



# The role of decision tree representation in regression problems – An evolutionary perspective



Marcin Czajkowski\*, Marek Kretowski

Faculty of Computer Science, Białystok University of Technology, Wiejska 45a, 15-351 Białystok, Poland

## ARTICLE INFO

### Article history:

Received 26 September 2015

Received in revised form 20 June 2016

Accepted 2 July 2016

Available online 16 July 2016

### Keywords:

Evolutionary algorithms

Data mining

Regression trees

Self-adaptable representation

## ABSTRACT

A regression tree is a type of decision tree that can be applied to solve regression problems. One of its characteristics is that it may have at least four different node representations; internal nodes can be associated with univariate or oblique tests, whereas the leaves can be linked with simple constant predictions or multivariate regression models. The objective of this paper is to demonstrate the impact of particular representations on the induced decision trees. As it is difficult if not impossible to choose the best representation for a particular problem in advance, the issue is investigated using a new evolutionary algorithm for the decision tree induction with a structure that can self-adapt to the currently analyzed data. The proposed solution allows different leaves and internal nodes representation within a single tree. Experiments performed using artificial and real-life datasets show the importance of tree representation in terms of error minimization and tree size. In addition, the presented solution managed to outperform popular tree inducers with defined homogeneous representations.

© 2016 Elsevier B.V. All rights reserved.

## 1. Introduction

Data mining [18] can reveal important and insightful information hidden in data. However, appropriate tools and algorithms are required to effectively identify correlations and patterns within the data. Decision trees [24,40] represent one of the main techniques for discriminant analysis prediction in knowledge discovery. The success of tree-based approaches can be explained by their ease of application, fast operation, and effectiveness. Furthermore, the hierarchical tree structure, in which appropriate tests from consecutive nodes are sequentially applied, closely resembles a human way of decision making. All this makes decision trees easy to understand, even for inexperienced analysts. Despite 50 years of research on decision trees, many problems still remain [30], such as searching only for a locally optimal split in the internal nodes; appropriate pruning criterion, efficient analysis of cost-sensitive data or performing multi-objective optimization. To help resolve some of these problems, evolutionary computation (EC) has been applied to decision tree induction [2]. The strength of this approach lies in the global search for splits and predictions. It results in higher accuracy and smaller output trees compared to popular greedy decision tree inducers.

Finding appropriate representation of the predictor before actual learning is a difficult task for many data mining algorithms. Often, the algorithm structure must be pre-defined and fixed during its life-cycle, which is a major barrier in developing *intelligent* artificial systems. This problem is well known [20] in artificial neural networks where the topology and the number of neurons is unknown, in support vector machines with their different types of kernels, and in decision trees where there is a need to select the type of node representation. One solution is to automatically adapt the structure of the algorithm to the analyzed problem during the learning phase, which can be accomplished using the evolutionary approach [27,33]. This approach is also applied to classification trees [29,26] where a mixed test representation in the internal nodes is possible.

In this paper, we want to investigate the role of regression tree representation and its impact on predictive accuracy and induced tree size as it has not been sufficiently explored. Using artificially generated datasets, we will reveal the pros and cons of trees with different representation types, focusing mainly on evolutionary induced trees for regression problems [2]. Differences in the representation of regression trees [30] can occur in two places: in the tests in the internal nodes and in the predictions in the leaves. For real-life problems, it is difficult to say which kind of decision tree (univariate, oblique, regression, model) should be used. It is often almost impossible to choose the best representation in advance. To top it all, for many problems heterogeneous node representation

\* Corresponding author.

E-mail address: [m.czajkowski@pb.edu.pl](mailto:m.czajkowski@pb.edu.pl) (M. Czajkowski).

is required within the same tree. This is why we also study a specialized evolutionary algorithm (EA) called the Mixed Global Model Tree (mGMT). It induces a decision tree that we believe self-adapts its structure to the currently analyzed data. The output tree may have different internal node and leaf representations, and for a given dataset it may be as good or even better than any tree with strict representation.

The paper is organized as follows. The next section provides a brief background on regression trees. Section 3 describes the proposed extension for evolutionary inducers with homogeneous representations. All experiments are presented in Section 4, and the last section comprises the conclusion and suggestions for future work.

## 2. Decision trees

We may find different variants of decision trees in the literature [30]. They can be grouped according to the type of problem they are applied to, the way they are induced, or the type of structure. In classification trees, a class label is assigned to each leaf. Usually, it is the majority class of all training instances that reaches that particular leaf. In this paper, we focus on regression trees that may be considered variants of decision trees designed to approximate real-valued functions instead of being used for classification tasks. Although regression trees are not as popular as classification trees, they are highly competitive with different machine learning algorithms [35] and are often applied to many real-life problems [16,28].

In the case of the simplest regression tree, each leaf contains a constant value, usually an average value of the target attribute. A model tree can be seen as an extension of the typical regression tree [46,31]. The constant value in each leaf of the regression tree is replaced in the model tree by a linear (or nonlinear) regression function. To predict the target value, the new tested instance is followed down the tree from a root node to a leaf using its attribute values to make routing decisions at each internal node. Next, the predicted value for the new instance is evaluated based on a regression model in the leaf. Examples of predicted values of classification, regression, and model trees are given in Fig. 1. The gray level color of each region represents a different class label (for a classification tree), and the height corresponds to the value of the prediction function (regression and model trees).

Most decision trees partition the feature space with axis-parallel decision borders [44]. This type of tree is called univariate because each split in the non-terminal node involves a single feature. For continuous-valued features, inequality tests with binary outcomes are usually applied, and for nominal features mutually exclusive groups of feature values are associated with the outcomes. When more than one feature is taken into account to build a test in an

internal node, we deal with multivariate decision trees [8]. The most common form of such a test is an oblique split, which is based on a linear combination of features. The decision tree that applies only oblique tests is often called oblique or linear, whereas heterogeneous trees with univariate, linear, and other multivariate (e.g., instance-based) tests are called mixed trees [29]. Fig. 2 shows an example of univariate and oblique decision trees. We can observe that if decision borders are not axis-parallel, then using only univariate tests may lead to an overcomplicated classifier. This kind of situation is known as a ‘staircase effect’ [8] and can be avoided by applying more sophisticated multivariate tests. While oblique trees are generally smaller, the tests are usually more difficult to interpret. It should be emphasized that the computational complexity of multivariate tree induction is significantly higher than that of univariate tree induction [3].

The role of tree representation has so far been discussed mainly in terms of classification problems. The study [25,8] shows that univariate inducers return larger trees than multivariate ones, and they are often less accurate. However, multivariate trees are difficult to understand and interpret, and the tree induction is significantly slower. Therefore, making a general conclusion is risky as the most important factors are the characteristics of the particular dataset [25]. To the best of our knowledge, there is no detailed report that refers to the role of representation in regression trees. It could be expected that univariate and multivariate regression trees should behave similarly to the classification ones. However, there is still an open question about the influence of the leaves’ representation on the tree performance. The paper focuses on evolutionary induced regression trees; therefore, to go further, we must briefly describe the process of creating a decision tree from the training set. The two most popular concepts for the decision tree induction are the top-down and global approaches. The first is based on a greedy procedure known as recursive partitioning [39]. In the top-down approach, the induction algorithm starts from the root node where the locally optimal split is searched according to the given optimality measure. Next, the training instances are redirected to the newly created nodes, and this process is repeated for each node until a stopping condition is met. Additionally, post-pruning [15] is usually applied after the induction to avoid the problem of over-fitting the training data.

One of the most popular representatives of top-down induced univariate regression trees is a solution proposed by Breiman et al. called Classification And Regression Tree (CART) [7]. The algorithm searches for a locally optimal split that minimizes the sum of squared residuals and builds a piecewise constant model with each terminal node fitted with the training sample mean. Other solutions have managed to improve the prediction accuracy by replacing single values in the leaves with more advanced models. The M5 system [46] induces a tree that contains multiple linear models in

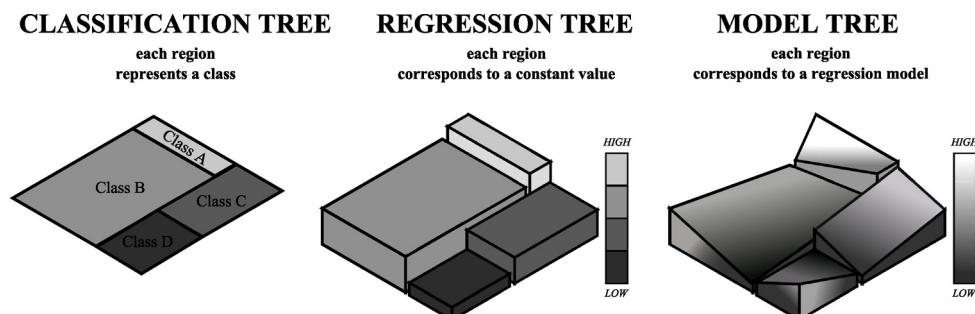


Fig. 1. An illustration of predicted values of the classification, regression, and model trees.

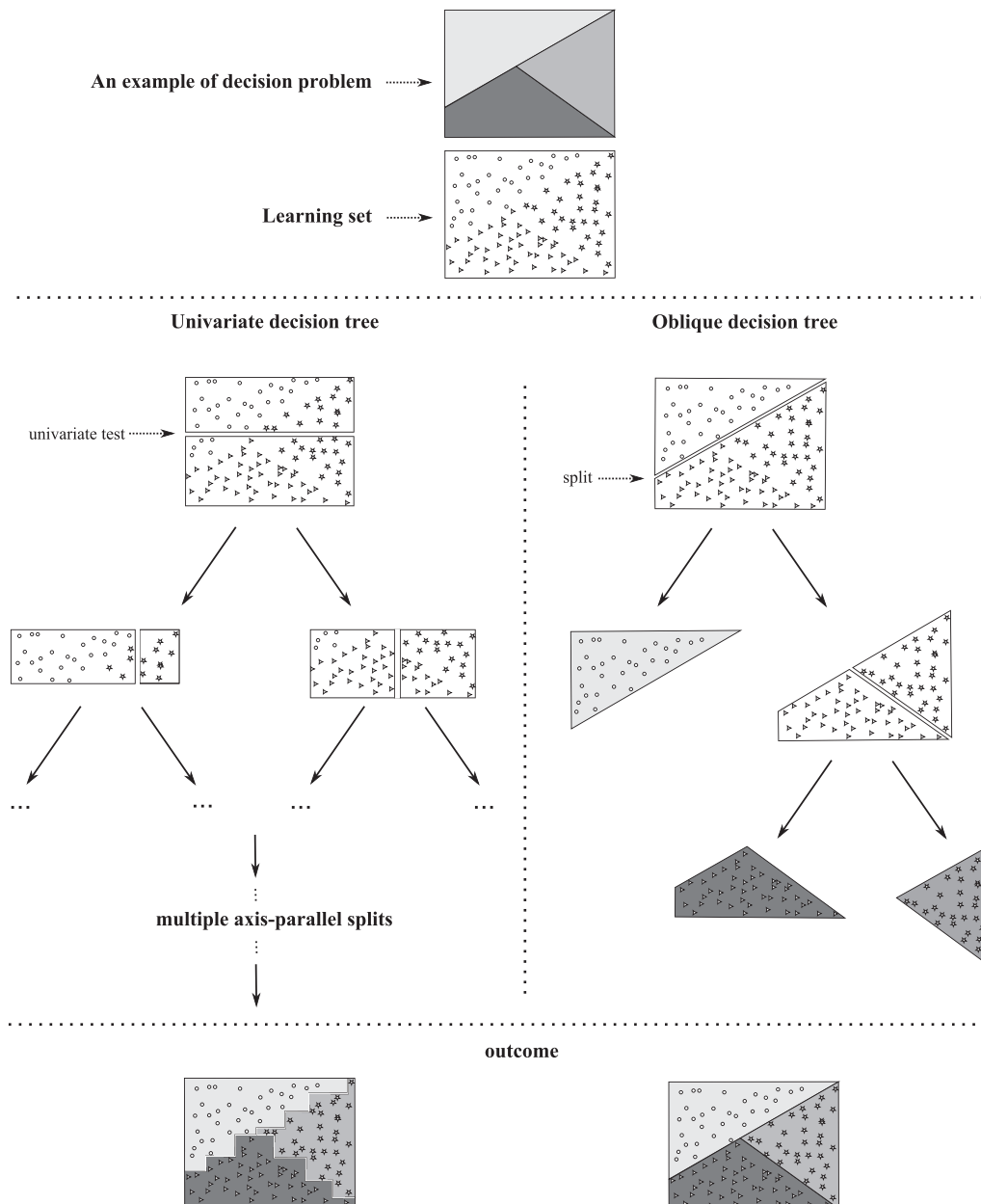


Fig. 2. An example of oblique and univariate decision trees.

the leaves. A solution called Stepwise Model Tree Induction (SMOTI) [31] can be viewed as an oblique model tree as the regression models are placed not only in the leaves but also in the upper parts of the tree. All aforementioned methods induce trees with the greedy strategy, which is fast and generally efficient but often produces only locally optimal solutions.

