LEGAL-Tree: A Lexicographic Multi-objective Genetic Algorithm for Decision Tree Induction

Márcio P. Basgalupp[1], Rodrigo C. Barros[2], André C.P.L.F. de Carvalho[1], Alex A. Freitas[3] and Duncan D. Ruiz[2]

[1]University of São Paulo
Av. Trabalhador São Carlense 400
P.O. Box 668, São Carlos – SP, Brazil
+55 16 3373.9646
{marciopb, andre}@icmc.usp.br

[2]Pontifical Catholic University of RS
Av. Ipiranga 6681, Porto Alegre – RS, Brazil
+55 51 3320.3611
{rodrigo.barros, duncan}@pucrs.br

[3]University of Kent
Canterbury, Kent, CT2 7NF, United Kingdom
+44 1227 827220
A.A.Freitas@kent.ac.uk

ABSTRACT

Decision trees are widely disseminated as an effective solution for classification tasks. Decision tree induction algorithms have some limitations though, due to the typical strategy they implement: recursive top-down partitioning through a greedy split evaluation. This strategy is limiting in the sense that there is quality loss while the partitioning process occurs, creating statistically insignificant rules. In order to prevent the greedy strategy and to avoid converging to local optima, we present a novel Genetic Algorithm for decision tree induction based on a lexicographic multi-objective approach, and we compare it with the most well-known algorithm for decision tree induction, J48, over distinct public datasets. The results show the feasibility of using this technique as a means to avoid the previously described problems, reporting not only a comparable accuracy but also, importantly, a significantly simpler classification model in the employed datasets.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Induction and Knowledge Acquisition – *decision trees induction, multi-objective genetic algorithms.*

General Terms

Algorithms.

Keywords

Lexicographic Multi-Objective Genetic Algorithms, Decision Tree Induction, Data Mining, Evolutionary Algorithms.

1. INTRODUCTION

Decision trees (DT) are a powerful and widely-used technique for data mining classification tasks. This can be explained by several factors, among them [18]: (i) *ease of understanding*, due to the knowledge representation method – a decision tree is a graphical representation and can be easily converted into a set of rules written in a natural language; (ii) *robustness* to the presence of noise; (iii) *availability of computationally inexpensive* DT induction algorithms, even for very large training datasets; and (iv) *good handling of irrelevant or redundant attributes*.

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

Some well-known algorithms for DT induction are Quinlan's ID3 [19] and C4.5 [20] and Breiman et al.'s CART (Classification and Regression Trees) [3]. Such algorithms typically rely on a greedy, top-down, recursive partitioning strategy for the growth of the tree. There are at least two major problems related to these characteristics: (i) the greedy strategy usually produces locally (rather than globally) optimal solutions, (ii) recursive partitioning iteratively degrades the quality of the dataset for the purpose of statistical inference, because the larger the number of times the data is partitioned, the smaller the data sample that fits the current split becomes, making such results statistically insignificant, contributing to a model that overfits the data.

To overcome these difficulties, different approaches were suggested, though not without drawbacks. These approaches can be divided into two basic threads: (i) multiple splits at non-terminal nodes; and (ii) multiple tree generation, so as to combine different views over the same domain. Approaches based on (i) result in the so-called Option Trees [3], which are not, per se, a DT. An Option Tree is hard to interpret, hurting what is probably one of the most important characteristics of DTs. Approaches based on (ii) can aggregate different trees' classifications into a single one, according to a given criterion. Well-known ensemble methods such as random forests [18], boosting and bagging [21] are the most common approaches based on (ii).

It is well-known that, in general, an ensemble of classifiers improves predictive accuracy by comparison with the use of a single classifier. On the other hand, the use of ensembles also tends to reduce the comprehensibility of the model. A single comprehensible predictive model can be interpreted by an expert, but it is not practical to ask an expert to interpret an ensemble consisting of a large number of comprehensible models. In addition to the obvious problem that such an interpretation would be time-consuming and tedious to the expert, the classification models being combined in an ensemble are often to some extent inconsistent with each other – this inconsistence is necessary to achieve diversity in the ensemble, which in turn is necessary to increase the predictive accuracy of the ensemble. Considering that each model can be regarded as a hypothesis to explain predictive patterns hidden in the data, this means that an ensemble does not represent a single coherent hypothesis about the data, but rather a large set of mutually inconsistent hypothesis, which in general would be too confusing for an application-domain expert [9].

