now let's analyze R_2, its performance indicates that it is able to classify with a good rate of precision (83%). Nevertheless, direct evaluation discards R_2 because it classifies the same true positive cases than R_B. However the fact that R_2 classifies a subset of R_B in training data does not mean that they classify the same cases in other dataset. SM only discards R_2 when a hard pruning threshold is applied (bigger than 82%).

E. Tree pruning

After rule selection is performed the rules that do not achieve the expected pruning threshold will be removed. During the pruning procedure the condition map is used to detect the interactions between rules and determine which

```
PROCEDURE Prune( T, R_k , ConditionMap )
BEGIN
   /*Given the rule R_k =(n_k1 ∩ n_k2...∩n_kj...)
    where n_kj is a conditional node and R_k is
    the k^th-row in ConditionMap*/

    FOR each n_kj ∈ R_k
     IF (n_kj ∉ R_i where i ≠ k) THEN
         /* If n_kj is not part of other rule, delete it*/
        BEGIN
            n_p → Parent of node n_kj
            n_b →The other child of node n_p
            n_g →Parent of node n_p
            n_c1, n_c2 → The two children of node n_kj
            /* Delete n_kj, its parent and its children*/
            T → T − n_kj, n_p, n_c1, n_c2
            T → T  + Link between n_g and n_b
        END
     ConditionMap_kj → 0 /* Set 0 in node map */
    RETURN T
END
```

Fig. 3. Pruning Psedocode

nodes in bad rules can be pruned without affecting the useful rules. The pruning pseudocode is described in Figure 3. Notice that not all decision trees are applicable to SM, in the following cases it is not posible to prune the tree:

1) The tree is composed by a single rule.
2) SM does not suggest pruning to improve the tree.
3) SM suggests pruning but all conditional nodes to prune are involved in good rules.
4) The evaluation of the best rule is inferior to zero.

IV. Experiments Description

To test our approach a series of experiments were performed. The objective was to find out the effects of scenario method in the performance of decision trees. The performace is measured in terms of the accuracy, precision and tree size. Scenario method was tested on series of 25 runs. Every series comprises five populations from different stages of the evolutionary process. In order to discover the impact of pruning treshold the experiment was tested with different values for this parameter. The results of the experiment were grouped and averaged by generation and pruning threshold. The training data description and the procedure to generate the population for the experiment are described in the following sections.

A. Training data description

The dataset that was used to train the GP in the experiment came from the London stock market. The dataset contains 756 records that describe the behaviour of the *closing price*[7] for TESCO stock (from January, 2001 to January, 2004). The attributes of each record are composed by indicators derived from financial technical analysis; these were calculated on the base of the daily closing price, volume and some financial indices as FOOTSIE[8] and LIBOR[9]. The financial indicators used in the experiment are listed in Table IV. These were calculated using two periods of time, one short and one long. Each period provides a different interpretation. The shorter the time span, the more sensitive the indicator will be to changes.

TABLE IV
FINANCIAL INDICATORS USED IN THE EXPERIMENT

Indicator name	Short period (Days)	Long period (Days)
Price moving average	12	50
Price Trading breaking rule	5	50
Filter rule	5	63
Price volatility	12	50
Volumen moving average	10	60
Momentum	10	60
Momentum 10 days moving average	10	–
Momentum 60 days moving average	60	–
Generalized Momentum indicator	10	60
FOOTSIE moving average	12	50
LIBOR: 3 months moving average	12	50

B. Creation of populations

To test our approach in different stages of the evolutionary process, it was necessary to generate populations from different points of the evolution. By doing so a population of 1,000 individuals was created using DG, it was evolved during 100 generations. Every twenty generations the whole population was saved, therefore the result was five populations of 1,000 individuals each, let us call them $P_{20}, P_{40}, \cdots, P_{100}$ where subscripts indicates the number of the generation. The mentioned procedure was repeated 25 times in other to test the experiment with different sets of decision trees. Finally the experiment results were grouped and averaged by generation and pruning threshold. Table V presents the GP parameters used to evolve the populations.

TABLE V
SUMMARY OF PARAMETERS.

Parameter	Value
Population size	1,000
Initialization method	Growth
Generations	100
Crossover Rate	0.8
Mutation Rate	0.05
Selection	Tournament (size 2)
Elitism	Size 1
Fitness function	Equation (1)
Control bloat growing	Tarpeian method, 50% of those trees whose largest branch exceed 7 were penalized with 20% of the fitness for each node that surpassed the largest branch allowed.

[7]The settled price at which a traded instrument is last traded at on a particular trading day

[8]An index of 100 large capitalization companies stock on the London Stock Exchange.