The global approach for the decision tree induction limits the negative effects of locally optimal decisions. It tries to simultaneously search for the tree structure, tests in the internal nodes, and models in the leaves. This process is obviously much more computationally complex but can reveal hidden regularities that are often undetectable by greedy methods. The global induction is mainly represented by systems based on an evolutionary approach [2,4]; however, there are solutions that apply, for example, ant colony optimization [36,6].

In the literature, there are relatively fewer evolutionary approaches for the regression and model trees than for the classification trees. Popular representatives of EA-based univariate regression trees are the TARGET solution [17] that evolve a CART-like regression tree with basic genetic operators and the uGRT algorithm [11] that introduces specialized variants of mutation and crossover. A strongly typed GP (Genetic Programming) approach called STGP was also proposed [21] for univariate regression tree induction. There are also globally induced systems that evolve univariate model trees, such as the E-Motion tree [1] that implements standard 1-point crossover and two different mutation strategies and the GMT system [12] that incorporates knowledge about the inducing problem for the global model tree into the evolutionary search. There are also preliminary studies on oblique trees called oGMT [10]. In the literature, we may also find the GP

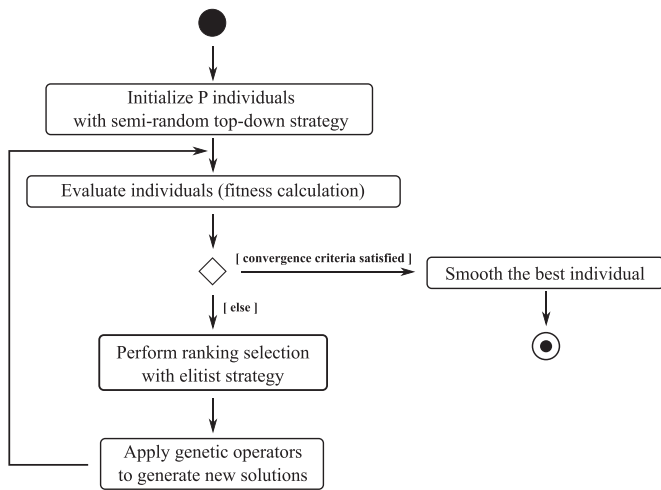


Fig. 3. The mGMT process diagram.

approach that evolves the model trees with nonlinear regression models in the leaves called GPMCC [38]. It is composed from the GP to evolve the structure of the model trees and GA to evolve polynomial expressions (GASOPE) [37].

### 3. Mixed Global Model Tree

This paper focuses on the representation of globally induced regression and model trees and its influence on the output tree. In this section, we propose an extension for the GMT and GRT systems [12] called the Mixed Global Model Tree (mGMT) to better understand the underlying process behind the selection of the representation. With the evolutionary tree induction, we are able not only to search for an optimal tree structure, tests in internal nodes, or models in the leaves but also to self-adapt the tree representation. The general structure of the algorithm follows a typical EA framework [32] with an unstructured population and a generational selection. It can be treated as a unified framework for both univariate and oblique tests in the internal nodes and regression and models leaves. The mGMT does not require to set the tree representation in advance because the EA validates different variants of the representations not only on the tree level but also on the node level and may induce a heterogeneous tree that we called a mixed tree. A description of the proposed approach is given, especially with respect to issues that are specific to mixed trees.

The process diagram of the mGMT algorithm is illustrated in Fig. 3. The proposed solution evolves the regression and model trees in their actual forms. The candidate solutions that constitute the population are initialized with the semi-random greedy strategy and are evaluated using the multi-objective weight formula fitness function. If the convergence criteria is not satisfied, a linear ranking selection is performed together with the elitist strategy. Next, genetic operators are applied, including different variants of specialized mutations and crossovers. After the evolution process is finished, the best individual found using the EA is smoothed. Each element of the mGMT solution is discussed in detail in the following sections.

#### 3.1. Representation

A mixed regression tree is a complex structure in which the number and the type of nodes and even the number of test outcomes are not known in advance for a given learning set.

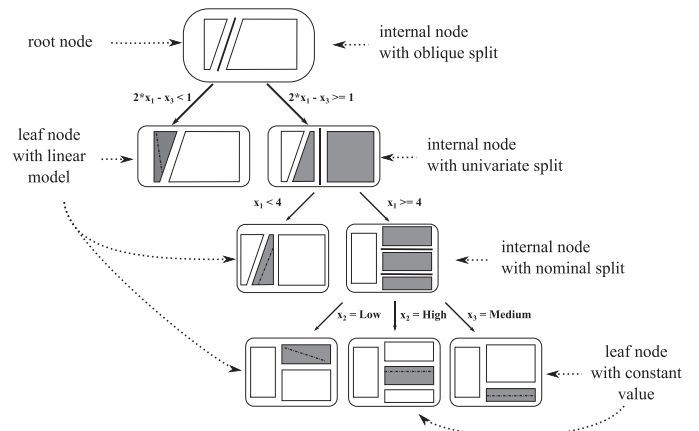


Fig. 4. An example representation of the mGMT individual.

Therefore, the candidate solutions that constitute the population are not encoded and are represented in their actual form (see Fig. 4).

There are three possible test types in the internal nodes: two univariate and one multivariate. In the case of univariate tests, a test representation concerns only one attribute and depends on the considered attribute type. For continuous-valued features, typical inequality tests with two outcomes are used. For nominal attributes, at least one attribute value is associated with each branch starting in the node, which means that an internal disjunction is implemented. Only binary or continuous-valued attributes are used to construct the oblique split. The feature space can be divided into two regions by a hyperplane:

$$H(\mathbf{w}, \theta) = \{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle = \theta\}, \tag{1}$$

where  $\mathbf{x}$  is a vector of feature values (objects),  $\mathbf{w} = [w_1, \dots, w_P]$  is a weight vector,  $\theta$  is a threshold,  $\langle \mathbf{w}, \mathbf{x} \rangle$  represents an inner product, and  $P$  is the number of independent variables. Each hyperplane is represented by a fixed-size  $P + 1$  – dimensional table of real numbers corresponding to the weight vector  $\mathbf{w}$  and the threshold  $\theta$ .

In each leaf of the mGMT system, a multiple linear model can be constructed using the standard regression technique. It is calculated only for objects associated with that node. A dependent variable  $y$  is explained by the linear combination of multiple independent variables  $x_1, x_2, \dots, x_P$ :

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_P * x_P, \tag{2}$$

where  $\beta_0, \dots, \beta_P$  are fixed coefficients that minimize the sum of the squared residuals of the model. If all  $\beta_i$  ( $0 < i \leq P$ ) are equal to 0, the leaf node will be a regression node with a constant equal to  $\beta_0$ . If only one  $\beta_i \neq 0$  then, we deal with simple linear regression; otherwise each leaf contains simple or multivariate linear regression.

#### 3.2. Initialization

Each initial individual in the population is created with the classical top-down approach that resembles the M5 solution [46]. The initial population of mGMT is heterogeneous and is composed of five types of standard regression trees with different representations (four homogeneous and one heterogeneous): a univariate regression tree; an oblique regression tree; a univariate model tree; an oblique model tree; and a mixed tree that contains different kinds of tests in the internal nodes (univariate and oblique) and different types of leaves (regression and model). In mixed trees, before each step of recursive partitioning, the type of node is selected

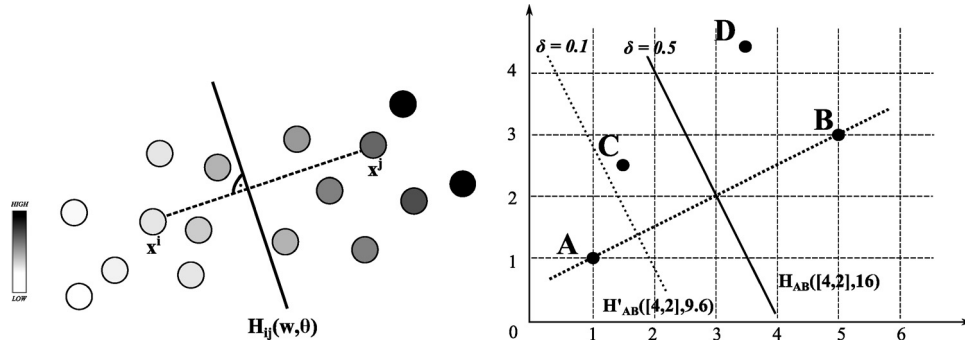


Fig. 5. Hyperplane initialization based on randomly chosen 'long dipole' (left) and an example illustrating how the oblique test is created (right).

randomly and an appropriate test or model is generated. The importance of such a heterogeneous initial population is its diversity. The recursive partitioning is finished when the dependent value is predicted for all training objects in the node or the number of instances in the node is small (default: five instances). Each initial individual is created based on a semi-random subsample of the original training data (default: 10% of data) to keep the balance between exploration and exploitation. To ensure that the subsample contains objects with various values of the predicted attribute, the training data is sorted by the predicted value and split into a fixed number of equal-size folds (default: 10). From these folds, an equal number of objects is randomly chosen and placed into the subsample. Tests in non-terminal nodes are calculated from a random subset of attributes (default: 50%).

In the case of the univariate internal nodes, one of three memetic search strategies [12] that involves employing the locally optimized tests is chosen:

- *Least Squares (LS)*: the test in the internal node is chosen according to the node impurity measured by the sum of the squared residuals.
- *Least Absolute Deviation (LAD)*: the test reduces the sum of the absolute deviations. It is more robust and has greater resistance to outlying values than *LS*.
- *Dipolar*: the test is constructed according to the 'long dipole' [12] strategy. At first, an instance that will constitute the dipole is randomly selected from the set of instances from the current node. The rest of the feature vectors are sorted in decreasing order according to the difference between the dependent variable values and the selected instance. The second instance that constitutes the dipole should have a much different value than the dependent variable. To find it, we applied a mechanism similar to the ranking linear selection [32]. Finally, the test that splits the dipole is constructed based on a randomly selected attribute where the boundary threshold is defined as a midpoint between the pairs that constitute the dipole.

The search strategy used to find splits in the internal nodes is different for the oblique tests. An effective test in a non-terminal node is searched only using the dipolar strategy. Fig. 5 (left) illustrates the hyperplane initialization based on a randomly chosen 'long dipole'. The hyperplane  $H_{ij}(\mathbf{w}, \theta)$  splits the dipole ( $\mathbf{x}^i, \mathbf{x}^j$ ) in such a way that the two feature vectors  $\mathbf{x}^i$  and  $\mathbf{x}^j$  are situated on the opposite sides of the dividing hyperplane:

$$(\langle \mathbf{w}, \mathbf{x}^i \rangle - \theta) * (\langle \mathbf{w}, \mathbf{x}^j \rangle - \theta) < 0. \quad (3)$$

The hyperplane parameters are as follows:  $\mathbf{w} = \mathbf{x}^i - \mathbf{x}^j$  and  $\theta = \delta * \langle \mathbf{w}, \mathbf{x}^i \rangle + (1 - \delta) * \langle \mathbf{w}, \mathbf{x}^j \rangle$ , where  $\delta \in (0, 1)$  is a randomly drawn coefficient that determines the distance between the opposite ends

of the dipole.  $H_{ij}(\mathbf{w}, \theta)$  is perpendicular to the segment connecting the dipole ends.