In order to alleviate the inherent problems of DT induction, and to avoid the shortcomings of the current approaches, we present a novel algorithm based on the Genetic Algorithms paradigm (GA)

[11]. Instead of local search, GAs perform a robust global search in the space of candidate solutions [7]. Through this evolutionary approach where each individual is a decision tree, we increase the chances of converging to a globally near-optimal solution. Furthermore, our approach results into a single decision tree, preserving the comprehensibility of the classification model.

We focus on comprehensible models because model comprehensibility is highly important in many data mining applications [20, 24], even though it is usually underrated by the data mining community. Indeed, a comprehensible model has advantages such as [9]: (a) improving the user's confidence in the prediction; (b) giving the user's new insights about the data and/or the application domain; and (c) potentially detecting errors in the model or in the data. To the best of our knowledge this is the first paper to propose a GA based on a lexicographic multi-objective criterion for the problem of decision tree induction. In essence, the proposed GA optimizes a decision tree's predictive accuracy as a higher-priority objective and tree size as a lower-priority objective. Computational results in 7 datasets have shown that, overall, the proposed GA improves decision tree simplicity without significantly reducing predictive accuracy, by comparison with the trees built by the very well-known J48 algorithm [24], WEKA's implementation of C4.5 [20].

This paper is organized as follows. Section 2 describes our novel algorithm's characteristics. Section 3 reports our experimental methodology. Section 4 reports the experimental results. Section 5 discusses related work and Section 6 presents our conclusions and some future research directions.

2. THE LEGAL-TREE ALGORITHM

We have created a new evolutionary method for inducing decision trees called LEGAL-Tree (LExicographical Genetic Algorithm for Learning decision Trees). The next sections detail the main steps of this algorithm.

2.1 Solution Representation

While GA applications generally rely on binary strings for representing each individual, we adopt the tree representation, because it seems logical that if each individual represents a decision tree, the solution is best represented as a tree. Thus, each individual is a set of nodes, with each node being either a non-terminal or a terminal (leaf) node. Each non-terminal node contains an attribute, and each leaf node contains a predicted class. A set of edges linking each node with their children is also a part of the tree representation. If a node *x* represents a categorical attribute, there are *n* edges, where *n* is the total number of categories (values) the attribute owns. If a node *x* represents a numeric continuous attribute, there is a binary split according to a given threshold automatically chosen by the algorithm.

We have chosen to deal with missing values through the following simple strategy: for categorical attributes, the missing values will be replaced by the mode (in training set) of all attribute values; and for numeric attributes, the missing values will be replaced by the mean (in training set) of all attribute values.

2.2 Generating the initial population

Although a random individual generation is the most common technique for generating the initial population in GA applications, we believe that by incorporating task-specific knowledge for inducing decision trees (i.e., knowledge about the meaning of a decision tree in the context of a classification task) we can derive better solutions, or at least equally good solutions in fewer generations. Our strategy for incorporating task-specific knowledge into the GA is described in Algorithm 1.

A decision stump is the simplest case of decision trees which consists of a single decision node and two predictive leaves [10]. We have extended such concept for categorical attributes, where each edge that represents an attribute category (value) will lead to a predictive leaf. Thus, the pseudo-code shown in Algorithm 1 will basically generate a set of decision stumps (in fact, 10 X *n* decision stumps, where *n* is the number of predictive attributes of the dataset). By dividing the training dataset in 10 different pieces, we hope to achieve a certain degree of heterogeneity for the decision stumps involving numeric attributes, because they will be essential in the generation of the initial population. Such process of generating decision stumps is far from being random, since the thresholds for numeric attributes are deterministically chosen. More precisely, for setting the threshold of numeric attributes, we use the *information gain* measure [18]. The final step for generating the initial population is the aggregation of the different decision stumps previously created.

