Evolutionary Model Tree Induction

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

[1]Pontifical Catholic University of RS
Av. Ipiranga 6681,
Porto Alegre – RS, Brazil
+55 51 3320 3611
{rodrigo.barros,duncan}@pucrs.br

[2]University of São Paulo
Av. Trabalhador São Carlense 400
São Carlos – SP, Brazil
+55 16 3373 9646
{marciopb,andre}@icmc.usp.br

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

ABSTRACT

Model trees are a particular case of decision trees employed to solve regression problems. They have the advantage of presenting an interpretable output with an acceptable level of predictive performance. Since generating optimal model trees is a NP-Complete problem, the traditional model tree induction algorithms make use of a greedy heuristic, which may not converge to the global optimal solution. We propose the use of the evolutionary algorithms paradigm (EA) as an alternate heuristic to generate model trees in order to improve the convergence to global optimal solutions. We test the predictive performance of this new approach using public UCI datasets, and compare the results with traditional greedy regression/model trees induction algorithms.

Categories and Subject Descriptors

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

Keywords

Model Trees, Evolutionary Algorithms, Data Mining, Multi-Objective optimisation.

1. INTRODUCTION

Within the data mining regression task, model trees are a popular alternative to classical regression methods, presenting good predictive performance and an intuitive interpretable output. Similarly to decision/regression trees, they are structured trees that represent graphically *if-then-else* rules, which seek to extract implicit knowledge from datasets. While decision trees are used to solve classification problems (i.e., the output is a nominal attribute), both model and regression trees are used to solve regression problems (i.e., the output is a continuous value). The main difference between these approaches is that while regression trees have a single value as the output in their leaves (corresponding to the average of values that reach the leaf), model trees hold linear regression models (equations) to calculate the final output.

A model tree is composed by non-terminal nodes, each one representing a test over a dataset attribute, and linking edges that partition the data according to the test result. In the bottom of the tree, the terminal nodes will hold linear regression models, which were built according to the data that reached each given node. Thus, for predicting the target-attribute value for a given dataset instance, we walk along the tree from the root node to the bottom, until a terminal node is reached, and then we apply the corresponding linear model. Model trees result in a clear knowledge representation, providing the user information on how the output was reached (i.e., the *if-then-else* rule that is provided by the tree once we follow the path until a terminal node).

SAC'10, March 22-26, 2010, Sierre, Switzerland.

In contrast, approaches such as neural networks and support vector machines, while more efficient than model trees in terms of predictive performance in many problems, lack on transparency, because they do not provide the user information about how outputs are produced [21].

Model trees are traditionally induced by divide-and-conquer greedy algorithms which are sequential in nature and locally optimal at each node split [9]. Since inducing the best tree is a NP-Complete problem [23], a greedy heuristic may not derive the best overall tree. In addition, 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 becomes the data sample that fits the specific split, leading to results without statistical significance and creating a model that overfits the training data [2].

In order to avoid the instability of the greedy tree-induction algorithms, recent works have focused on powerful ensemble methods (e.g., bagging [5], boosting [13], random forests [6], etc.), which attempt to take advantage of this unstable process by growing a forest of trees from the data and later averaging their predictions. While presenting very good predictive performance, ensemble methods fail to produce a single-tree solution, operating also in a black-box fashion. We highlight the importance of validation and interpretation of discovered knowledge in many data mining applications that can lead to new insights and hypotheses upon the data [12]. Hence, we believe there should be a trade-off between predictive performance and model interpretability, so a predictive system can be useful and helpful in real-world applications.

Evolutionary algorithms are a solid heuristic able to deal with a variety of optimization problems, performing a robust global search in the space of candidate solutions [11]. Evolutionary induction of decision trees for classification tasks is well-explored in the research community. For instance, Basgalupp et al. [3] proposed an evolutionary algorithm for the induction of decision trees, named LEGAL-Tree, which looks for a good trade-off between accuracy and model comprehensibility. Very few works however propose evolving regression/model trees in order to avoid the problems previously mentioned.

In this work, we propose a new algorithm based on the core idea presented in LEGAL-Tree to deal with regression problems, and we test the predictive performance and comprehensibility of this new algorithm using UCI regression datasets [1]. For such, we briefly review model trees in Section 2 and then we present our new approach for evolving model trees in Section 3. The experiments are presented in Sections 4 and 5, where we compare our approach to M5 [20], the most traditional model trees induction algorithm, and to a regression trees induction algorithm called REPTree [25]. We finish this paper with a brief description of related works and our conclusions.