To provide a numeric example illustrating how an oblique test is created, let's imagine the two 2 dimensional space illustrated in Fig. 5 (right). After the selection of two randomly chosen dipoles with Cartesian coordinates equal to  $A(1, 1)$ ,  $B(5, 3)$ , and coefficient  $\delta = 0.5$ , the splitting hyperplane  $H$  parameters are:  $w[5 - 1, 3 - 1]$  and  $\theta = 0.5 * ((5 - 1) * (1 + 5)) + 0.5 * ((3 - 1) * (1 + 3)) = 16$ . Therefore, the hyperplane  $H_{AB}$  is a line described as:  $y = -2 * x + 8$ . To perform a split, we simply check on which side of the hyperplane  $H$  all instances from the internal node are positioned. Let's consider point  $C(1.5, 2.5)$ . By applying it to the hyperplane equation  $w(1.5 * 4 + 2.5 * 2)$ , we see that the score 11 is smaller than the value of  $\theta$ . Using a different point, for example,  $D(3.5, 4.5)$  would result in value 23, which means that the point  $D$  lies on the opposite side of the hyperplane to point  $C$ . For this particular example, the parameter  $\delta$  equals 0.5; therefore, the hyperplane  $w$  intersects the midpoint between dipoles  $A$  and  $B$ . However, if we change the parameter to  $\delta = 0.1$ , then the hyperplane denoted as  $H'_{AB}$  shifts towards point  $A$ . We can observe that for this hyperplane  $H'$  point  $C$  and point  $D$  lie on the same side and thus both instances would be directed after the split to the same sub-node.

### 3.3. Goodness of fit

The evolutionary search process is very sensitive to the proper definition of the fitness function. In the context of regression trees, a direct minimization of the prediction error measured in the learning set usually leads to the over-fitting problem. In typical top-down induction of decision trees [39], this problem is partially mitigated by defining a stopping condition and by applying post-pruning [15]. In the case of the evolutionary approach, the multi-objective function is required to minimize the prediction error and the tree complexity at the same time.

In our approach, a Bayesian information criterion (BIC) [41] is used as a fitness function. It was shown that this criterion worked well with regression and model trees [17,12] and outperforms other popular approaches. *BIC* is given by:

$$Fit_{BIC}(T) = -2 * \ln(L(T)) + \ln(n) * k(T), \quad (4)$$

where  $L(T)$  is the maximum of the likelihood function of the tree  $T$ ,  $n$  is the number of observations in the data, and  $k(T)$  is the number of model parameters in the tree. The log(likelihood) function  $L(T)$  is typical for regression models and can be expressed as:

$$\ln(L(T)) = -0.5n * [\ln(2\pi) + \ln(SS_e(T)/n) + 1], \quad (5)$$

where  $SS_e(T)$  is the sum of squared residuals of the tree  $T$ . The term  $k(T)$  can also be viewed as a penalty for over-parametrization.

The proposed mixed tree representation requires defining a new penalty for the tree over-parametrization. It is rather obvious that

in internal nodes an oblique split based on a few features is more complex than a univariate test. The same applies to the different leaf representations. As a consequence, the tree complexity  $k(T)$  should not only reflect the tree size but also the complexity of the tests in internal nodes and models in the leaves. However, it is not easy to arbitrarily set the importance of different measures because it often depends on the dataset being analyzed. In such a situation, the tree complexity  $k(T)$  is defined as:

$$k(T) = \alpha_1 * Q(T) + \alpha_2 * O(T) + \alpha_3 * W(T), \quad (6)$$

where  $Q(T)$  is the number of nodes in the model tree  $T$ ;  $O(T)$  is equal to the sum of the number of non-zero weights in the hyperplanes in the internal nodes, and  $W(T)$  is the sum of the number of attributes in the linear models in the leaves. Default values of the parameters are  $\alpha_1 = 2.0$ ,  $\alpha_2 = 1.0$ , and  $\alpha_3 = 1.0$ ; however, further research to determine their values is needed. If the  $i$ -th internal node  $T_i$  is univariate, the value of  $O(T_i)$  equals 1. If the  $j$ -th leaf contains a constant value, then the parameter  $W(T_j)$  equals zero because there are no attributes in the linear model. Otherwise, the value of  $O(T_i)$  and  $W(T_j)$  equals the number of attributes used to build the test in internal node  $i$  or the model in leaf  $j$ .

The flexibility of the fitness function allows its simple configuration based on additional knowledge or user preferences, for example, if users know the basic relationships in the data or want to limit tree representations to the desired ones, the fitness function can assign a high value to  $\alpha_2$  or  $\alpha_3$  or both.

### 3.4. Genetic operators

To maintain genetic diversity, the mGMT algorithm applies two specialized genetic operators corresponding to classical mutation and crossover. In globally induced trees with strict representations, there are several variants of the operators [11,12]; however, their availability mainly depends on the representation type. Both operators are applied with a given probability and influence the tree structure, the tests in non-terminal nodes, and optionally the models in the leaves. After any successful mutation or crossover, it is usually necessary to relocate learning vectors between the parts of the tree rooted in the altered node. This can cause pruning of certain parts of the tree that do not contain any learning vectors. In addition, the corresponding models in the affected individual leaves are recalculated. Due to performance reasons, the coefficients in the existing linear models are recalculated to fit a randomly selected sample of the actual data (no more than 50 instances) in the corresponding leaves.

Each crossover begins with randomly selecting two individuals from the population that will be affected. Next, the crossover points in both individuals are determined. We have adapted all variants proposed in the univariate tree inducer [12] to work with the mixed representation, visualized in Fig. 6:

- (a) *exchange subtrees*: exchanged of subtrees starting in randomly selected nodes;
- (b) *exchange branches*: exchanges of branches that starts from selected nodes in random order;
- (c) *exchange tests*: recombines the tests (univariate nominal, univariate continuous-valued, and oblique) associated with randomly selected internal nodes;
- (d) *with best*: crossovers with the best individual;
- (e) *asymmetric*: duplicates subtrees with small mean absolute errors and replaces nodes with high errors.

Selected nodes for the recombination must have the same number of outputs; however, they may have different representations. This way crossovers shift not only the tree structure but also the nodes' representations. In the variants (d) *with best* and

(e) *asymmetric*, the additional mechanism is applied to decide which node would be affected. The algorithm ranks all tree nodes in both individuals according to their absolute error divided by the number of instances in the node. The probability of selecting nodes is proportional to the rank in a linear way. The nodes with a small average error per instance are more likely to be donors, whereas the weak nodes (with a high average error per instance) are more likely to be replaced by the donors from the second individual (and have a higher probability of becoming receivers).

The mutation of an individual starts with the selection of a node type (equal probability of selecting a leaf or an internal node). Next, a ranked list of nodes of the selected type for this individual is created. Depending on the type of node, the ranking takes into account the location for internal nodes (nodes in the lower parts of the tree are mutated with higher probability) and the prediction error of the node (nodes with a higher error per instance are more likely to be mutated). Finally, a mechanism analogous to the ranking linear selection [32] is applied to decide which node in the individual will be affected. Depending on the node's representation, different variants of operators are available in internal nodes:

- *prune*: changes internal node to a leaf (acts like a pruning procedure);
- *parent with child (branches)*: replaces a parent node with a randomly selected child node (internal pruning);
- *parent with child (tests)*: exchanges tests between parent and randomly selected child nodes;
- *new dipolar test*: tests in affected node is reinitialized by a new one selected using the dipolar strategy;
- *new memetic test*: tests in node is reinitialized by one of the optimality strategies proposed in Section 3.2;
- *modify test*: shifts hyperplane or set random weights (oblique test); shifts threshold (univariate test on a continuous attribute) or re-groups nominal attribute values by adding/merging branches or moving values between them;
- *recalculate models*: recursively recalculates linear models using all the instances in the corresponding leaves;

and in the leaves:

- *dipolar expand*: transforms leaf into internal node with a new dipolar test (random type);
- *memetic expand*: transforms leaf into internal node with a new test selected by one of the optimality strategies;
- *change model*: extends/simplifies/changes the linear model in the leaf by adding/removing/replacing a randomly chosen attribute or removing the least significant one.

For a more detailed description of mutation variants, please refer to [12].

In addition, we propose a new mechanism called Switch that assures the diversity of node representations within the population. It is embedded in the specified variants of the mutation (*prune*, *expand*, and *new test*) that require finding new tests in the internal nodes or models in the leaves. The Switch mechanism with assigned probability changes the initial representation of the selected nodes:

- the test in the internal node when calculating a new test with the same number of outputs:
  - with the change from univariate to oblique (internal nodes), a new calculated hyper-plane involves an attribute from the univariate test;

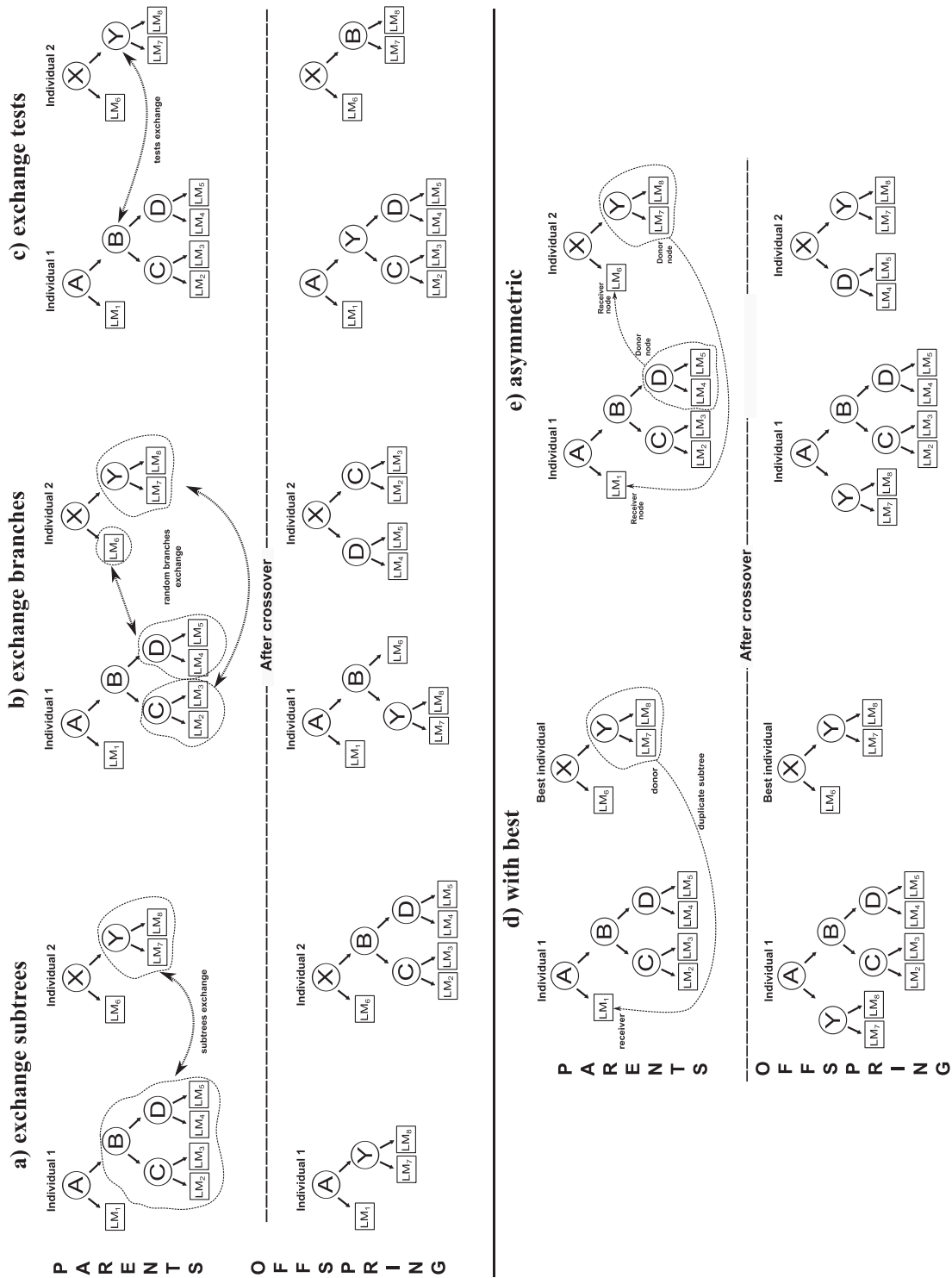


Fig. 6. Visualization of crossovers, from top left to bottom right: (a) exchange subtrees, (b) exchange branches, (c) exchange tests, (d) with best, and (e) asymmetric.

- with the change from oblique to univariate (internal nodes), a new univariate test is based on a randomly selected attribute from the oblique test.
- newly created nodes that inherit their representation from the initial representation
  - leaves flip representation from the regression constant value to linear regression model (or vice versa) when pruning internal nodes;
  - internal nodes flip representation from the oblique test to the univariate one (or vice versa) when expanding the leaves.