Algorithm 1: Decision Stumps Generation

```
1.    let x be the training dataset
2.    divide x in 10 different pieces
3.    let dsList be a list of decision stumps
4.    let y_j represent the j^th piece of x
5.    for each j do
6.      let a_i be the i^th dataset attribute
7.      for each i
8.        if a_i is numeric
9.          threshold = infoGain(y_j,a_i)
10.         dsList.add(new numericDS(y_j,a_i,threshold))
11.       else
12.         dsList.add(new categoricalDS(y_j,a_i))
```

The user sets the depth for the trees that will be part of the initial population, and the algorithm randomly combines different decision stumps in order to create a tree with the informed depth. For instance, consider that three randomly picked decision stumps (A, B and C) are merged into one complete tree of depth 2. The randomly selected root is decision stump A, while its two child nodes are replaced by B and C. All trees generated in the initial population are complete trees resulting from the merging of randomly picked decision stumps. However, the trees that will populate the next generations will not necessarily be complete due to the genetic operations that will affect their structures.

The two main advantages of this approach for generating the initial population of decision trees are that: (a) the thresholds for numerical attributes in non-terminal nodes are chosen in a data-driven manner based on information gain, rather than randomly chosen; and (b) the class associated with a leaf node is always the most frequent class among the examples covered by the leaf node, rather than a randomly chosen class. Both advantages are a result of incorporating general knowledge about the classification task being solved (rather than knowledge specific to a given application domain like finance or medicine) into the GA, which tends to increase its effectiveness.

2.3 Lexicographic multi-objective fitness function

It is also usually accepted that the knowledge discovered by a data mining algorithm should be not only accurate but also comprehensible to the user [24], where comprehensibility (or

simplicity) is usually estimated by the size of the classifier – smaller classifiers are assumed to be preferable, other things being equal. This is also justified by Occam's razor [5], a principle often used in science, which essentially says that, out of multiple hypothesis that are equally consistent with the data, one should choose the simplest hypothesis. In this context, we present a lexicographic multi-objective approach, which basically consists of assigning different priorities to different objectives and then focusing on optimizing them in their order of priority [7]. In this work, we only consider *predictive accuracy* and *tree size* as objectives, since they are the most common measures for evaluating decision trees.

Consider the following example. Let *x* and *y* be two decision trees and *a* and *b* two evaluation measures. Besides, consider that *a* has the highest priority between the measures and that t_a and t_b are tolerance thresholds associated with *a* and *b* respectively. The lexicographic approach works according to the following analysis: if $|a_x - a_y| > t_a$ then it is possible to establish which decision tree is "better" considering only measure *a*. Otherwise, the lower-priority measure *b* must be evaluated. In this case if $|b_x - b_y| > t_b$ then the fittest tree considering *x* and *y* can be decided only by considering measure *b*. If it still is the case the difference between values falls within the assigned threshold t_b, the best value of the higher-priority measure *a* is used to determine the fittest tree.

There are other approaches for coping with multi-objective problems. For instance, there is the conventional weighted-formula, in which a single formula containing each objective adjusted by weights reduces the multi-objective problem into a single-objective one. Also, there is the Pareto approach, where instead of transforming a multi-objective problem into a single-objective one and then solving it by using a single-objective search method, one should use a multi-objective algorithm to solve the original multi-objective problem in terms of Pareto dominance [4]. A weighted-formula approach suffers from several known problems, namely *the "magic number" problem* (setting the weights in the formula is an ad-hoc procedure), *"mixing apples and oranges"* (adding up non-commensurable criteria such as accuracy and tree size) and *mixing different units of measurement* (operating on different objective scales and introducing a bias when choosing a suitable normalization procedure) [8]. A Pareto approach also has its drawbacks, with one of the most expressive ones being the difficulty of choosing a single best solution to be used in practice. Another shortcoming of such approach is the difficulty in handling different-priority objectives, that is, cases where one objective is significantly more important than the other to the user. In particular, in the classification task of data mining, most researchers and users agree that predictive accuracy is usually considered more important than tree size for decision trees, but a Pareto approach would not be able to recognize that accuracy is more important than size. Consider, for instance, two decision trees t_1 and t_2, where t_1 has 90% of accuracy and 20 nodes, and t_2 has 70% of accuracy and 15 nodes. Most data mining researchers and users would clearly prefer tree t_1 over t_2. However, the Pareto approach would consider that none of these two trees dominates the other.