[9]London interbank offered rate

V. Main Results

We now document the results obtained by applying SM to the set of populations described in section IV. The performance of the experiment is measured in terms of the prediction accuracy, precision, tree size and number of pruned trees. The experiment was tested using different pruning thresholds (PT = 0%,10%,..90%). All figures given in this section denote average results from series of 25 test runs.

A. Number of pruned trees

Figure 4 plots the number of trees that were pruned by SM. Every series represents a population in a specific generation $P_{20}, P_{40}, ..P_{100}$. As can be seen in X-axis every population was tested using different pruning thresholds.

Not surpisingly these results show that the increase in pruned trees is related to the number of generations. The number of pruned trees grows when the number of generations increases, this occurs because the tree size rises and there are more opportunities to perform a pruning. During earliest generations the number of pruned trees is low because in early stages of the evolutionary process the tree size is small and trees must contain more than one rule in order to be pruned. As an instance the average number of rules per tree in a population of generation twenty is 1.9. It means that there is a high number of trees that hold only one rule, as a consequence they can not be pruned.

On the other hand, as one might expect, the number of pruned trees increases when pruning threshold increases. This occurs because SM removes the rules whose PI does not achieve the PT and the increase of this threshold causes that many rules have to be pruned. Table VIII displays the number of pruned trees per population and threshold.

B. Accuracy improvement

Figure 5 displays the accuracy improvement achieved by SM. Every curve represents a population tested with different pruning thresholds. As it can be seen SM helped to improve the accuracy in almost all cases. The average improvement in accuracy is 4.6%. The best results are obtained when PI is less than 60%. However, the number of pruned trees decreases when threshold does the same. According to the experiment results the best thresholds are between 40% and 60%. In this range the accuracy improvement and the number of pruned trees are high . The results of some experiments uncovered that it is possible to have a slightly decrease in the accuracy when the threshold is close to 100% and the population has converged. It is because, when the PI is big the selection becomes stricter and some useful rules could be pruned. Table VI shows the accuracy of a standard GP and the new accuracy when SM is applied.

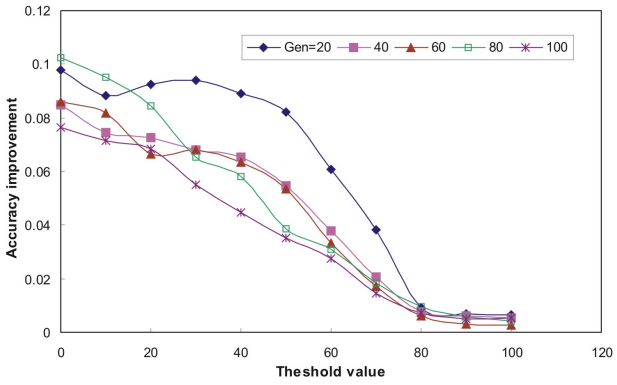

Fig. 5. Accuracy improvement

C. Precision improvement

Table VII describes the precision of a standar GP before and after SM was applied. The average precision improvement is 9%. The minimum improvement is 2% and the maximum is 16%. The improvement is affected by the tree generation and the PT. As it can be seen from table 6 the precision improvement declines when PT is close to 100%, or when the population starts to converge.

D. Tree size reduction

Table IX shows the tree size reduction per each generation. As can be seen SM reduces considerably the size of the trees, the average reduction is 27%. As one might expect