In the rest of the mutation variants, the Switch mechanism is not applied. Preserving the representation in, for example, the *modify test* or *change model* variant allows exploring the neighborhood space of solutions rather than starting the search from a new place.

### 3.5. Selection, termination condition, and smoothing

The ranking linear selection is applied as a selection mechanism. In each generation, the single individual with the highest value of the fitness function in the current population is copied to the next

one (*elitist strategy*). Evolution terminates when the fitness of the best individual in the population is not improved during the fixed number of generations (default: 1000). In the case of a slow convergence, the maximum number of generations is also specified (default value: 10,000) to limit the computation time.

The mGMT system uses a form of smoothing that was initially introduced in the M5 algorithm [46] for a univariate model tree. As in the basic GMT solution [12], the smoothing is applied only to the best individual returned by EA when the evolutionary induction is finished. The role of the smoothing is to reduce sharp discontinuities that occur between adjacent linear models in the leaves. For every internal node of the tree, the smoothing algorithm generates an additional linear model that is constituted from features that occur along the path from the leaf to the node. This way, each tested instance is predicted not only by a single model at a proper leaf but also by the different linear models generated for each of the internal nodes up to the root node. Due to the oblique splits that may appear in the tree induced by the mGMT system, we have updated the smoothing algorithm to use all attributes that constitute the tests in the internal nodes.

#### 4. Experimental validation

To verify the role of tree representations, we have performed experiments on both artificial and real life datasets. In the first section below, the impact of the tree representation is assessed using four algorithms with different homogeneous representations and the proposed mGMT inducer. Next, the mGMT solution is compared with the results from paper [23] that cover experiments with popular tree inducers on publicly available datasets. Finally, the prediction performance of the proposed solution is tested on a larger group of publicly available datasets.

In all experiments reported in this section, a default set of parameters for all algorithms is used in all tested datasets. Results presented in the paper correspond to averages of 50 runs.

##### 4.1. Role of the tree representation

In this section, five types of tree representations are analyzed:

- univariate Global Regression Tree (denoted as uGRT) that has axis-parallel decision borders and simple constant predictions in the leaves;
- univariate Global Model Tree (uGMT) that has axis-parallel decision borders and multivariate linear regression models in the leaves;
- oblique Global Regression Tree (oGRT) that constructs oblique splits on binary or continuous-valued attributes in the internal nodes;
- oblique Global Model Tree (oGMT) – the most complex tree representation (oblique splits and multivariate linear regression models);
- mixed Global Model Tree (mGMT) that self-adapts the tree representation to the currently analyzed data.

The first four algorithms are based on the existing solutions [10–12], and the proposed mGMT algorithm can be treated as an extension and unification.

The impact of representation on the tree performance is tested on two sets of artificially generated datasets:

- *armchair* – variants of the dataset proposed in [11] that require at least four leaves and three splits;
- *noisy* – datasets with various data distributions and additional noise.

**Table 1**  
Default parameters of uGRT, uGMT, oGRT, oGMT and mGMT.

Parameter	Value
Population size	50 individuals
Crossover rate	20% assigned to the tree
Mutation rate	80% assigned to the tree
Elitism rate	2% of the population (1 individual)
Maximum amount of generation without improvement	1000
Max total number of generation	10,000

All artificial datasets have analytically defined decision borders that fit to particular tree representations: univariate regression (UR), univariate model (UM), oblique regression (OR), oblique model (OM), and mixed (MIX). Each set contains 1000 instances, where 33% of the instances constitute the training set and the rest of the instances constitute the testing set. A visualization and description of the artificial datasets are included in [Appendix](#).

##### 4.1.1. Parameter tuning

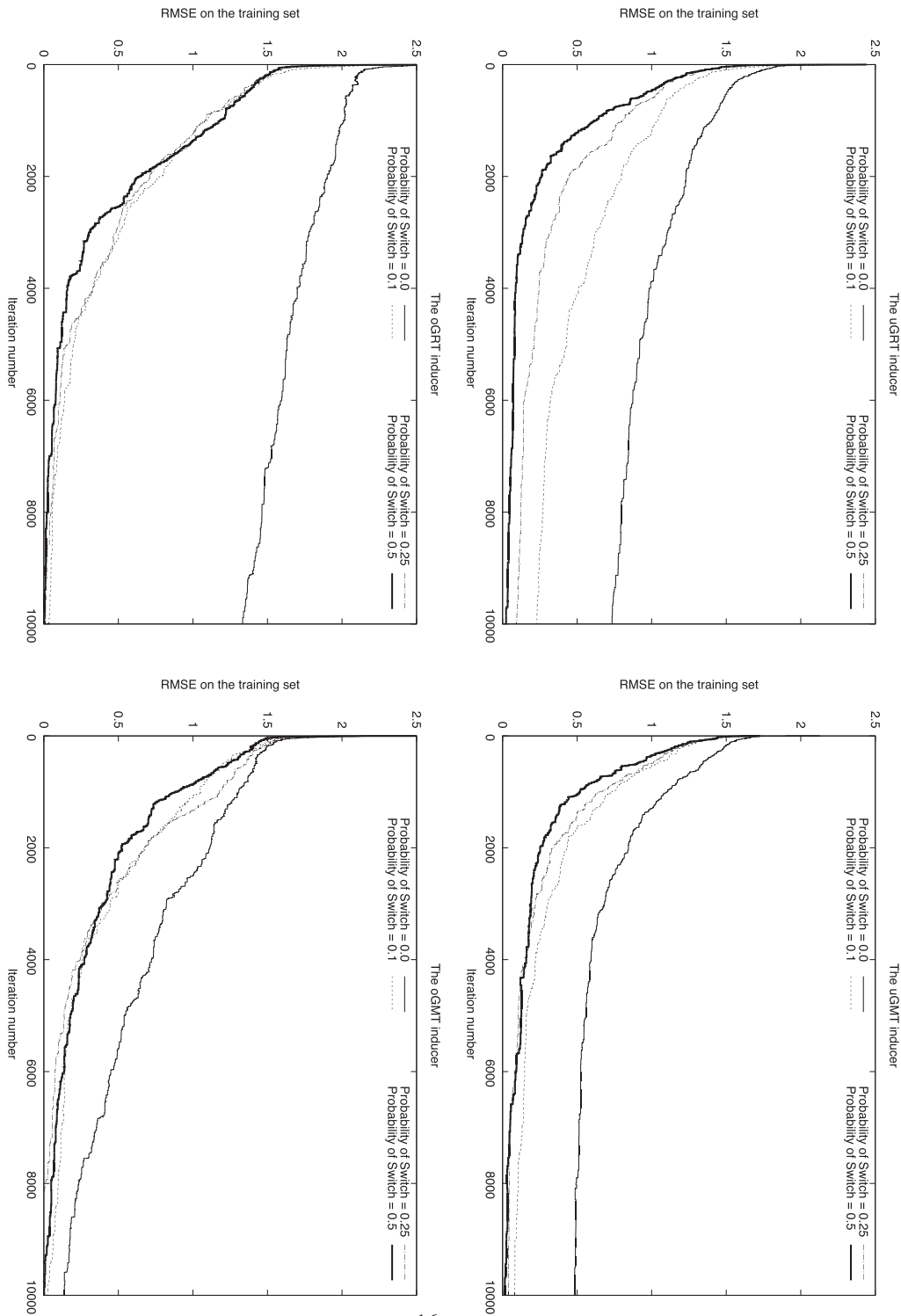
Parameter tuning for EAs is a difficult task. Hopefully, all important EA parameters (e.g., population size, the probability of mutation and crossover, etc.) and the decision tree parameters (maximum size, minimum objects to make a split) were experimentally validated and tuned in previous papers for trees with homogeneous representations [12]. Those general settings should also work well with the mixed regression trees; therefore, they can be treated as default. The main parameter for all algorithms is given in [Table 1](#) and the probabilities of selecting mutation operator variants are shown in [Table 2](#) (the probability of selecting each crossover variant is equal to 20%). This way, only the role of the Switch mechanism that is embedded in different variants of mutation operators and directly switches the node representation, for example, from univariate to oblique in the internal node and from constant prediction to multivariate linear regression model in the leaf, should be investigated.

Parameter tuning was performed on the *armchair* dataset (version AMix1) according to the guidelines proposed in [14]. Four different Switch mechanism values that correspond to the probability of node representation change were tested: 0.0, 0.1, 0.25, and 0.5. The impact of this setting on the proposed mGMT solution and on the rest of the tree inducers with a homogeneous initial population was checked. For example, when the uGRT algorithm is evaluated and the Switch mechanism is enabled, then the representation of mutated nodes with assigned probabilities can change. This way, the algorithm can have a mixed representation and is able to have oblique splits or multivariate regression models in the leaves. [Figs. 7 and 8](#) show the tree error (RMSE) of the best

**Table 2**  
Probability of selecting a single variant of the mutation operator in uGRT, uGMT, oGRT, oGMT, and mGMT.

Mutation operator	Probability in:	
	uGRT &oGRT	uGMT, oGMT &mGMT
<i>prune</i>	30	20
<i>parent with son (branches)</i>		5
<i>parent with son (tests)</i>		2.5
<i>new dipolar test</i>		10
<i>new memetic test</i>		2.5
<i>modify test</i>		15
<i>recalculate models</i>		2.5
<i>dipolar expand</i>	30	20
<i>memetic expand</i>		2.5
<i>change model</i>	0	20





16

Fig. 7. Impact of the Switch mechanism on the best individual for the uGRT, uGMT, oGRT, and oGMT inducers on the *armchair AMix1* dataset.

individual during the learning phase performed on the training set for all five algorithms: uGRT, uGMT, oGRT, oGMT, and mGMT.

One can observe that the impact of the Switch mechanism is especially visible for the algorithms with homogeneous initial populations. In the Fig. 7, enabling the Switch is the only way to find optimal solutions for the uGRT, oGRT, and uGMT algorithms. When the Switch is set to 0.5, which equals to the random representation selection, the inducers have the fastest convergence. In the oGMT

algorithm, which is capable of finding the optimal solution on its own, the application of the Switch mechanism shortens the inducers' convergence time. A statistical analysis of the results using the Friedman test and the corresponding Dunn's multiple comparison test (significance level equals 0.05), as recommended by Demsar [13], showed that there exists significant differences between the Switch parameter setting for all four algorithms with strict representations. The performed experiments showed that the optimal

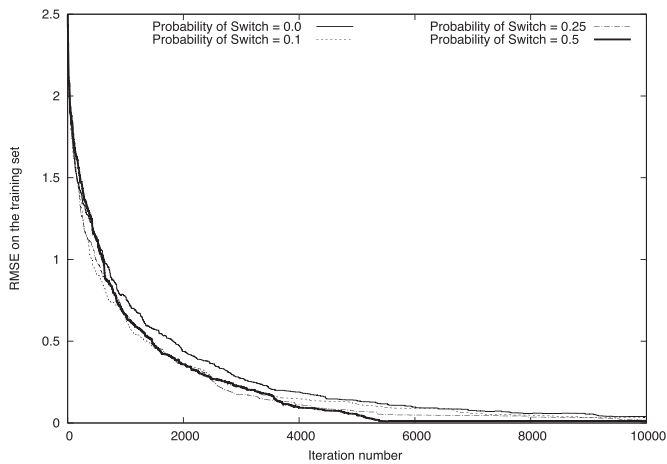


Fig. 8. Impact of the Switch mechanism on the best individual for the mGMT inducer on the *armchair AMix1* dataset.

Switch settings for the inducers with homogeneous representation is 0.5, which equals a random representation of the newly created node.

The mGMT results visualized in Fig. 8 show there are no big differences between the algorithms with various Switch settings. This can be explained by the construction of the initial population of the algorithm, which is composed of five types of representations. The individual representations can be successfully combined with the crossover operators. However, we can observe a slight improvement in the algorithm convergence to the optimal solution when the Switch mechanism is enabled.

4.1.2. Comparison of representations

To show the impact of tree representation, five inducers were tested on two groups of datasets, *armchair* and *noisy* (each set with six variants), described in Appendix. Four metrics were collected and illustrated:

- Root Mean Squared Error (RMSE) calculated on the testing set (Fig. 9);
- average number of leaves in the tree (Fig. 10);

- average number of attributes in the regression models in the leaves (Fig. 11). Univariate inducers are not shown as the average number of tests is always equal to their size decreased by 1;
- average number of attributes in the tests in the internal nodes (Fig. 12). Regression inducers are not shown as there are no models in the leaves; therefore, the average number of attributes is always equal to zero.