2. MODEL TREES

Suppose we have d predictor variables, $X_1, X_2, \dots, X_d$ and a response continuous variable Y, which is the target of the prediction model. The training set has n records, (x_i, y_i) where x_i is the vector of predictor variable values for the i^{th} training record, i.e., $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ and y_i is the target attribute value for this same i^{th} record. A model tree holds T terminal nodes that will partition the predictor variable space into R_t regions, where $t = 1,2,3 \dots, T$. Each region R_t will predict the value of Y through a multivariate linear regression procedure, created through a least-square method [4]. Considering that each terminal node will have α predictor variables (a subset of d) and β records (a subset of n), the linear regression demands the creation of T $\beta \times \alpha$ *design matrices*, $M_1, M_2, \dots, M_T$ such as the one shown in Equation 1.

$$M_i = \begin{pmatrix} 1 & x_{11} & \dots & x_{1\alpha} \\ 1 & x_{21} & \dots & x_{2\alpha} \\ \dots & \dots & \dots & \dots \\ 1 & x_{\beta 1} & \dots & x_{\beta\alpha} \end{pmatrix} \quad (1)$$

Considering $\varphi = (\omega_0, \omega_1, \dots, \omega_{\alpha-1})^T$ the α-dimensional vector that represents the regression coefficients that minimize the sum of the squared error for the response variable Y, and $y = (y_1, y_2, \dots, y_\beta)^T$ the β-dimensional vector that represents the β values of the response variable Y, we can define the coefficients by solving the matrix equation given in Equation 2.

$$\varphi = (M^T M)^{-1} M^T y \quad (2)$$

Once the coefficients are known, each terminal node will hold a regression model such as the one in Equation 3.

$$y = \omega_0 + \sum_{i=1}^{\alpha} \omega_i x_i \quad (3)$$

Hence, we can notice that model trees are more sophisticated than either regression trees (whose terminal nodes hold the average of all the training examples' attribute values to which the terminal node applies) or linear regression. Since each terminal node will hold a regression model based on the instances that reach that node, model trees can even approximate non-linear problems, which is the case of a wide range of mining applications.

For growing a model tree, most algorithms rely on a greedy top-down strategy for splitting recursively the nodes. These algorithms seek to minimize some error measure that results from testing each attribute for splitting the node. M5 [20] uses the standard deviation as the error measure for choosing the best split at each node. The goal is to maximize the standard deviation reduction (SDR) by testing the possible splits over the training data that reach a particular node, as shown in Equation 4.

$$SDR = sd(D) - \sum_i \frac{|D_i|}{|D|} \times sd(D_i) \quad (4)$$

where $sd(.)$ is the standard deviation, D is the training set portion that reaches the node that is being tested and D_i the training set portion that results from splitting the node according to a given attribute and split value.

The tree induction of model trees through greedy algorithms is sequential in nature and locally optimal at each node split, which means that convergence for a global optimal solution is hardly feasible. In addition, minor modifications in the training set often lead to large changes in the final model due to the intrinsic instability of these algorithms [9]. Ensemble methods were proposed to take advantage of these unstable algorithms by growing a forest of trees from the data and averaging their predictions. While these methods often present a better predictive performance, it is well-known that the use of ensembles tends to reduce the model comprehensibility. Thus, one has to choose between a simple and comprehensible model with reduced predictive performance and models with a higher predictive capability that sacrifice the principles of simplicity and comprehensibility.

In order to find a good trade-off between predictive performance and model interpretability, we present a novel algorithm based on Genetic Algorithms (GAs) [15]. Instead of local search, GAs perform a robust global search in the space of candidate solutions. Through this evolutionary approach where each individual is a model tree, we increase the chances of converging to a globally near-optimal solution. Furthermore, our approach results in a single model tree, preserving the comprehensibility of the regression model. We call our algorithm E-Motion (Evolutionary MOdel Trees InductiON), which is presented in the next section.

3. E-MOTION

Evolutionary model trees induction (E-Motion) is a novel evolutionary algorithm for creating model trees. It is closely related to LEGAL-Tree [2], [3], which induces decision trees through the evolutionary paradigm. We have extended such approach for dealing with regression problems, in order to generate efficient and comprehensible regression models for a variety of data mining applications.