A lexicographic approach does not suffer from the mentioned problems (in the above example, it would choose tree t_1 over t_2 for any sensible accuracy tolerance threshold < 20%, respecting the user's preferences), and it is conceptually simple and easy to use. Note that, to the best of our knowledge, a lexicographic approach has not yet been used for evolutionary decision tree induction.

2.4 Selection

LEGAL-Tree uses the popular and effective tournament selection method. It also implements the elitism technique, i.e., it preserves a percentage of individuals based on their fitness values.

2.5 Crossover

LEGAL-Tree implements the crossover operation as follows. First, two individuals randomly picked among the selected ones (selection operation) will exchange sub-trees. According to a randomly number which varies from 1 to *n* (number of nodes), LEGAL-Tree performs an adapted pre-order tree search method, visiting recursively the root node and then its children from left to right. For numeric nodes, such search method is equivalent to the traditional binary pre-order search. If the attribute is categorical and has more than 2 children, the search method will visit each child from left to right, according to an index that identifies each child node. After identifying the nodes sorted by the randomly picked number in both parents, LEGAL-Tree will exchange the whole sub-tree represented by the sorted node.

Consider two individuals presented as "parents" in Figure 1. For Parent 1, the sorted node was node C, while for Parent 2 it was node M. After this, the crossover operation will make sure two children individuals are created keeping the tree structure from one of the parents but with the sorted nodes being exchanged. Child 1 keeps the tree structure from Parent 1, but inherits node M from Parent 2. Similarly, Child 2 keeps the structure from Parent 2 but inherits node C from Parent 1. By exchanging the whole sub-trees from the sorted node and not only specific nodes, we avoid problems such as domain irregularities, because each edge refers to attribute characteristics that are represented by a node. It does not prevent, however, redundant rules and inconsistencies. Section 2.7 details how LEGAL-Tree addresses such issues.

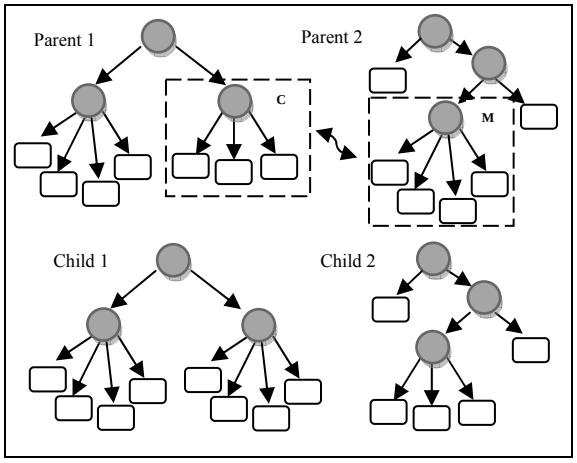

Figure 1. Crossover between Parents 1 and Parent 2.

2.6 Mutation

LEGAL-Tree implements two different strategies for mutation of individuals. The first one considers the exchanging of a whole sub-tree, selected randomly from an individual, by a leaf node representing the most frequent class attribute value among the examples covered by that leaf. The second strategy replaces a randomly selected leaf node in an individual by a decision stump created in the population initialization process. Figure 2 depicts both strategies abovementioned. Such strategies aim at increasing or diminishing the individual's size, increasing the heterogeneity of the population and avoiding convergence to local optima.

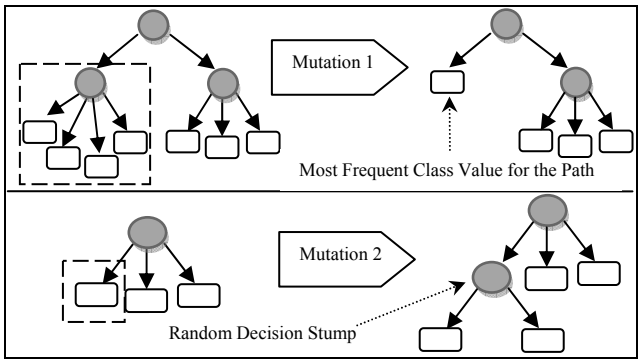

Figure 2. Two strategies for mutating individuals.