All four figures should be analyzed at the same time to understand how each global inducer works.

Artificial datasets were designed to be solved by one of the tested systems and the abbreviations of datasets reveal which inducer is most appropriate to use. In general, all inducers with the appropriate individual representation managed to successfully induce the defined tree. However, when the representation does not fit the specifics of the dataset, it is too simple (univariate split, regression leaf) or too advanced (oblique split, model in the leaf), and the evolutionary inducers with homogeneous representations sometimes have difficulty finding an optimal solution. In contrast to the four global inducers with defined representations (uGRT, oGRT, uGMT, and oGMT), the mGMT system has flexible representation. The results presented in Figs. 9–12 show that mGMT successfully adapts the tree structure to the specifics of each artificially generated dataset. In the datasets denoted as *UR*, *UM*, *OR*, and *OM*, the mGMT system managed to keep up with the algorithms whose structure fitted the characteristics of the datasets. As for the *Mix* dataset variants, mGMT managed to outperform the rest of the tree inducers.

There are at least two reasons why the systems with strict representations of the individuals have difficulty with some variants of the datasets. The first reason is the limitation in the individuals' representation. The no axis-parallel decision borders can easily be solved with oGRT or oGMT algorithms. The application of univariate splits may cause the 'staircase effect' [8]. This problem is similar for the regression trees applied for the *UM* and *OM* datasets that require regression models in the leaves. To overcome these restrictions in the representation, regression trees (uGRT, uGMT, oGRT) increase their tree sizes; however, the limitation still exists. The large size of the induced tree influences not only its clarity but may cause overfitting to the training data and thus a larger prediction error. Let us explain this for different variants of the *armchair* dataset described in Appendix:

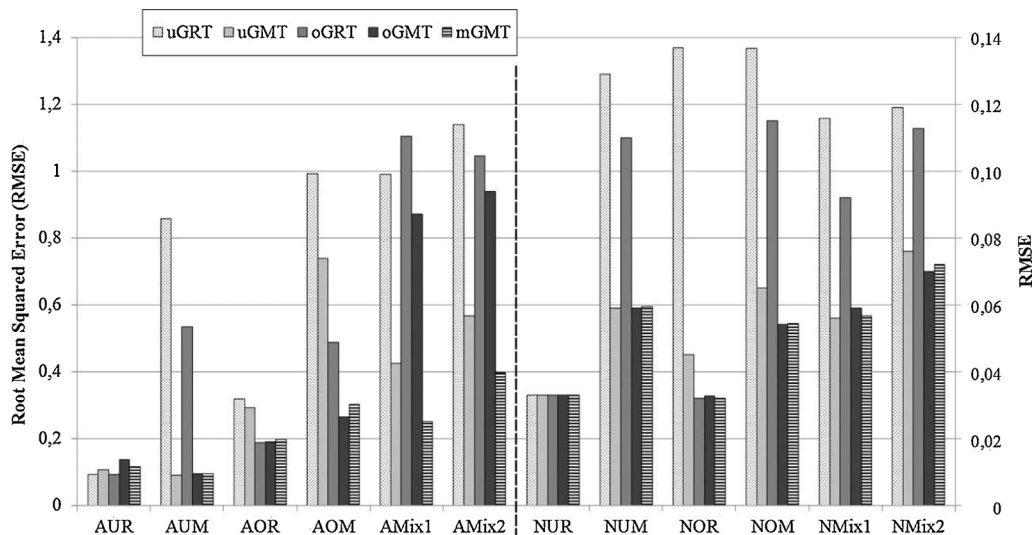


Fig. 9. Relative Mean Squared Error (RMSE) of the algorithms on 12 artificial datasets described in Appendix. Tested algorithms: univariate Global Regression Tree (uGRT), oblique Global Regression Tree (oGRT), univariate Global Model Tree (uGMT), oblique Global Model Tree (oGMT), and mixed Global Model Tree (mGMT). For illustrative purposes, the values of the RMSE error for the *noisy* dataset have been rescaled.

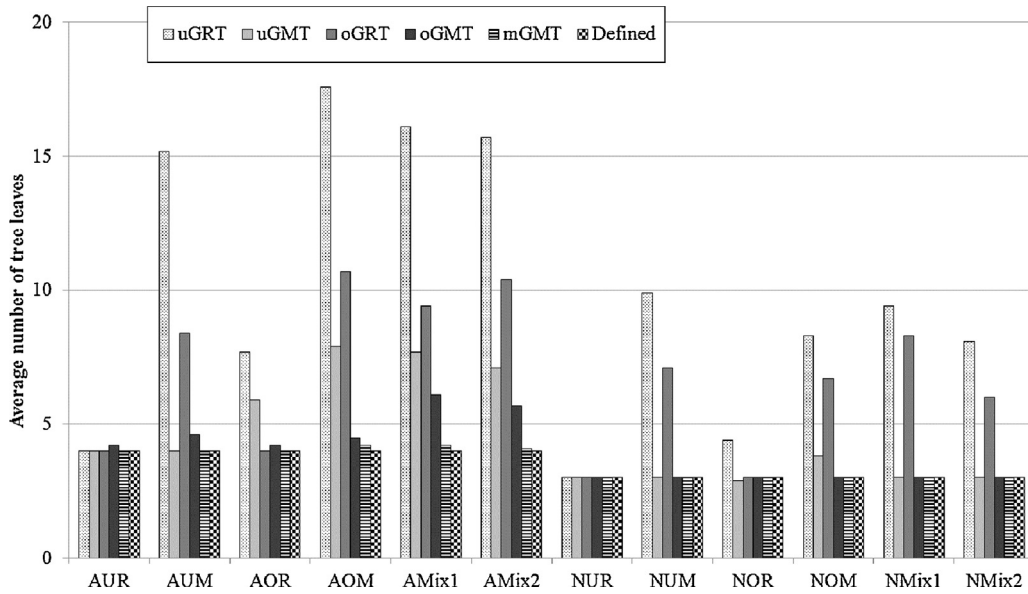


Fig. 10. Average number of leaves in the tree for different GMT variants. The defined bars represent the reference values that are equal to the optimal numbers of leaves for the datasets.

- *AUR* – can be perfectly predicted by univariate regression trees. All aforementioned inducers are capable of finding decision trees with small RMSE (Fig. 9), four leaves (Fig. 10), three univariate splits (Fig. 11), and no regression model in the leaves (Fig. 12). Even the oGMT system managed to find the decision borders despite its advanced node representation of the individuals. The univariate split is just a special case of an oblique split, and a constant value is just a special case of a regression model.
- *AUM* – can be perfectly predicted by univariate model trees. This dataset is difficult for the uGRT and oGRT systems because they induce only the regression trees. For these systems, we can observe a much higher error rate (*RMAE*) and trees that are 2–3 times larger. It is typical for the regression trees to reduce the tree error by adding many leaves with a small number of instances. In addition, the oGRT inducer applied unnecessary oblique splits in order to minimize *RMSE*. The rest of the algorithms had no problem with this dataset and induce trees with four leaves, three

univariate splits, and usually perfect regression models in the leaves.

- *AOR* – can be perfectly predicted by oblique regression trees. The application of the algorithms with univariate tests (uGRT and uGMT) to the dataset with non-axis parallel decision borders led to their approximation by a very complicated stair-like structure.
- *AOM*, *AMix1*, and *AMix2* – can be perfectly predicted only by the inducers with the most advanced tree representation (oblique splits and models in the leaves). Therefore, it is not surprising that the algorithms uGRT, oGRT, and uGMT induce overgrown decision trees. It is worth noting that of those three systems, the largest trees are induced by the system that has the most limitations in the representation of the individuals – the uGRT.

The second issue is the large search space of the inducers with advanced tree representation that requires extensive calculations to find a good solution. It can be observed especially for the trees

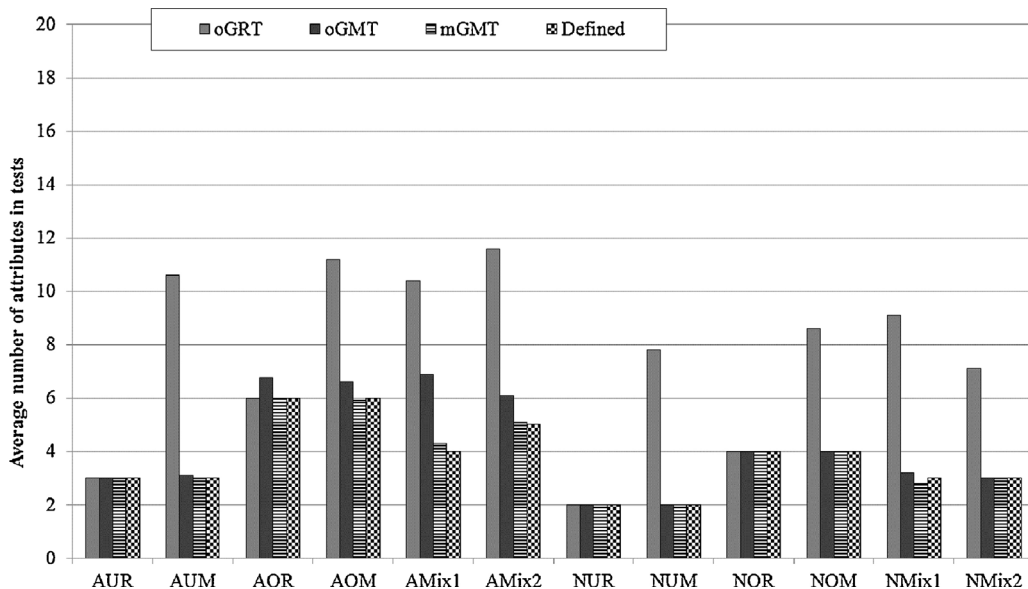


Fig. 11. The sum of an average number of attributes used in the internal node tests for different GMT variants. The defined bars are equal to the optimal numbers of attributes in the internal node tests.

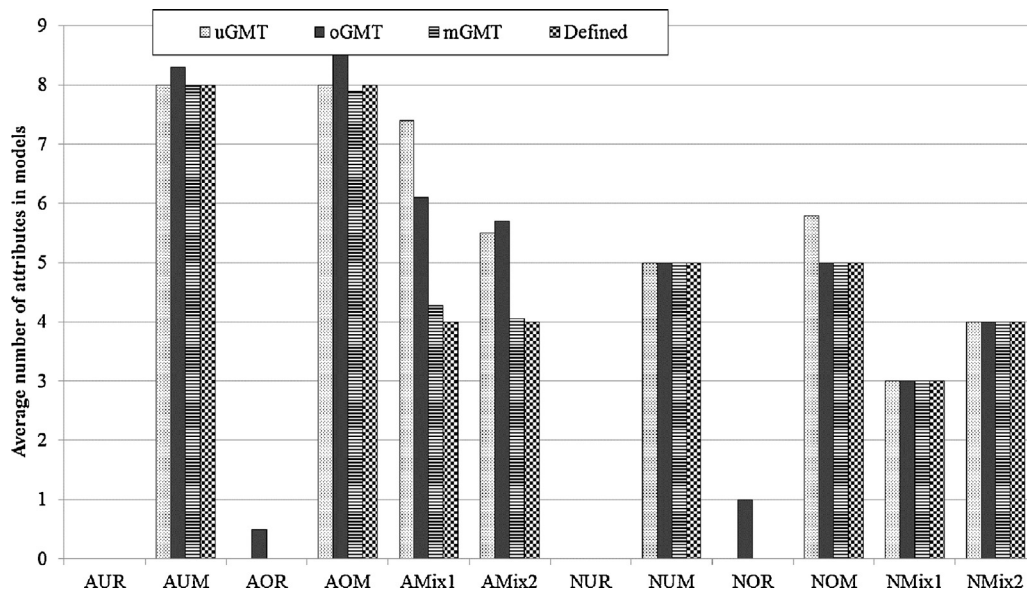


Fig. 12. The sum of an average number of attributes that constitute leaves' models for different GMT variants. When the induced tree has only regression leaves, then no value appears on the chart as in the AUR or NUR dataset. The defined bars are equal to the optimal numbers of attributes in the leaves' modes.

with oblique splits. Theoretically, the oGMT system should be able to find optimal decisions in all datasets as it induces trees with the most complex representation. However, we can observe that the trees induced by the oGRT and oGMT systems do not always have an optimal structure (even if they are capable of finding it). For the simplest datasets like AUR, the inducers with oblique splits need significantly more time than the uGRT solution (which finds optimal decisions almost instantly). This situation is illustrated in Fig. 13. Although the mGMT system needed additional iterations to set the appropriate tree representation, it still outperforms oGRT and oGMT. In Fig. 13, we can see that the largest number of iterations is required by the inducers with oblique splits in the internal nodes. The oGRT and oGMT systems needed significantly more iterations than uGRT but managed to successfully reduce the prediction error calculated on the training set to zero. It can be seen that the oGMT inducer did not find the optimal tree size for all 50 runs. For a few runs, the oGMT algorithm needed over 10,000 iterations, but additional experiments showed that it is capable of finding optimal

trees. In addition, the loop time for global inducers differs significantly, as different variants of mutation operators are applied. The average loop times (in seconds) calculated for all iterations of all artificial datasets are shown in Table 3.