3.1 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 model 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 node. Each non-terminal node contains an attribute, and each leaf node contains a linear regression model. A set of edges linking each node with its respective children is also a part of the tree representation. There are two distinct possible cases of node relations: (i) the relationship between a categorical node and its children: if a node *x* represents a categorical attribute, there will be *n* edges, where *n* is the total number of categories the attribute owns; and (ii) the relationship between a numeric continuous node and its children: if a node *x* represents a numeric continuous attribute, there will be a binary split according to a given threshold chosen by the algorithm.

3.2 Initial Forest

The initial population generation of model trees implemented by E-Motion is twofold. First, the algorithm produces basic trees, based on each dataset attribute. For categorical attributes, the basic tree is composed by a root node that holds the test over that given attribute, and an edge for each category the attribute owns. The leaf nodes initially do not compute the linear regression models for these basic trees, because they will not be used for prediction purposes at this point. For numeric attributes, E-Motion seeks to incorporate task-specific knowledge in order to derive reasonable threshold values. This strategy is depicted in Algorithm 1.

Five different basic trees are created for each numeric attribute of the dataset. The first one uses the SDR of the entire training set to define the best threshold value. The other four make use of the thresholds given by the SDR of four different partitions of the training set. This approach has two main advantages: (a) we define the thresholds in a data-driven manner (i.e., by using the standard deviation reduction for selecting interesting threshold values), instead of selecting random values, which is the case of most evolutionary approaches; and b) we can achieve a certain degree of heterogeneity by partitioning the training set into different pieces, increasing the chances of selecting a good threshold value. Both advantages are a result of incorporating general knowledge about the regression 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.

Algorithm 1: Numeric basic trees generation.

```
1.    let x be the training dataset
2.    let a[i] be the i^th attribute of x
3.    for each a[i] do
4.      root = a[i]
5.      threshold = SDR(a[i],x)
6.      new basicTree(root,threshold)
7.      divide x in 4 different pieces
8.      let y[k] be the k^th piece of x
9.      for each y[k]
10.       threshold = SRD(a[i],y[k])
11.       new basicTree(root,threshold)
```

The second step for generating the initial forest is the aggregation of different basic trees that were previously created. The user sets the maximum depth value for the trees that will be part of the initial forest, and E-Motion randomly combines different basic trees so as to create a tree with depth that can range from 1 to the maximum depth value informed.

3.3 Multi-objective fitness function

After generating an initial forest, E-Motion evaluates each single tree in terms of its predictive performance. It makes use of two error measures to evaluate the fitness of each model tree: root mean-squared error (*RMSE*) and mean absolute error (*MAE*). The mean-squared error is the most common measure to evaluate numeric predictions. We take the square root to give it the same dimensions as the predicted value itself. The *RMSE* is given by

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \qquad (5)$$

where y_i is the actual value of the target attribute and $\hat{y}_i$ is the estimated value of the target attribute. *MAE*, which is another common choice for evaluating regression problems, is given by

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \qquad (6)$$

It is known that any given evaluation measure is biased to some extent. *RMSE* for instance is quite sensitive to outliers, while *MAE* treats all size of errors evenly according to their magnitude. It is clear that each evaluation measure can serve a different purpose and its use depends on a large extent upon the objectives of the prediction system user. We have chosen these two error measures because they are the most common for regression problems, and even though they present significant differences when dealing with particular cases, it turns out that in most practical situations the best numeric prediction method is still the best for whatever error measure used.

Even though predictive performance is clearly of great importance and has been the primary focus of researchers when developing prediction models, we believe it is not sufficient to indicate a robust predictive approach. *Comprehensibility* or the capacity of explaining to the end-user how a prediction was obtained is crucial in several applications. Understanding the predictions made by a model helps the end-user to get more confidence in the prediction, and more importantly, can provide the basis for the end-user to have new insight about the data, confirming or rejecting hypotheses previously formed. A comprehensible model can even allow the end-user to detect errors in the model or in the data [12].

Burgess and Lefley [8] suggest that an evolutionary algorithm that optimises a single evaluation measure is faded to degrade the other measures, and that a fitness function that is not tied to one particular measure may present more acceptable overall results. Based on such assumption, we have decided to use three different measures to evaluate how fit an individual is: *RMSE* and *MAE* as error measures and tree size (number of nodes) as a measure of model comprehensibility, assuming that smaller trees are more comprehensible than larger ones. We emphasize that E-Motion is a multi-objective evolutionary algorithm that seeks a trade-off between predictive performance and comprehensibility. There are several strategies for coping with multi-objective optimization. Freitas [10] discusses three common approaches: (i) weighted-formula; (ii) Pareto dominance; and (iii) lexicographic analysis.