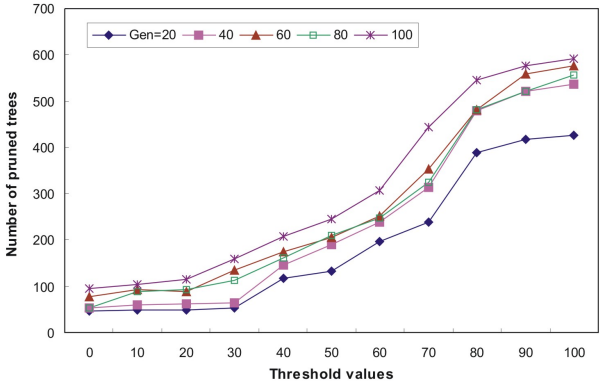

Fig. 4. Pruned trees

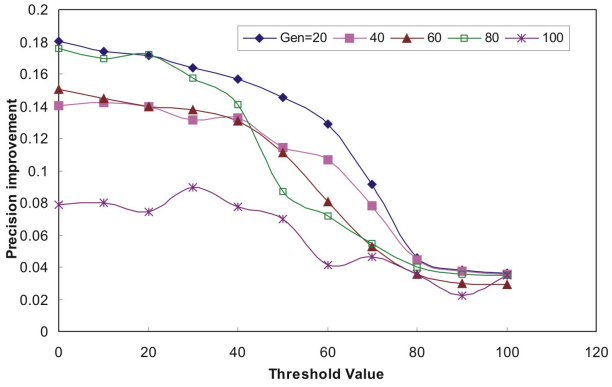

Fig. 6. Precision improvement

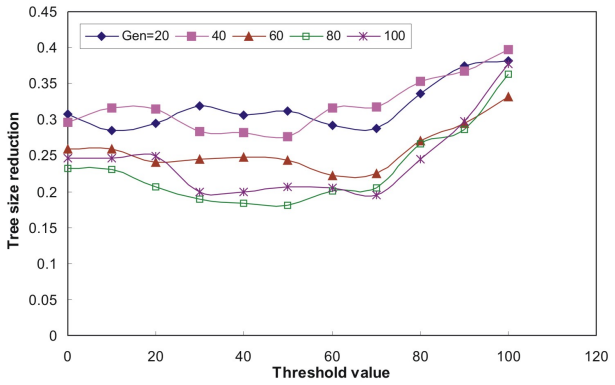

Fig. 7. Tree size reduction

the tree size reduction is related with the pruning threshold. When PT increases, the pruning is intensified because only rules with high PI can achieve the mentioned threshold. On the other hand the tree size reduction increases when the number of generations does the same. It can be explained that during earliest stages of the evolutionary process the tree size is small. When a tree holds only one rule, it can not be pruned so the possibilities to be pruned increases when the tree size rises. Figure 7 shows the tree size reduction achieved in the experiment.

VI. Conclusions

A pruning method for decision trees called *Scenario Method* (SM) has been presented. This pruning procedure applied for trees generated by genetic programming whose objective is to discover classification rules. The aim of this method is to identify rules that enhance the classification task as well as those that deteriorate the performance of the tree. The approach is based on the analysis of scenarios. The intensity of pruning is controlled by a pruning threshold. The new approach is tested on a financial classification problem. From the experimental results it can be concluded that SM is able to select useful parts of the tree, indicating which of them are able to contribute with the classification task. The pruning of non useful conditions improved the accuracy and precision of the decision trees. In addition SM shows that it is able to reduced the tree size. The improvement achieved by SM varies, it depends on the stage of the evolutionary process and the pruning threshold.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments of the paper. The first author thanks to Consejo Nacional de Ciencia y Tecnología (CONACyT) to support her studies at the University of Essex.

TABLE VI

Accuracy before and after scenario method was applied.

PT	Gen=20		Gen=40		Gen=60		Gen=80		Gen=100	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
0	.54	.65	.56	.66	.57	.66	.53	.67	.58	.65
10	.55	.64	.57	.65	.58	.65	.53	.67	.59	.65
20	.54	.64	.57	.65	.58	.65	.54	.66	.59	.65
30	.55	.65	.58	.65	.59	.65	.60	.68	.61	.66
40	.55	.65	.59	.66	.59	.65	.61	.68	.62	.65
50	.56	.65	.60	.66	.61	.66	.63	.68	.62	.65
60	.59	.66	.63	.67	.64	.67	.64	.67	.64	.66
70	.63	.67	.65	.68	.66	.68	.66	.68	.67	.68
80	.67	.68	.67	.68	.68	.69	.67	.69	.68	.69
90	.67	.68	.68	.68	.69	.69	.68	.69	.69	.69
100	.67	.68	.68	.68	.69	.69	.68	.69	.69	.69

(a) Accuracy before SM, (b) Accuracy after SM

TABLE VII

Precision before and after scenario method was applied

PT	Gen=20		Gen=40		Gen=60		Gen=80		Gen=100	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
0	.41	.59	.43	.57	.47	.62	.46	.64	.50	.58
10	.42	.59	.46	.60	.47	.62	.46	.63	.51	.59
20	.42	.59	.46	.60	.48	.62	.47	.64	.51	.58
30	.43	.59	.47	.60	.49	.63	.53	.69	.52	.61
40	.44	.60	.47	.61	.50	.64	.55	.69	.54	.62
50	.46	.61	.51	.62	.54	.65	.59	.67	.55	.62
60	.54	.66	.57	.67	.58	.66	.60	.67	.59	.64
70	.61	.70	.62	.70	.64	.69	.64	.69	.66	.71
80	.69	.74	.69	.74	.71	.74	.70	.74	.71	.75
90	.72	.75	.72	.76	.74	.77	.72	.76	.74	.76
100	.72	.76	.72	.76	.75	.77	.73	.76	.74	.78