All observations made for the *armchair* dataset are also confirmed for the *noisy* dataset. The mGMT solution managed to find all defined splits and models despite the noise and different data distributions. From the dataset visualization included in Appendix, it can be seen that finding appropriate decision borders is not an easy task. The oGMT usually kept up with mGMT because the decision tree was smaller (the defined tree has two internal nodes and three leaves).

From the performed experiments, we can observe that every inducer with the strict tree representation has its pros and cons. The systems for univariate regression trees are very fast and generate simple tests in internal nodes; however, the tree error and size are usually large. Oblique regression trees are slightly smaller and more accurate, but the searching of the splitting rules is much more computationally demanding and the simplicity of the output tree is lost. The results generally confirm what is observed for the univariate and oblique classification trees. Currently, the most popular trees for the regression problems are univariate model trees. From the results, we see that they have a good trade-off between the tree complexity and the prediction performance; induced trees are accurate and relatively small. Theoretically, if the computational complexity of the algorithm was not an issue, the oblique model trees should be as good as all aforementioned algorithms in terms of prediction power. Unfortunately, the induction time and the complexity of the solution often hinder the practical application of the inducer, especially for the large datasets.

If we knew the characteristics of the dataset we could pre-select the inducer with the most appropriate representation. However, this is often not the case; therefore, it may be better to consider

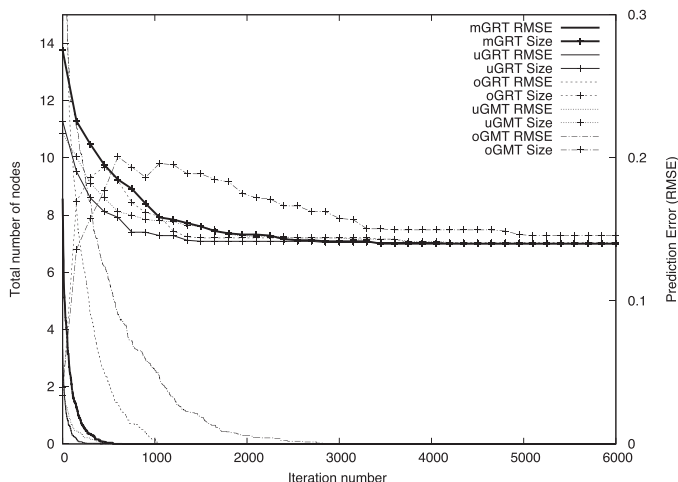


Fig. 13. Influence of the tree representation on the performance of the best individual on the AUR training set for 5 inducers.

Table 3

Average single loop times of all iterations of all datasets for different systems.

Algorithm	uGRT	uGMT	oGRT	oGMT	mGMT
Average time	0.0013	0.0036	0.0017	0.0043	0.0024
$\pm$ (stdev)	0.0002	0.0004	0.0005	0.0010	0.0003

a system like mGMT that is able to self-adapt to the analyzed data. The mGMT system managed to gain advantages from all four inducers and limit the disadvantages. It has the best performance in terms of tree error and size among all systems for all datasets. According to the Friedman test, there exists significant differences in *RMSE* and the tree size between induced trees in favor of mGMT (significance level equal to 0.05, *P* value < 0.0001, *F*-statistic = 175.8). The corresponding Dunn's multiple comparison test showed a significant difference in rank sum for *RMSE* between the mGMT and uGRT systems and between the mGMT and oGRT systems, and for tree size between themGMT and all other systems.

#### 4.2. mGMT vs. popular tree approaches

In this set of experiments, we compared the proposed mGMT inducer with different popular tree approaches. In order to make a proper comparison with the state of the art and the latest algorithms in the literature, we selected the benchmark datasets also used in [23]. We precisely followed the preprocessing and the experimental procedure in [23] to make the comparison to the results of that paper as accurate as possible. Two popular synthetic datasets and two real-life datasets from the well-known UCI Machine Learning Repository [5] were used:

- Fried – artificial dataset proposed by Friedman [25] containing ten independent continuous attributes uniformly distributed in the interval [0,1]. The value of the output variable is obtained with the equation:

$$y = 10 * \sin(\pi * x_1 * x_2) + 20 * (x_3 - 0.5)^2 + 10 * x_4 + 5 * x_5 + \sigma(0, 1);$$

- 3DSin – artificial dataset containing two continuous predictor attributes uniformly distributed in interval [3,3], with the output defined as

$$y = 3 * \sin(x_1) * \sin(x_2);$$

- Abalone – dataset used to predict the age of abalone from physical measurements (4177 instances with eight attributes – one nominal and seven continuous);
- Kinman – dataset containing information on the forward kinematics of an eight link robot arm (8192 instances with eight continuous attributes).

As in previous research [23], 3000 points were generated and the data was normalized to zero mean and unit variance for both artificial datasets.

We recalled the results of four algorithms, performed testing using WEKA software [19] on two additional tree inducers, and included the results for our mGMT system:

- Hinge algorithm [23] that is based on on hinging hyperplanes identified by a fuzzy clustering algorithm;
- FRT – fuzzy regression tree;
- FMID – fuzzy model identification;
- CART – state-of-the-art univariate regression tree proposed by Breiman et al. [7];
- REPTree (RT) – popular top-down inducer that builds a univariate regression tree using variance and prunes it using reduced-error pruning (with backfitting);
- M5 – state-of-the-art univariate model tree inducer proposed by Quinlan [46];
- mGMT – proposed global tree inducer with mixed representation.

The performance of the models is measured by the (RMSE), a well known regression performance estimator. Testing was performed with 10-fold cross-validation, and 50 runs were performed for the tested (by the authors) algorithms. We have also included the information about the algorithms' standard deviation (unfortunately [23], do not include this information). The results shown in Table 4 indicate that the mGMT solution can successfully compete with popular decision tree inducers.

As the mean value is not presented in the research [23], we have performed Friedman tests (significance level equal to 0.05) using RMSE error values on two groups:

- mGMT vs Hinge, FRT, FMID and CART;
- mGMT vs uGRT, uGMT, oGRT, oGMT, RT and M5.

**Table 4**  
Comparison of RMSE results of different algorithms. Algorithms with \* were tested in [23] and their results are recalled. Results for mGMT also include the standard deviation of RMSE and the number of leaves in the tree. The smallest RMSE and size results for each dataset are bolded.

Algorithm	Metric	Fried	3DSin	Abalone	Kinman
Hinge*	RMSE	0.92	0.18	4.1	0.16
	Leaves	<b>8</b>	<b>11</b>	8	<b>6</b>
CART*	RMSE	2.12	0.17	2.87	0.23
	Leaves	495.6	323.1	664.8	453.9
FMID*	RMSE	2.41	0.31	2.19	0.20
	Leaves	12	12	12	12
FRT*	RMSE	0.70	0.18	2.19	0.15
	Leaves	15	12	4	20
RT	RMSE	2.25 ± 0.10	0.6 ± 0.01	2.33 ± 0.13	0.19 ± 0.01
	Leaves	445.7 ± 37.6	724.2 ± 30.1	168.8 ± 33.7	720.8 ± 78.1
M5	RMSE	1.81 ± 0.09	0.23 ± 0.01	<b>2.12 ± 0.14</b>	0.16 ± 0.01
	Leaves	52.5 ± 13.5	197.3 ± 11.8	8.59 ± 3.2	<b>109.7 ± 18.0</b>
mGMT	RMSE	<b>0.67 ± 0.01</b>	<b>0.15 ± 0.003</b>	2.13 ± 0.08	<b>0.14 ± 0.001</b>
	Leaves	14.9 ± 2.2	53.6 ± 8.9	<b>2.1 ± 0.7</b>	6.4 ± 1.3
uGRT	RMSE	3.66 ± 0.09	0.53 ± 0.04	2.55 ± 0.03	0.21 ± 0.007
	Leaves	11.5 ± 0.8	40.0 ± 0.54	4.4 ± 0.34	11.4 ± 1.2
uGMT	RMSE	0.66 ± 0.01	0.15 ± 0.003	2.19 ± 0.001	0.16 ± 0.002
	Leaves	16.4 ± 0.43	56.3 ± 1.9	2.1 ± 0.03	8.6 ± 0.6
oGRT	RMSE	3.41 ± 0.05	0.62 ± 0.008	2.50 ± 0.10	0.19 ± 0.01
	Leaves	5.7 ± 0.03	22.5 ± 1.3	3.4 ± 0.05	6.6 ± 0.2
oGMT	RMSE	1.13 ± 0.02	0.15 ± 0.01	2.21 ± 0.05	0.17 ± 0.001
	Leaves	6.6 ± 0.4	44.7 ± 1.9	2.1 ± 0.09	4.4 ± 0.2

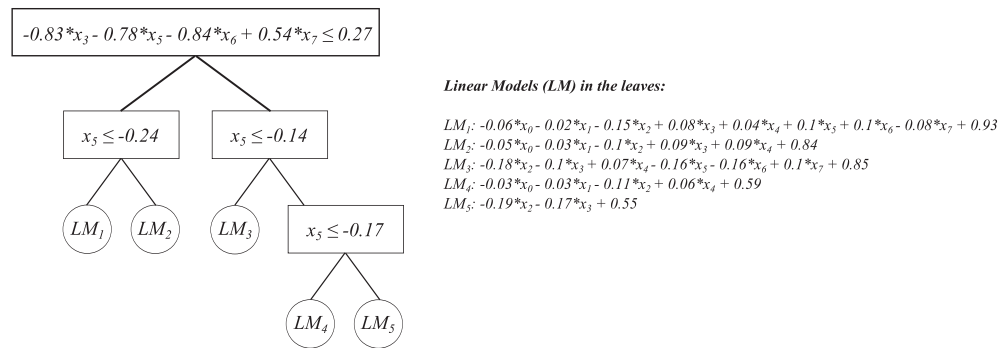


Fig. 14. An example of induced tree for mGMT for the Kinman dataset.

For first group, the Friedman test showed significant statistical differences between algorithms ( $P$  value = 0.0109,  $F$ -statistic = 10.62); however, a Dunn’s multiple comparison test did not show any significant differences in rank sum, which may be caused by a small sample size (only four values for five algorithms). For the second group, a Friedman test also showed significant statistical differences between algorithms ( $P$  value < 0.0001,  $F$ -statistic = 194.6). A corresponding Dunn’s multiple comparison test showed significant differences in rank sum between mGMT and all algorithms except uGMT. It should also be noted that mGMT managed to induce much smaller trees, often by an order of magnitude smaller than the tested counterparts. A relatively higher number of leaves for the mGMT inducer for the *3DSin* and *Fried* datasets can be explained by high non-linearity in the datasets. As the mGMT applies multivariate linear regression functions in the leaves, it requires more splits to fit to the non-linear datasets characteristics.

The cost of finding possibly new hidden regularities is the tree induction time. It is well known that the EAs in comparison to the greedy solutions are slower, and the mGMT is no exception. The efficiency comparison between mGMT and both tested greedy inducers showed that the proposed solution is significantly slower (verified with Friedman test,  $P$  value < 0.0001) than both algorithms: *M5* and *RT*. The mGMT tree induction time was smaller to that of the GMT solution [12] (Table 3) and took, depending on the dataset, from several seconds to a few minutes on a regular PC computer. However, the process of evolutionary induction is progressive; therefore, intermediate solutions from pre-maturely aborted runs may also yield high-quality results. In addition, EAs are naturally prone to parallelism; therefore, the efficiency problem can be partially mitigated.