The lexicographic approach seems to be an interesting choice considering that it recognizes the non-commensurability of the different criteria, and it allows the user to determine which criteria is more important without the need of identifying the correct weight of each measure, while preserving the simplicity of the weighted-formula approach and returning a single solution as the fittest one. E-Motion makes use of the lexicographic analysis among *RMSE*, *MAE* and *tree size*. The priorities of each measure and the tolerance thresholds are detailed in the experimental methodology, Section 4.

3.4 Selection

E-Motion uses tournament selection, a popular and effective selection method. A percentage *t* of the current population of individuals is selected randomly, and they "battle"' against each other, i.e., the fittest individual is chosen to undergo crossover or mutation. E-Motion also implements the elitism technique, which

means it preserves a percentage *x* of the fittest individuals of the current population for the next one.

3.5 Crossover

E-Motion implements the crossover operation as follows. First, two individuals randomly chosen among the selected ones (selection operation) will exchange sub-trees. According to a randomly selected number which varies from 1 (root node) to *n* (total number of tree nodes), E-Motion 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. In case 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 according to the randomly selected number in both parents, E-Motion will exchange the whole sub-trees which are represented by the selected nodes, generating two new individuals (offspring).

Figure 1 illustrates the crossover operation, where the offspring is created by keeping the structure of the parents but with the selected nodes being replaced. By exchanging the whole sub-trees from the selected nodes and not only specific nodes, we avoid domain irregularities, because each edge refers to attribute characteristics that are represented by a node. It does not prevent, however, redundant rules and inconsistencies. See Section 3.7 for details on how E-Motion addresses these issues.

3.6 Mutation

E-Motion 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, acting like a pruning procedure. The second strategy replaces a randomly selected leaf node in an individual by a basic tree generated during the initial forest creation. These strategies aim at increasing or diminishing the individual's size, improving the population heterogeneity and avoiding local optima convergence.

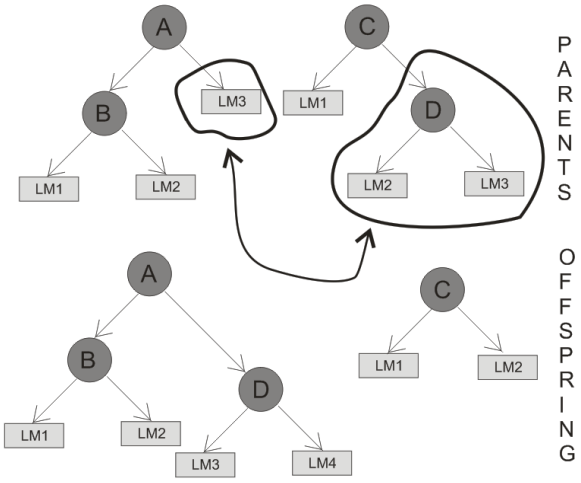

Figure 1. Crossover between two individuals.

3.7 Validity Issues

The stochastic operators of the evolutionary algorithm, due to their own nature, may create incoherent scenarios regarding the logical structure of model trees. E-Motion implements a filtering process to deal with these validity issues. After each evolutionary iteration and before each individual is evaluated, the training dataset is distributed along each individual so the linear models are calculated according to the instances that reach the leaf nodes. The following inconsistent scenarios may occur:

- *A categorical attribute appears more than once in a sub-tree*. It is well-known that the same categorical attribute test should not be done more than once in a same sub-tree.

- *Incoherent thresholds for repeated numeric nodes*. If a numeric attribute appears in a same sub-tree more than once, a special caution should be taken regarding the threshold values. For instance, if a given numeric attribute x is tested according to a threshold 10, the sub-tree that results from the test $x < 10$ may possibly contain another test over the attribute x, but no threshold larger than 10 would result in a node that holds instances. Incoherent thresholds also should be prevented from happening.

To deal with these two inconsistent scenarios, E-Motion analyzes whether the parent node of a given empty (with no instances) leaf node is categorical or numeric. If it is the case the parent is numeric, it means that no training instance fits that given numeric threshold and hence there is no need to make the binary split to the parent node. Thus, the parent node that was once a numeric node is transformed into a leaf node, and the child sub-trees are pruned. In the case where the parent node is categorical, each one of its children is verified, and the same procedure is applied for the case that only one child node holds instances. This process provides smaller trees that are consistent with the training dataset, preventing empty leaf nodes and consequently incoherent numeric thresholds and repeated categorical attributes in a same sub-tree. Figure 2 presents both cases, with the original and filtered trees.