2.7 Candidate Solution Validity Issues

After crossover and mutation operations, there are cases where evolution generates inconsistent scenarios. For instance, consider that a sub-tree from tree *a* has been replaced by a sub-tree from tree *b* generating a child individual during the crossover process. If the new sub-tree has a node representing an attribute already specified by an ancestral node, actions must be taken to avoid redundant rules or inconsistent threshold intervals. If the node's attribute is categorical and an ancestor represents the same attribute, then we have some redundancy which can be eliminated. In such case, we have chosen to replace this sub-tree by a leaf node representing the most frequent class value for that given leaf node. Similarly, if the node is numeric and its threshold interval is inconsistent with some ancestral node, then such interval must be adjusted in order not to generate rules that are not satisfied by any data instance. This adjustment also considers task-specific knowledge because it recalculates the information gain for defining the new threshold interval value. Such problems are addressed by LEGAL-Tree through a filtering process applied after the crossover and mutation operations.

Another problem that may occur after the crossover and mutation operations is when leaf nodes stop representing the most frequent class attribute value. Such irregularity is also addressed by LEGAL-Tree during this filtering process, which recalculates the correct class attribute value for the modified leaf, considering the training set. Dealing with such issues was an essential step in the construction of LEGAL-Tree, since we can converge faster to a near-optimal solution by implementing this filtering process.

3. EXPERIMENTAL METHODOLOGY

We have applied the proposed approach of inducing decision trees through a lexicographic multi-objective genetic algorithm to several classification problems collected in the UCI machine learning repository [17], including Credit-a, Credit-g, Colic, Diabetes, Glass, Hepatitis and Sonar (Table 1).

First, we have analyzed the predictive accuracy and tree size obtained by the J48 decision tree induction algorithm (WEKA's implementation of C4.5) [24] in each dataset. All parameter settings used are the algorithm's default ones, and we have used 10-fold cross-validation, a widely disseminated approach for validating classification models. In each of the 10 iterations of the cross-validation procedure, the training set is divided into sub-training and validation sets, which are used to produce the decision stumps (sub-training set), filtering process (sub-training) and fitness function (sub-training or validation set, according to the approach). Each set represents 50% of the full training set.

Table 1. Datasets specification.

Dataset	Instances	Numeric Attributes	Categorical Attributes	Classes
Colic	368	7	15	2
Credit-a	690	6	9	2
Credit-g	1000	7	14	2
Diabetes	768	8	0	2
Glass	214	9	0	6
Hepatitis	155	6	13	2
Sonar	208	60	0	2

We have executed LEGAL-Tree with the parameter values described in Table 2, but with two different fitness measures. For the first case, L1, we have used as fitness measures the accuracy of an individual in the validation set and its tree size, in this priority order, with thresholds of 1% for accuracy and 2 nodes for tree size. Analysis of the results with this L1 version of LEGAL-Tree revealed that the algorithm was overfitting the validation set, since the "learning stage" of our algorithm does not make extensive use of the sub-training set, especially by comparison with the extensive use of the entire training set by C4.5. Hence, to mitigate this problem, we have created a second configuration, L2, where we have used as fitness measures the accuracy in the validation set, the accuracy in the sub-training set and tree size, in this priority order, and the thresholds of 2% for both accuracies and 2 nodes for tree size. This approach preserves the highest-priority of accuracy in validation set but introduces an additional selective pressure to maximize accuracy in the sub-training set as a second criterion (still more important than tree size), and so it helps avoiding overfitting to the validation set.

Table 2. LEGAL-Tree parameters for the experiments.

Parameter	Value
Initial population max depth	3
Population size	500
Improvement rate	3%
Max number of generations	500
Tournament rate	0.6%
Elitism rate	5 %
Crossover rate	90%
Mutation rate	5%

The parameter values in Table 2 were based on our previous experience in using evolutionary algorithms, and we have made no attempt to optimize parameter values, a topic left for future research. *Improvement rate* (i.e., rate of the maximum number of generations without fitness improvement) and *Max number of generations* are the algorithm's stopping criteria. Due to the fact that GA is a non-deterministic technique, we have run LEGAL-Tree 30 times for each one of the 10 training/test set folds generated by the 10-fold cross-validation procedure. These folds were the same ones used by J48, to make the comparison between the algorithms as fair as possible. After running LEGAL-Tree over the 7 datasets presented in Table 1, we have calculated the average and standard deviation of the 30 executions for each fold, and then the average of the ten folds. Considering J48, we have calculated the averages and standard deviations for the ten folds.