In Fig. 14, we present one of the trees induced by mGMT for the Kinman dataset. For this particular real-life dataset, all induced trees contained oblique and univariate splits and almost always multivariate linear regressions in the leaves. This may suggest that this mixed representation is the most suitable one for this particular

dataset and may reveal new relationships and information hidden in the data. The output tree is much smaller and has the smallest prediction error, especially when compared to the results of state-of-the-art solutions like CART and *M5*. However, it should be noticed that in case of the mixed, oblique or model trees the size of the tree is not an accurate reflection of its complexity. The trees with more advanced tree representation are usually smaller which is why the *M5* algorithm induces much smaller trees than CART. Therefore, even very small tree induced by the mGMT but with complex oblique splits and models in the leaves can be less comprehensible than, for example, larger univariate regression tree. In an extreme scenario, the proposed solution can be as complex as trees induced by the oGMT system or as simple as ones induced by the uGRT algorithm. However, mGMT is capable of adjusting the representation of the nodes to automatically fit to the analyzed which is not possible in the competitive solutions which have only homogeneous tree representation. Although, the trade-off between the comprehensibility and prediction performance in mGMT still exists, it can be easily adjusted to the user preferences due to the parameters in the fitness function of the mGMT algorithm.

### 4.3. Overall prediction performance of mGMT

In the last step of the experiments, we compared the prediction performance of the mGMT inducer with that of other popular systems on multiple datasets. Tests were performed with WEKA software [19] using the collection of benchmark regression datasets provided by Louis Torgo [45]. From this package of 30 datasets (available on the WEKA page), we selected only those with a minimum of 1000 instances, described in Table 5. We decided that datasets with, for example, 43 instances and two variables are not the best for validation. The datasets have been processed by WEKA’s supervised NominalToBinary filter that converts nominal attributes into binary numeric attributes and the unsupervised ReplaceMissingValues filter that replaces missing values with the attributes’

Table 5

Dataset characteristics: name, numeric attributes number (Num), nominal attributes number (Nom), and the number of instances.

ID	Name	Num	Nom	Instances	ID	Name	Num	Nom	Instances
1	<i>2dplanes</i>	10	0	40,768	11	<i>elevators</i>	18	0	8752
2	<i>abalone</i>	7	1	4177	12	<i>fried</i>	10	0	40,768
3	<i>aileron</i> s	40	0	13,750	13	<i>house 16H</i>	16	0	22,784
4	<i>bank32nh</i>	32	0	8192	14	<i>house 8L</i>	8	0	22,784
5	<i>bank8FM</i>	8	0	8192	15	<i>kin8nm</i>	8	0	8192
6	<i>cal housing</i>	8	0	20,640	16	<i>mv</i>	7	3	40,768
7	<i>cpu act</i>	21	0	8192	17	<i>pol</i>	48	0	15,000
8	<i>cpu small</i>	12	0	8192	18	<i>puma32H</i>	32	0	8192
9	<i>delta aileron</i> s	5	0	7129	19	<i>puma8NH</i>	8	0	8192
10	<i>delta elevator</i> s	6	0	7129					

**Table 6**  
Comparison results of RMSE for tested systems. The standard deviation is shown for nondeterministic methods. Occasional catastrophic failures of systems for particular datasets (with a minimum 2× higher RMSE value in comparison to the score achieved by the best algorithm for the dataset) are bolded.

ID	mGMT	uGRT	uGMT	oGRT	oGMT	LR	RT	M5	SMO	AR	BG	NN
1	0.996 ±7E-3	0.998 ±0.01	0.996 ±0.01	1.059 ±0.01	1.01 ±0.02	<b>2.378</b>	1.045	0.996	<b>2.380</b>	<b>2.474</b>	1.04	1.481
2	2.220 ±6E-3	2.357 ±0.07	2.218 ±1E-3	2.321 ±0.02	2.283 ±4E-3	2.186	2.299	2.131	2.206	2.365	2.145	2.244
3	1.6E-4 ±1E-7	2.0E-4 ±3E-6	1.6E-4 ±1E-6	2.2E-4 ±2E-5	1.7E-4 ±3E-7	1.7E-4	2.0E-4	1.6E-4	1.7E-4	2.2E-4	1.8E-4	2.6E-4
4	0.083 ±1E-4	0.099 ±1E-3	0.084 ±1E-3	0.087 ±7E-4	0.084 ±2E-4	0.083	0.094	0.082	0.089	0.092	0.087	<b>0.142</b>
5	0.029 ±2E-4	0.040 ±4E-4	0.030 ±2E-4	0.036 ±6E-4	0.030 ±5E-4	0.038	0.040	0.030	0.038	0.057	0.033	0.033
6	7.4E4 ±6E3	9.1E4 ±4E3	8.5E4 ±7E3	8.5E4 ±1E4	7.2E4 ±7E3	7.5E4	9.7E4	13.3E4	7.4E4	8.8E4	7.8E4	7.1E4
7	2.654 ±0.06	3.420 ±0.06	2.643 ±0.06	3.594 ±0.21	2.566 ±0.08	<b>9.288</b>	3.294	2.655	<b>10.686</b>	4.145	2.794	4.041
8	3.263 ±0.08	3.811 ±0.02	3.276 ±0.11	3.947 ±0.13	3.439 ±0.50	<b>9.478</b>	3.945	3.284	<b>10.772</b>	4.655	3.220	3.569
9	1.7E-4 ±2E-6	1.8E-4 ±4E-6	1.6E-4 ±3E-6	1.8E-4 ±2E-6	1.7E-4 ±1E-6	1.7E-4	1.8E-4	1.7E-4	1.7E-4	1.8E-4	1.7E-4	1.8E-4
10	1.4E-3 ±1E-6	1.4E-3 ±7E-6	1.4E-3 ±1E-6	1.4E-3 ±1E-6	1.4E-3 ±1E-6	1.4E-3	1.4E-3	1.4E-3	1.4E-3	1.5E-3	1.4E-3	1.4E-3
11	2.3E-3 ±2E-5	3.8E-3 ±9E-5	2.3E-3 ±2E-5	<b>4.8E-3</b> ±7E-4	2.5E-3 ±1E-4	2.8E-3	3.9E-3	2.5E-3	2.9E-3	<b>4.7E-3</b>	3.2E-3	2.2E-3
12	1.067 ±0.01	2.007 ±0.05	1.056 ±2E+3	2.049 ±0.08	1.551 ±0.05	<b>2.638</b>	1.920	1.554	<b>2.662</b>	<b>2.497</b>	1.496	1.534
13	3.9E4 ±4E3	4.5E4 ±1E3	4.1E4 ±4E3	4.2E4 ±2E3	3.9E4 ±1E3	4.4E4	3.9E4	3.6E4	4.6E4	4.1E4	3.4E4	4.2E4
14	3.3E4 ±1E3	3.7E4 ±1E3	3.6E4 ±4E3	3.6E4 ±7E2	3.4E4 ±9E2	4.1E4	3.5E4	3.2E4	4.6E4	3.5E4	3.1E4	3.4E4
15	0.141 ±6E-3	0.187 ±2E-3	0.151 ±4E-3	0.158 ±4E-3	0.141 ±5E-3	0.204	0.198	0.176	0.208	0.212	0.165	0.156
16	0.088 ±0.10	<b>0.380</b> ±0.10	<b>0.411</b> ±0.69	<b>1.177</b> ±1.85	0.076 ±0.08	<b>4.493</b>	<b>0.348</b>	<b>0.205</b>	<b>5.316</b>	<b>4.806</b>	<b>0.197</b>	<b>0.254</b>
17	7.080 ±0.38	7.360 ±0.30	11.05 ±5.42	8.618 ±0.71	13.80 ±5.66	<b>30.52</b>	8.965	6.870	<b>30.71</b>	<b>24.31</b>	6.377	<b>14.52</b>
18	0.007 ±1E-4	0.009 ±8E-5	0.007 ±6E-5	<b>0.021</b> ±8E-3	<b>0.021</b> ±9E-4	<b>0.027</b>	0.009	0.008	<b>0.027</b>	<b>0.027</b>	0.008	<b>0.040</b>
19	3.202 ±0.02	3.412 ±0.03	3.208 ±0.01	3.372 ±0.02	3.232 ±0.03	4.478	3.424	3.216	4.558	4.375	3.266	4.128

corresponding mean value. The first 50% of each dataset constitute the training set and the rest of the data constitute the testing set.

In the comparison, we tested previously compared algorithms, mGMT, uGRT, mGRT, oGRT, oGMT, RT, and M5 and some state-of-the-art:

- Linear (Ridge) Regression (LR);
- SVM SMOreg (SMO) – algorithm that implements the support vector machine for regression [42];
- AdditiveRegression (AR) – meta predictor that enhances the performance of a regression base classifier;
- Bagging (BG) – ensembles of regression trees;
- MultilayerPerceptron (NN) – a feedforward artificial neural network with backpropagation.

All algorithms for all datasets were performed with the default set of parameters.

Comparison results of the prediction errors (RMSE) for the tested systems are shown in Table 6. As the training and testing sets were specified for all datasets, the standard deviation is shown only for

evolutionary tree inducers. It can be observed that mGMT is the most stable one, with no occasional catastrophic failures (bolded results), which is in contrast to all other algorithms. In most of the tested datasets, the proposed solution managed to achieve the smallest (or almost the smallest) prediction error. The results in Table 6 also show that the prediction performance between mGMT and uGMT is similar for many datasets. However, for some datasets, for example, *cal housing* (6) and *mv* (16), to reduce RMSE the mGMT system induced trees with more oblique splits and model leaves like oGMT. For other datasets, for example, *pol* (17), mGMT induced trees similar to uGRT. The benefits of allowing the tree (and node) representation to self-adapt to the currently analyzed data can be seen for other tested datasets.

A statistical analysis performed using the Friedman test showed that there are significant statistical differences between the algorithms (significance level equal to 0.05, *P* value < 0.0001, *F*-statistic = 110.7). Table 7 shows the results of a Dunn's multiple comparison test between mGMT and the tested algorithms based on the difference in rank sum. We can observe that mGMT managed to significantly outperform 7 of 11 tested solutions. We believe this

**Table 7**  
Results of Dunn's multiple comparison test between mGMT and the rest of the systems.

Algorithm	uGRT	uGMT	oGRT	oGMT	LR	RT	M5	SMO	AR	BG	NN
Significant?	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	No	Yes
<i>P</i> value	<0.0001	–	<0.0001	–	<0.0001	<0.0001	–	<0.0001	<0.0001	–	<0.001

is a good result, especially considering that the Dunn’s test is the most conservative option (less likely to find a significant difference) among all multiple comparison procedures [9,43].

**5. Conclusion and future works**

This article focuses on different representations of the decision tree applied to regression problems. We have validated five evolutionary tree inducers with different tree representations and showed their advantages and disadvantages. We also proposed a unified framework for the global induction of mixed trees that may have both univariate and oblique tests in internal nodes and regression planes and models in leaves. Embedding representation modification in different variants of the mutation operators allowed switching the node representation during the evolutionary tree induction. Modification of the BIC fitness function allowed minimization of the tree error and the tree complexity, including the tree size, number of attributes in the internal nodes, and the complexity of regression models in the leaves.

Experimental validation shows that the tree representation plays an important role in the final prediction model. The proposed solution not only extends the search space of possible solutions for the tree inducers with limited tree representations like uGRT but also significantly improves the speed of convergence algorithms with advanced representations like oGMT. In contrast to the rest of the homogeneous algorithms, the mGMT solution is capable of self-adapting to the problem that is being solved and choosing the most suitable representation for a particular tree node. The overall performance of mGMT in comparison to other state-of-the-art

solutions showed that its prediction performance is highly competitive.

We see many promising directions for future research. In particular, we are currently working on improving the algorithm efficiency, especially for large datasets, with OpenMP+MPI and GPGPU parallelization. We are also considering including the non-linear regression models in the leaves and extending multi-objective optimization to the Pareto dominance approach. Additional adaptive schemes that might cover, for example, self-adaptive parameters or variants of genetic operators are also considered.

**Acknowledgments**

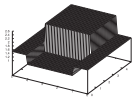
This project was funded by The National Science Center allocated on the basis of a decision 2013/09/N/ST6/04083. The second author was supported by the grant S/W1/2/13 from Bialystok University of Technology funded by Ministry of Science and Higher Education.

**Appendix.**

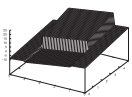
The appendix includes examples of different variants of two artificial datasets: *armchair* and *noisy*.