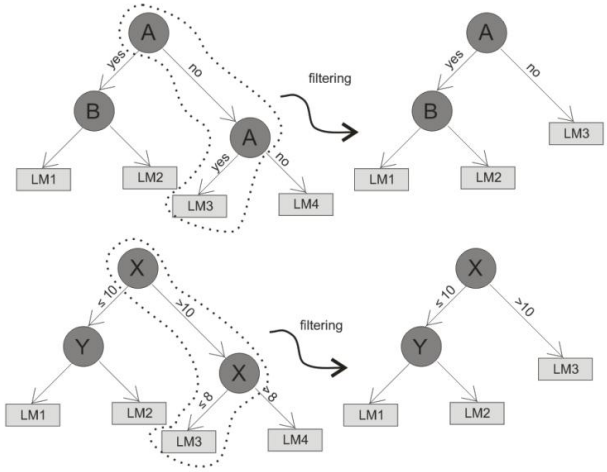

Figure 2. Filtering process.

4. EXPERIMENTAL METHODOLOGY

We have applied the proposed approach of inducing model trees through a lexicographic multi-objective GA to several regression datasets available in the UCI machine learning repository [1], namely AutoMpg {A}, BreastTumor {B}, FishCatch {F}, MachineCPU {M}, Quake {Q}, Stock{S_o}, Strike {S_i} and Veteran {V} (Table 1). The machine used in our experiments is a dual Intel Xeon Quad-core E5310 running at 1.6 GHz each, 8 MB L2 and 4 GB RAM. First, we have analyzed the *RMSE*, *MAE* and *tree size* obtained by the M5P algorithm (WEKA's

implementation of the well-known M5) [24] and REPTree [25], a regression tree algorithm also implemented in WEKA, for the datasets listed in Table 1.

Table 1. Public datasets used for experimentation.

Dataset	#Instances	Numeric Attributes	Categorical Attributes
AutoMPG	398	4	3
BreastTumor	286	1	8
MachineCPU	209	6	0
Quake	2178	3	0
Stock	950	9	0
Strike	625	5	1
Veteran	137	3	4

The M5P and REPTree parameter settings used are the default ones and we have used 10-fold cross-validation, a widely disseminated approach for validating prediction models. In each of the ten iterations of the cross-validation procedure, the training set is divided into sub-training and validation sets, which are used to produce the basic trees and linear models (sub-training set), filtering process (sub-training set) and fitness function (validation set). Each set represents 50% of the full training set. This split is intended to avoid training data overfitting.

E-Motion was executed according to the parameters listed in Table 2. The parameter values were based on our previous experience in using evolutionary algorithms for decision tree induction, and we have made no attempt to optimise parameter values, a topic left for future research. For the lexicographic analysis, *RMSE* is the highest-priority measure, followed by *MAE* and tree size, respectively. Thresholds were calculated dynamically, where each error measure has a threshold of 5% of its average value within the current population, and 20% for the average tree size. These parameters were defined empirically, through previous experimentation. "*Convergence rate*" (i.e., rate of the maximum number of generations without fitness improvement) and "*max number of generations*" are the algorithm's stopping criteria.

Table 2. E-Motion parameters.

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

Due to the fact that GA is a non-deterministic technique, we have run E-Motion 30 times (varying the random seed across the runs) for each one of the ten training/test set folds generated by the 10-fold cross-validation procedure. These folds were the same ones used by M5P and REPTree, to make the comparison among the algorithms as fair as possible. After running E-Motion on each of the eight 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. We have calculated the averages and standard deviations for the ten folds of M5P and REPTree since these are deterministic algorithms.

To assess the statistical significance of the differences observed in the experiments for each dataset, we have executed the corrected paired t-test [17], with a significance level of $\alpha = 0.05$ and 9 degrees of freedom. The measures we analysed were the same we used in the lexicographic analysis: tree size, *RMSE* and *MAE*. To assess the statistical significance of the differences observed in the experiments for each dataset, we have executed the corrected paired t-test [17], with a significance level of $\alpha = 0.05$ and 9 degrees of freedom. The measures we analysed were the same we used in the lexicographic analysis: tree size, *RMSE* and *MAE*.