A few tests were executed to assess the statistical significance of the differences observed in the experiments. The data used consist of both the mean classification predictive accuracy and the tree size for each test fold. The statistical test used was the corrected paired t-test [16], with a significance level of $\alpha = 0.05$ and 9 degrees of freedom.

4. EXPERIMENTAL RESULTS

Table 3 shows the mean predictive accuracy (in the test set) and the tree sizes for J48 and the 2 versions of LEGAL-Tree: L1 and L2. A summarized comparison between the results of LEGAL-Tree and J48, in terms of the previously mentioned statistical test of significance, is shown in Table 4. In this table, each cell $C_{i,j}$ contains the label of the datasets in which technique *i* is significantly better than technique *j* (Table 4a) or *i* is significantly worse than *j* (Table 4b). The dataset labels are as follows {C_o}olic, {C_a}redit-a, {C_g}redit-g, {D}iabetes, {G}lass, {H}epatitis and {S}onar.

Table 3. Mean predictive accuracy (%), tree size (in number of nodes) and respective standard deviation (in parenthesis).

Dataset	Predictive Accuracy			Tree Size		
	J48	L1	L2	J48	L1	L2
Colic	85.30 (4.50)	82.72 (3.24)	84.72 (2.69)	8.10 (2.02)	15.37 (7.73)	8.62 (5.01)
Credit-a	86.08 (3.56)	85.76 (1.49)	85.45 (1.01)	28.10 (8.58)	6.37 (4.04)	4.74 (2.55)
Credit-g	70.50 (3.41)	71.24 (2.39)	71.86 (2.57)	117.40 (28.20)	13.82 (7.38)	16.23 (10.73)
Diabetes	73.83 (5.37)	72.94 (3.10)	73.69 (3.09)	37.40 (12.38)	16.05 (6.22)	34.08 (25.53)
Glass	66.75 (7.53)	60.28 (7.03)	62.94 (6.07)	44.20 (5.23)	18.55 (6.55)	22.59 (10.06)
Hepatitis	83.79 (6.87)	78.97 (6.63)	81.13 (4.29)	17.80 (4.40)	15.99 (4.87)	20.39 (6.58)
Sonar	71.16 (6.74)	69.79 (8.87)	72.22 (8.79)	29.20 (3.40)	27.33 (8.41)	45.93 (12.57)

As it can be seen in Table 4, LEGAL-Tree's lexicographic L1 version obtained a predictive accuracy significantly worse than J48 in just one dataset – in the other 6 datasets there was no significance difference. On the other hand, L1 obtained a significantly simpler (smaller) decision tree than J48 in 4 datasets, and the opposite was true in just one dataset. Hence, L1 obtained competitive results with respect to J48.

Table 4. L1 and L2 significantly better (a) or worse (b) than J48 according to the corrected paired t-test.

(a)		J48
Accuracy	**L1**	- - - - - - -
	L2	- - - - - - -
Tree Size	**L1**	-$C_a$$C_g$DG- -
	L2	-$C_a$$C_g$-G- -

(b)		J48
Accuracy	**L1**	C_o- - - - - -
	L2	- - - - - - -
Tree Size	**L1**	C_o- - - - - -
	L2	- - - - - - S

LEGAL-Tree's lexicographic L2 version performed particularly well. Its predictive accuracy was statistically indistinguishable from J48's one in all 7 dataset, but L2 obtained a significantly simpler decision tree in 3 datasets, whilst the opposite was true in only one dataset. This shows that the lexicographic approach used in L2, based on choosing the smallest out of competing trees when their accuracy is not significantly different, is working well, leading to a significant reduction in the size of the DT produced by the system in approximately half of the datasets, without sacrificing accuracy in any dataset, by comparison with J48.

As a final note, we have also stored the execution time for both algorithms, and LEGAL-Tree was, as expected for being an evolutionary algorithm, slower than J48. Nevertheless, our current efforts are to maximize LEGAL-Tree's performance.