(a) Precision before SM, (b) Precision after SM

TABLE VIII

Number of pruned trees by scenario method

PT	Gen=20	Gen=40	Gen=60	Gen=80	Gen=100
0	47.3	53.1	77.0	53.0	95.4
10	48.1	58.8	92.3	88.2	104.8
20	49.4	60.9	89.3	93.8	114.0
30	52.6	63.7	135.7	112.8	159.1
40	116.4	144.7	175.2	162.2	207.5
50	132.4	190.1	204.9	209.6	245.3
60	195.5	237.8	252.1	246.5	306.7
70	238.3	313.7	353.6	324.5	443.3
80	388.4	478.5	481.7	482.5	546.5
90	417.0	521.0	558.4	521.7	575.7
100	427.1	535.9	575.5	556.9	591.9

TABLE IX

Tree size redution produced by scenario method

PT	Gen=20	Gen=40	Gen=60	Gen=80	Gen=100
0	31%	30%	26%	23%	26%
10	28%	32%	26%	23%	26%
20	29%	31%	24%	21%	26%
30	32%	28%	24%	16%	21%
40	31%	28%	25%	16%	20%
50	31%	28%	24%	18%	21%
60	29%	32%	22%	20%	20%
70	29%	32%	22%	20%	19%
80	34%	35%	27%	27%	24%
90	37%	37%	30%	28%	28%
100	36%	40%	33%	36%	36%

REFERENCES

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. United States of America: Wadsworth International Group, 1984.

[2] J. R. Quinlan, "Simplifying decision trees," in *International Journal of Machine studies*, 1986, pp. 221–234.

[3] J. R. Quinlan., *C.45 Programs for Machine Learning*. San Mateo California: Morgan Kaufmann, 1993.

[4] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, 1992.

[5] P. Angeline, "Genetic Programming and Emergent Intelligence," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. MIT Press, 1994, ch. 4, pp. 75–98.

[6] P. Nordin, F. Francone, and W. Banzhaf, "Explicitly Defined Introns and Destructive Crossover in Genetic Programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, J. P. Rosca, Ed., Tahoe City, California, USA, 9 July 1995, pp. 6–22.

[7] T. Soule and J. A. Foster, "Code size and depth flows in genetic programming," in *Proceeding of the Second Annual Conference*, J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. R. Riolo, Eds. Morgan Kaufmann, 1997, pp. 313–320.

[8] W. B. Langdon, "Quadratic bloat in genetic programming," in *Proceedings of the Genetic and evolutionary Computation Conference (GECCO-2000)*, 2000, pp. 451–458.

[9] T. Soule and J. A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," vol. 6, no. 4, Winter 1998, pp. 293–309. [Online]. Available: citeseer.ist.psu.edu/article/soule98effects.html

[10] R. Poli, "A simple but theoretically-motivated method to control bloat in genetic programming," in *Proceedings of the 6th European Conference*. Springer-Verlag, 2003, pp. 204–217.

[11] H. Iba, H. de Garis, and T. Sato, "Genetic Programming using a Minimum Description Length Principle," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. MIT Press, 1994, pp. 265–284.

[12] T. Soule, *Code Growth in Genetic Programming*. Moscow, Idaho, USA: PhD Thesis, College of Graduate Studies, University of Idaho, 15 May 1998.

[13] J. Eggermont, J. N. Kok, and W. A. Kosters, "Detecting and pruning introns for faster decision evolution," in *The 8th International Conference of Parallel Problem Solving from Nature*. Springer-Verlag, 2004.

[14] R. Kohavi and F. Provost, "Glossary of terms," in *Edited for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, vol. 30, February 1998.

[15] E. P. Tsang, J. Li, and J. Butler, "Eddie beats the bookies," in *International Journal of Software, Practice Experience*, ser. 10, vol. 28. Wiley, August 1998, pp. 1033–1043.

[16] E. P. Tsang, P. Yung, and J. Li, "Eddie-automation, a decision support tool for financial forecasting," in *Journal of Decision Support Systems, Special Issue on Data Mining for Financial Decision Making*, ser. 4, vol. 37, 2004.

[17] E. P. Tsang, S. Markose, and H. Er, "Chance discovery in stock index option and future arbitrage," in *New Mathematics and Natural Computation, World Scientific*, ser. 3, vol. 1, 2005, pp. 435–447.

[18] W. F. Sharpe, G. J. Alexander, and J. V. Bailey, *Investments*. Upper Saddle River, New Jersey 07458: Prentice-Hall International, Inc, 1995.

[19] N. Chomsky, *Aspects of the theory of syntax*. Cambridge M.I.T. Press, 1965.

[20] A. Teller and M. Veloso, "Neural programming and an internal reinforcement policy," in *In fisrt international Conference on Simulated Evolution and learning*. Springer-Verlag, 1996, pp. 279–286.

[21] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," in *Machine Learning*, vol. 30. 195-215, 1998.