5. EXPERIMENTAL RESULTS

Table 3 presents the average values for the error measures for E-Motion (E), M5P and REPTree (REP). Standard deviation values are within parentheses.

Table 3. Error measures for E-Motion, M5P and REPTree.

Dataset	RMSE			MAE		
	E	M5P	REP	E	M5P	REP
AutoMPG	2.98 (0.65)	2.72 (0.52)	3.44 (0.59)	2.20 (0.38)	2.00 (0.33)	2.53 (0.36)
BreastTumor	10.45 (1.24)	9.94 (1.36)	10.47 (1.02)	8.36 (1.15)	8.05 (1.08)	8.30 (0.92)
FishCatch	62.72 (0.98)	59.92 (15.96)	139.30 (60.56)	43.41 (0.81)	40.76 (10.45)	81.72 (29.36)
MachineCPU	45.91 (21.82)	54.81 (27.38)	93.13 (58.08)	29.51 (11.13)	29.82 (10.27)	49.73 (24.28)
Quake	0.19 (0.01)	0.19 (0.01)	0.19 (0.01)	0.15 (0.01)	0.15 (0.01)	0.15 (0.01)
Stock	1.04 (0.07)	0.92 (0.20)	1.21 (0.24)	0.81 (0.05)	0.67 (0.08)	0.84 (0.12)
Strike	440.44 (282.66)	436.99 (277.65)	459.87 (274.23)	217.52 (52.36)	211.29 (48.91)	225.40 (54.68)
Veteran	133.35 (86.34)	126.85 (78.12)	140.01 (81.91)	92.68 (47.97)	91.95 (42.88)	98.98 (37.39)

Table 4 presents the results regarding the average tree size (number of nodes) of the solutions produced by each algorithm. Table 5 shows in which datasets the differences regarding the error measures and tree size were statistically significant, according to the corrected paired t-test. This table is divided into two parts. The left part indicates the datasets in which E-Motion was significantly better than M5P or REP according to each of the three criteria in column (a). Each entry in the column E-Motion contains 8 values, one for each of the datasets, and the value in question is a dataset identifier if E-Motion was significantly *better* than the corresponding algorithm in the second column (M5P or REP) according to the corresponding measure in column (a); otherwise the value in question is "-". The second part of the table has a similar structure, but now each entry in the column E-Motion indicates for which dataset E-Motion was significantly *worse* than the corresponding algorithm in the second column, according to the corresponding measure in column (b).

Table 4. Tree Size for E-Motion, M5P and REPTree.

Dataset	Tree Size		
	E	M5P	REP
AutoMPG	3.17 (0.26)	6.60 (2.63)	71.90 (14.08)
BreastTumor	2.59 (0.80)	1.80 (1.03)	10.30 (12.60)
FishCatch	3.34 (0.60)	7.80 (4.34)	38.60 (10.72)
MachineCPU	6.18 (0.83)	6.00 (3.16)	21.20 (14.28)
Quake	1.00 (0.00)	3.60 (2.99)	18.00 (33.88)
Stock	11.78 (0.67)	87.80 (15.70)	160.40 (15.69)
Strike	4.93 (0.72)	10.40 (7.55)	41.20 (16.10)
Veteran	3.31 (0.75)	1.80 (2.53)	9.80 (8.44)

Regarding statistical significance results, we can notice that E-Motion performs similarly to M5P and REPTree in terms of error measures. It outperforms M5P in the MachineCPU dataset (*RMSE* and *MAE*), and it outperforms REPTree in FishCatch (*RMSE* and

MAE) and MachineCPU (MAE). E-motion is never outperformed by REPTree considering the error measures, and it is outperformed by M5P only in Stock (*MAE*). Considering absolute values only, E-Motion is superior to REPTree in 7 datasets for *RMSE* and 6 for *MAE*. However, it is outperformed by M5P, though by statistically insignificant margins.

Table 5. E-Motion significantly better (a) or worse (b) than M5P and REP according to the corrected paired t-test.