5. RELATED WORK

One branch of Evolutionary Algorithms (EAs), *Genetic Programming* (GP), has been largely used as an induction method for decision trees, in works such as [1, 6, 12, 13, 14, 22, 23, 25, 26]. Koza [12] was the pioneer in inducing decision trees with GP, converting the attributes of Quinlan's weather problem [19] into functions. In [25], it is proposed a tool which allows the user to set different parameters for generating the best computer program to induce a classification tree. This work takes into account the cost-sensitivity of misclassification errors, in a multi-objective approach to define the optimal tree. In [6], it was implemented an algorithm of tree induction in the context of bioinformatics, in order to detect interactions in genetic variants. Bot and Langdon [1] proposed a solution for linear classification tree induction through GP, where an intermediate node is a linear combination of attributes. In [13] the authors proposed different alternatives to represent a multi-class classification problem through GP, and later extended their work [14] for dealing with nominal attributes. In [22, 26], the authors proposed the design of binary classification trees through GP, and in [23] it is described a GP algorithm for tree induction that considers only binary attributes. In [15] GP with lexicographic fitness is proposed, but that work does not involve decision tree induction.

At this point it is important to discuss an issue of terminology. In GP each individual of the population is a computer program (containing data and operators/functions applied to that data), generally with its structure being represented in the form of trees. Ideally a computer program should be generic enough to process any instance of the target problem (in our case, the evolved program should be able to induce decision trees for any classification dataset in any application domain). It should be noted, however, that in the above GP works each individual is a decision tree for the dataset being mined, and not a generic computer program responsible for generating decision trees from any given classification dataset. This is also the approach followed in this paper. Although the above work are presented as GP, we prefer to call our EA a Genetic Algorithm (GA), to emphasize the fact that the EA is evolving just a decision tree (which is not a program in the conventional sense), i.e., just a solution to one particular instance of the problem of inducing decision trees from a given dataset; rather than evolving a generic computer program (like C4.5) that can induce decision trees from any given dataset. Regardless of terminology issues, all these GP algorithms differ from ours in an important way: we propose a multi-objective GA based on the lexicographic criterion.

6. CONCLUSION AND FUTURE WORK

Decision trees have been widely used to build classification models which are easy to comprehend, mainly because such models resemble the human reasoning. Recall that the comprehensibility of the discovered classification model is important in many applications where decisions will be made by a human being based on the discovered model. Hence, there is a clear motivation to discover decision trees that are not only accurate but also relatively simple.

Traditional decision tree induction algorithms which rely on a recursive top-down partitioning through a greedy split evaluation are relatively fast but susceptible to converging to local optima, while an ideal algorithm should choose the correct splits in order to converge to a global optimum. With this goal in mind, we have proposed a novel Genetic Algorithm for inducing decision trees called LEGAL-Tree. LEGAL-Tree avoids the greedy search performed by conventional decision tree induction algorithms, and performs instead a global search in the space of candidate decision

trees. In addition, it differs from previously proposed evolutionary algorithms for decision tree induction in a very important aspect: the fitness function. While other approaches typically rely on a single objective evaluation (possibly collapsing multiple objectives into a single objective using a weighted formula) or on an evaluation based on Pareto dominance, we propose a lexicographic approach, where multiple measures are evaluated in order of their priority. This approach is relatively simple to implement and control and does not suffer from the problems the weighted-formula and Pareto dominance do, as discussed earlier.

LEGAL-Tree's fitness function considers the two most common measures used for evaluating decision trees: classification accuracy and tree size. In experiments with 7 datasets, the first version of LEGAL-Tree was moderately successful, and its second version was quite successful, obtaining significantly better results than the very well-known J48 algorithm overall. More precisely, the former obtained statistically significantly simpler decision trees than the latter in 3 datasets, whilst the opposite was true in just one dataset; and such an overall improvement in simplicity was obtained without any significant loss in predictive accuracy in any dataset. This is a clearly positive result, which is also supported by the aforementioned Occam's Razor, a principle very often used in data mining and science in general [5].

Some possibilities for future research are as follows. First, the setting of input parameters for LEGAL-Tree could be done through a supportive GA, helping to achieve convergence to a global optimum. In addition, we are working on alternative methods for mutation and dealing with categorical missing values. A future mutation implementation will be a tree pruning method which will have a higher probability of occurring.