(a)		E-Motion
RMSE	M5P	---M----
	REP	--F-----
MAE	M5P	---M----
	REP	--FM----
Tree Size	M5P	--F-QS_o--
	REP	A-FM-S_oS_i-

(b)		E-Motion
RMSE	M5P	--------
	REP	--------
MAE	M5P	-----S_o--
	REP	--------
Tree Size	M5P	---M----
	REP	--------

Once E-Motion induces trees with predictive performance similar to M5P and REPTree, it is interesting to analyse the comprehensibility of the models generated. As we can see in Table 4, E-Motion always produces smaller trees when compared to REPTree. In addition, it produced trees that are smaller than M5P ones in 5 out of the 8 datasets we used. This difference is statistically significant in 5 out of 8 datasets when comparing E-Motion to REPTree, and in 3 out of 8 when comparing E-Motion to M5P. Only in the MachineCPU dataset M5P was able to generate trees that are smaller than E-Motion's trees with statistical significance. Overall, these results suggest that E-Motion seems to produce trees which are often both accurate and significantly smaller than the other two traditional algorithms.

We conclude by arguing that the lexicographic analysis used as fitness function has achieved its goal by providing a good trade-off between predictive performance and model comprehensibility, which we consider to be the main contribution of this work. We point out that E-Motion's execution time is coherent to most evolutionary algorithms, which means it turns out to be around 10 times slower than most greedy algorithms for generating model trees. Note that in predictive data mining tasks such as regression, computational time is normally considered much less important than solution-quality criteria such as prediction error and tree size.

6. RELATED WORK

Evolutionary induction of decision trees is a well-addressed issue, as presented in works such as [2], [3], [14], [26]. However, very few studies have discussed the evolutionary induction of regression/model trees, and we briefly review them below.

TARGET [9], [16] is a GA proposed to evolve regression trees. It makes use of a Bayesian information criterion as the measure of tree fitness, which is basically a weighted-formula that penalizes for model complexity. TARGET is compared to the traditional regression tree induction algorithm CART [7], and also to the ensemble methods Random Forests and Bagging in two public regression datasets. It outperforms CART in terms of MSE, but is outperformed by both ensemble methods. The authors claim that TARGET presents a good trade-off in terms of error measures and model comprehensibility (final result is a single tree), whereas CART presents only good comprehensibility and the ensemble methods only good predictive performance.

GPMCC [18] is a framework proposed to evolve model trees. Its structure is divided in three different parts: (1) GASOPE [19], which is a GA to evolve polynomial expressions; (2) K-Means, which is a traditional clustering algorithm and is used in this framework to sample the training set; and (3) a genetic algorithm to evolve the structure of trees through a fitness function that is an extended form of the adjusted coefficient of determination (a weighted-formula approach that penalizes for the size of the tree and the complexity of each terminal model). GPMCC handles trees that contain non-linear models in their terminal nodes, a variation of traditional model trees. It is compared to a neural network approach called NeuroLinear [22] and to a commercial version of Quinlan's M5 [20], outperforming these approaches only by the number of rules generated (paths of the tree), and being outperformed in terms of predictive performance (mean-absolute error).

Our approach for evolving trees (E-Motion) differs from TARGET since we are evolving model trees and not regression trees. It is important to notice that model trees are a more robust and effective approach when compared to regression trees, often presenting more accurate results [25]. Our strategy, while having the same goal as the GPMCC framework, makes use of a single GA to evolve model trees, which makes the induction of model trees faster and simpler. Additionally, GPMCC relies on a single-objective evaluation, which can explain the fairly inaccurate results it provides. Our approach, on the other hand, makes use of a lexicographic analysis to better handle the multi-objective optimisation and find a good trade-off between predictive performance and comprehensibility. To the best of our knowledge, E-Motion is the first lexicographic multi-objective evolutionary algorithm for model tree induction.

7. CONCLUSIONS AND FUTURE WORK

Model trees are a popular alternative to classical regression methods, mainly because the models they provide resemble the human reasoning. We emphasize that the comprehensibility of the discovered model is important in many applications where decisions will be made by human beings based on the discovered knowledge. Therefore, there is a clear motivation to provide model trees that are not only accurate but also relatively simple.

Traditional model tree induction algorithms which rely on a recursive top-down greedy strategy are relatively fast but susceptible to converging to local optima, while an ideal algorithm should choose the correct tree partitioning in order to converge to a global optimum. With this goal in mind, we have proposed a novel Genetic Algorithm for inducing model trees called E-Motion. It avoids the greedy search performed by conventional tree induction algorithms, and performs instead a global search in the space of candidate model trees. Additionally, while other approaches typically rely on a single objective evaluation (possibly collapsing multiple objectives into a single objective using a weighted formula), 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.