7. ACKNOWLEDGMENTS

Our thanks to *Fundo de Amparo à Pesquisa do Estado de São Paulo* (FAPESP) for supporting this research.

8. REFERENCES

[1] Bot, M.C.J., Langdon, W.B. 2000. Application of genetic programming to induction of linear classification trees, Proceedings of the Third European Conference on Genetic Programming, 247–258.

[2] Breiman, L., Friedman, J., Olshen, R., Stone, C. 1984 Classification and Regression Trees. Wadsworth.

[3] Buntine, W. 1993. Learning Classification Trees. Statistics and Computing, v. 2 (2), 63-73.

[4] Coello, C., Veldhuizen, D., Lamont, G. 2002 Evolutionary Algorithms for Solving Multi-Objective Problems. Springer.

[5] Domingos, P. 1999. The Role of Occam's razor in Knowledge Discovery. Data Mining and Knowledge Discovery, v. 3 (4), 409-425.

[6] Estrada-Gil, J. K. *et al*. 2007. GPDTI: A Genetic Programming Decision Tree Induction Method to find epistatic effects in common complex diseases. Bioinformatics, v. 23, 167-174.

[7] Freitas, Alex A. 2007. A Review of Evolutionary Algorithms for Data Mining. *Soft Computing for Knowledge Discovery and Data Mining*. Springer-Verlag New York, Inc., 79-111.

[8] Freitas, A. A. 2004. A critical review of multi-objective optimization in data mining: a position paper. SIGKDD Explorations Newsletter 6, 2 (Dec. 2004), 77-86.

[9] Freitas, Alex A., Wieser, Daniela C., Apweiler, Rolf. 2008. On the importance of comprehensible classification models for protein function prediction. To appear in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

[10] Freund, Y., Mason, L. 1999. The alternating decision tree learning algorithm. Proceedings of the 16th International Conference on Machine Learning, 124-133.

[11] Goldberg, D. 1989 Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley.

[12] Koza, J.R. 1991. Concept formation and decision tree induction using the genetic programming paradigm. Proceedings of the First Workshop on Parallel Problem Solving from Nature, 124–128.

[13] Loveard, T., Ciesielski, V. 2001. Representing classification problems in genetic programming. Proceedings of the 2001 Congress on Evolutionary Computation, 1070–1077.

[14] Loveard, T., Ciesielski, V. 2002. Employing nominal attributes in classification using genetic programming. Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning, 487 – 491.

[15] Luke, S., and L. Panait. 2002. Lexicographic Parsimony Pressure. In GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference. W. B. Langdon et al, eds. Morgan Kauffman. 829-836.

[16] Nadeau, C., Bengio, Y. 2003. Inference for the generalization error. Machine Learning, v. 52 (3), 239–281.

[17] Newman, D. J., Hettich, S., Blake, C. L., Merz, C. J. 1998. UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[18] Tan, P., Steinbach, M., Kumar, V. 2005 Introduction to Data Mining. Addison-Wesley Longman Publishing.

[19] Quinlan, J. R. 1986. Induction of Decision Trees. Machine Learning, v. 1, 81-106.

[20] Quinlan, J. R., 1993 C4.5: Programs for Machine Learning. Morgan Kaufmann.

[21] Quinlan, J. R., 1996. Bagging, boosting, and C4.5. In Proceedings of the Fourteenth National Conference on Artificial Intelligence.

[22] Shirasaka, M., Zhao, Q., Hammarmi, O., Kuroda, K., Saito, K. 1998. Automatic design of binary decision trees based on genetic programming. Proceedings of the Second Asia-Pacific Conference on Simulated Evolution and Learning.

[23] Tür, G., Güvenir, H.A. 1996. Decision tree induction using genetic programming. Proceedings of the Fifth Turkish Symposium on Artificial Intelligence and Neural Networks.

[24] Witten, I. H., Frank, E. 2005 Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco.

[25] Zhao, H. 2007. A multi-objective genetic programming approach to developing Pareto optimal decision trees. *Decision Support Systems* 43, 3 (Apr. 2007), 809-826.

[26] Zhao, Q., Shirasaka, M. 1999. A study on evolutionary design of binary decision trees, Proceedings of the 1999 Congress on Evolutionary Computation, 1988–1993.