E-Motion considers the two most common error measures used for evaluating regression problems: root mean-square error and mean-absolute error. Also, it considers tree size as a measure of model comprehensibility, assuming that smaller trees are easier to interpret. In experiments with 8 datasets, E-Motion's results did not significantly differ from the popular M5 algorithm regarding the error measures, but E-Motion consistently induced smaller

model trees. This means an overall improvement in simplicity was obtained without any statistically significant loss in predictive performance in most datasets. This is a clearly positive result, which is also supported by the well-known principle of Occam's razor, a principle very often used in data mining and science in general. Regarding the comparison with REPTree, E-Motion is superior both in error measures and tree size, which is partially explained by the different types of trees that are induced (model trees versus regression trees), but also by the capacity of E-Motion of producing reduced and accurate trees.

Some possibilities for future research are as follows. First, the setting of input parameters for E-Motion could be done through a supportive GA, helping to achieve convergence to a global optimum. In addition, we intend to extend the leaf models in order to generate polynomial expressions, so each model tree will hold non-linear models in its leaves. Finally, we are looking for parallel solutions in order to speed-up the algorithm's execution time.

8. ACKNOWLEDGMENTS

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

REFERENCES

1 Asuncion, A. and Newman, D. *UCI Machine Learning Repository*. 2007.

2 Basgalupp, M., Barros, Rodrigo C., Carvalho, A. P. L. F. de, Freitas, Alex A., and Ruiz, Duncan D. LEGAL-Tree: A lexicographic multi-objective genetic algorithm for decision tree induction. In *Proceedings of the 24th Annual ACM Symposium on Applied Computing* (Honolulu, Hawaii, USA 2009), 1085-1090.

3 Basgalupp, M., Barros, Rodrigo C., Carvalho, A. P. L. F. de, Freitas, Alex A., and Ruiz, Duncan D. Lexicographic multi-objective evolutionary induction of decision trees. *International Journal of Bio-Inspired Computation*, 1 (2009), 105-117.

4 Bjorck, A. *Numerical Methods for Least Square Problems*. SIAM, 1996.

5 Breiman, L. Bagging Predictors. *Machine Learning*, 24, 2 (1996), 123-140.

6 Breiman, L. Random Forests. *Machine Learning*, 45, 1 (2001), 5-32.

7 Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Chapman & Hall, 1984.

8 Burgess, C. J. and Lefley, M. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, 43, 14 (2001), 863-873.

9 Fan, G. and Gray, J. B. Regression tree analysis using target. *Journal of Computational and Graphical Statistics*, 14, 1 (2005), 206-218.

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

11 Freitas, Alex A. A Review of Evolutionary Algorithms for Data Mining. In *Soft Computing for Knowledge Discovery and Data Mining*. Springer US, 2008, 79-111.

12 Freitas, Alex A, Wieser, D. C., and Apweiler, R. On the importance of comprehensible classification models for protein function prediction. *To appear in IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

13 Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 1 (1997), 119-139.

14 Fu, Z., Golden, B. L., Lele, S., Raghavan, S., and Wasil, E. Diversification for better classification trees. *Computers & Operations Research*, 33, 11 (2006), 3185-3202.

15 Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Boston, 1989.

16 Gray, J. B. and Fan, G. Classification tree analysis using TARGET. *Computational Statistics & Data Analysis*, 52 (2008), 1362-1372.

17 Nadeau, C. and Bengio, Y. Inference for the generalization error. *Machine Learning*, 52, 3 (2003), 239-281.

18 Potgieter, G. and Engelbrecht, A. Evolving model trees for mining data sets with continuous-valued classes. *Expert Systems with Applications*, 35 (2008), 1513-1532.

19 Potgieter, G. and Engelbrecht, A. Genetic algorithms for the structural optimisation of learned polynomial expressions. *Applied Mathematics and Computation*, 186, 2 (2007), 1441-1466.

20 Quinlan, J. R. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence* (1992), 343-348.

21 Rokach, L. and Maimon, O. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing, 2008.

22 Setiono, R., Leow, W. K., and Zurada, J. M. Extraction of rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks*, 13, 3 (2002), 564-577.

23 Tan, P. N., Steinbach, M., and Kumar, V. *Introduction to Data Mining.* Pearson Education, 2006.

24 Wang, Y. and Witten, I. Inducing model trees for continuous classes. In *Poster papers of the 9th European Conference on Machine Learning* (1997).

25 Witten, I. H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

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