*Armchair dataset*

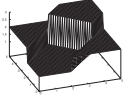
The *armchair* dataset has uniform distribution, and its variants are as follows (Figs. 15–20):

$$y(x_1, x_2) = \begin{cases} 1.5, & x_2 < 1 \\ 1.5, & x_2 \geq 4 \\ 2.5, & x_1 \geq 2; 1 \leq x_2 < 4 \\ 1, & x_1 < 2; 1 \leq x_2 < 4 \end{cases}$$



**Fig. 15.** Armchair Univariate Regression (AUR) dataset.

$$y(x_1, x_2) = \begin{cases} 2 * x_1 - 1.5 * x_2 + 2, & x_2 < 1 \\ 1.5 * x_1 - 1.5 * x_2 - 1, & x_2 \geq 4 \\ 3.5 * x_1 - 2.5 * x_2 + 10, & x_1 \geq 2; 1 \leq x_2 < 4 \\ 0.5 * x_1 - 2.5 * x_2 + 1.5, & x_1 < 2; 1 \leq x_2 < 4 \end{cases}$$


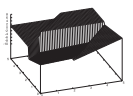
**Fig. 16.** Armchair Univariate Model (AUM) dataset.

$$y(x_1, x_2) = \begin{cases} 2, & x_2 - x_1 > 2 \\ 2, & x_2 - x_1 \leq -2 \\ 3, & x_2 + x_1 > 6; -2 < x_2 - x_1 \leq 2 \\ 1, & x_2 + x_1 \leq 6; -2 < x_2 - x_1 \leq 2 \end{cases}$$


**Fig. 17.** Armchair Oblique Regression (AOR) dataset.

$$y(x_1, x_2) = \begin{cases} 4 * x_1 - 1.5 * x_2 + 2, & x_2 - x_1 > 2 \\ 3 * x_1 - 1.5 * x_2 - 1, & x_2 - x_1 \leq -2 \\ 3.5 * x_1 + 1.5 * x_2 - 3, & x_2 + x_1 > 6; -2 < x_2 - x_1 \leq 2 \\ 0.5 * x_1 - 2.5 * x_2 + 1.5, & x_2 + x_1 \leq 6; -2 < x_2 - x_1 \leq 2 \end{cases}$$


**Fig. 18.** Armchair Oblique Model (AOM) dataset.

$$y(x_1, x_2) = \begin{cases} -2, & x_2 > 4 \\ 3 * x_1 - 1.5 * x_2 - 1, & x_2 \leq 1 \\ 3, & x_1 + x_2 > 5.5; 1 < x_2 \leq 4 \\ 0.5 * x_1 - 2.5 * x_2 + 1.5, & x_1 + x_2 \leq 5.5; 1 < x_2 \leq 4 \end{cases}$$


**Fig. 19.** Armchair Mix 1 – all 4 types of nodes (AMix1) dataset.



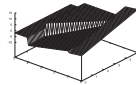
$$y(x_1, x_2) = \begin{cases} 1, & x_2 - x_1 > 2 \\ 3 * x_1 - 1.5 * x_2 - 8, & x_2 - x_1 \leq -2 \\ 4, & x_2 > 3; -2 < x_2 - x_1 \leq 2 \\ 0.5 * x_1 - 2.5 * x_2 - 1.5, & x_2 \leq 3; -2 < x_2 - x_1 \leq 2 \end{cases}$$


Fig. 20. Armchair Mix 2 – all 4 types of nodes (AMix2) dataset.

Noisy dataset

The noisy dataset has three different data distributions (one for each region):

- normal distribution centered at  $(x_1 = 2, x_2 = 0)$ ;

- random distribution centered at  $(x_1 = 0.5, x_2 = 1.5)$ ;
- chi squared distribution centered at  $(x_1 = 3, x_2 = 3)$ .

Noise for each set equals  $\pm 10\%$  of the  $y(x_1, x_2)$  value. Datasets variants are as follows (Figs. 21–26):

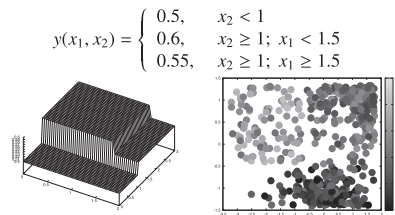


Fig. 21. Noisy Univariate Regression (NUR) dataset.

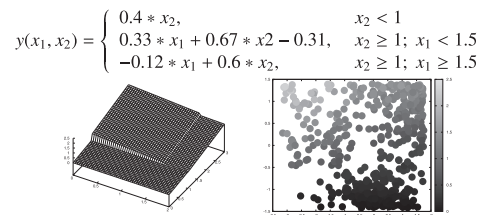


Fig. 22. Noisy Univariate Model (NUM) dataset.

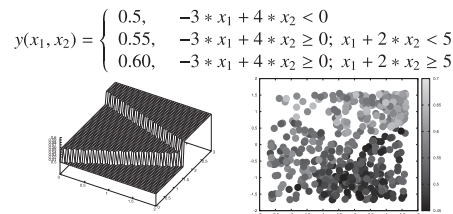


Fig. 23. Noisy Oblique RegressionI (NOR) dataset.

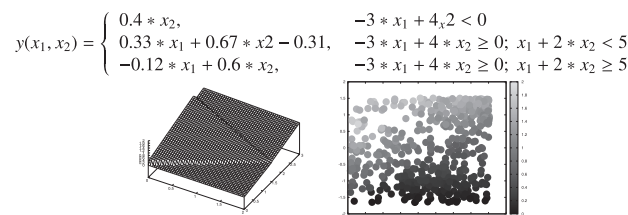


Fig. 24. Noisy Oblique Model (NOM) dataset.

$$y(x_1, x_2) = \begin{cases} 0.75, & x_2 < 1 \\ 0.33 * x_1 + 0.67 * x_2 - 0.31, & x_2 \geq 1; x_1 + 2 * x_2 < 5 \\ -0.12 * x_1 + 0.6 * x_2, & x_2 \geq 1; x_1 + 2 * x_2 \geq 5 \end{cases}$$

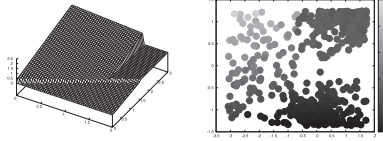


Fig. 25. Noisy Mix 1 – all 4 types of nodes (NMix1) dataset.

$$y(x_1, x_2) = \begin{cases} 0.4 * x_2, & -3 * x_1 + 4 * x_2 < 0 \\ 0.33 * x_1 + 0.67 * x_2 - 0.31, & -3 * x_1 + 4 * x_2 \geq 0; x_1 < 1.5 \\ 0.75, & -3 * x_1 + 4 * x_2 \geq 0; x_1 \geq 1.5 \end{cases}$$

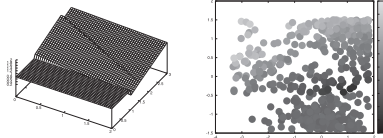


Fig. 26. Noisy Mix 2 – all 4 types of nodes (NMix2) dataset.

## References

- [1] R.C. Barros, D.D. Ruiz, M. Basgalupp, Evolutionary model trees for handling continuous classes in machine learning, *Inf. Sci.* 181 (2011) 954–971.
- [2] R.C. Barros, M.P. Basgalupp, A.C. Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, *IEEE Trans. SMC Part C* 42 (3) (2012) 291–312.
- [3] R.C. Barros, P.A. Jaskowiak, R. Cerri, A. Carvalho, A framework for bottom-up induction of oblique decision trees, *Neurocomputing* 135 (2014) 3–12.
- [4] R.C. Barros, A.C. Carvalho, A.A. Freitas, *Automatic Design of Decision-Tree Induction Algorithms*, Springer, 2015.
- [5] C. Blake, E. Keogh, C. Merz, *UCI Repository of Machine Learning Databases*, 1998 <http://www.ics.uci.edu/mllearn/MLRepository.html>.
- [6] U. Boryczka, J. Kozak, Enhancing the effectiveness of Ant Colony Decision Tree algorithms by co-learning, *Appl. Soft Comput.* 30 (2015) 166–178.
- [7] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees*, Wadsworth Int. Group, 1984.
- [8] C.E. Brodley, P.E. Utgoff, Multivariate decision trees, *Mach. Learn.* 19 (1) (1995) 45–77.
- [9] H.J. Cabral, Multiple comparisons procedures, *Circulation* 117 (5) (2008) 698–701.
- [10] M. Czajkowski, M. Kretowski, Global induction of oblique model trees: an evolutionary approach, in: *Proc. of ICAISC'13. LNAI 7895*, 2013, pp. 1–11.
- [11] M. Czajkowski, M. Kretowski, An evolutionary algorithm for global induction of regression and model trees, *Int. J. Data Mining Modell. Manage.* 5 (3) (2013) 261–276.
- [12] M. Czajkowski, M. Kretowski, Evolutionary induction of global model trees with specialized operators and memetic extensions, *Inf. Sci.* 288 (2014) 153–173.
- [13] J. Demsar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.* 7 (2006) 1–30.
- [14] A.E. Eiben, S.K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, *Swarm Evolut. Comput.* 1 (1) (2011) 19–31.
- [15] F. Esposito, D. Malerba, G. Semeraro, A comparative analysis of methods for pruning decision trees, *IEEE Trans. Pattern Anal. Mach. Intell.* 19 (5) (1997) 476–491.
- [16] A. Fakhari, A.M.E. Moghadam, Combination of classification and regression in decision tree for multi-labeling image annotation and retrieval, *Appl. Soft Comput.* 13 (2) (2013) 1292–1302.
- [17] G. Fan, J.B. Gray, Regression tree analysis using TARGET, *J. Comput. Graph. Stat.* 14 (1) (2005) 206–218.
- [18] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996.
- [19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, *SIGKDD Explor.* 11 (1) (2009).
- [20] H. He, *Self-Adaptive Systems for Machine Intelligence*, Wiley, 2011.
- [21] A. Hazan, R. Ramirez, E. Maestre, A. Perez, A. Pertusa, Modelling expressive performance: a regression tree approach based on strongly typed genetic programming, *Appl. Evolut. Comput. LNCS 3907* (2006) 676–687.
- [22] T. Kenesei, J. Abonyi, Hinging hyperplane based regression tree identified by fuzzy clustering and its application, *Appl. Soft Comput.* 13 (2) (2013) 782–792.
- [23] S.B. Kotsiantis, Decision trees: a recent overview, *Artif. Intell. Rev.* 39 (2013) 261–283.
- [24] M. Koziol, M. Wozniak, Multivariate decision trees vs. univariate ones, *Adv. Intell. Soft Comput.* 57 (2009) 275–284.
- [25] M. Kretowski, M. Grześ, Evolutionary induction of mixed decision trees, *Int. J. Data Warehous. Mining* 3 (4) (2007) 68–82.
- [26] A.R. Lima, A.J. Cannon, W.W. Hsieh, Nonlinear regression in environmental sciences by support vector machines combined with evolutionary strategy, *Comput. Geosci.* 50 (2013) 136–144.
- [27] J. Liu, C. Sui, D. Deng, J. Wang, B. Feng, W. Liu, C. Wu, Representing conditional preference by boosted regression trees for recommendation, *Inf. Sci.* (2015), <http://dx.doi.org/10.1016/j.ins.2015.08.001>.
- [28] X. Llorca, S. Wilson, Mixed decision trees: minimizing knowledge representation bias in LCS, in: *Proc. of GECCO'04, LNCS 3103*, 2004, pp. 797–809.
- [29] W. Loh, Fifty years of classification and regression trees, *Int. Stat. Rev.* 83 (3) (2014) 329–348.
- [30] D. Malerba, F. Esposito, M. Ceci, A. Appice, Top-down induction of model trees with regression and splitting nodes, *IEEE Trans. PAMI* 26 (5) (2004) 612–625.
- [31] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer, 1996.
- [32] H. Moriguchi, S. Honiden, CMA-TWEANN: efficient optimization of neural networks via self-adaptation and seamless augmentation, in: *Proc of GECCO'12*, 2012, pp. 903–910.
- [33] F.M. Ortuno, O. Valenzuela, et al., Comparing different machine learning and mathematical regression models to evaluate multiple sequence alignments, *Neurocomputing* 164 (2015) 123–136.
- [34] F.E.B. Otero, A.A. Freitas, C.G. Johnson, Inducing decision trees with an ant colony optimization algorithm, *Appl. Soft Comput.* 12 (11) (2012) 3615–3626.
- [35] G. Potgieter, A. Engelbrecht, Genetic algorithms for the structural optimisation of learned polynomial expressions, *Appl. Math. Comput.* 186 (2) (2007) 1441–1466.
- [36] G. Potgieter, A. Engelbrecht, Evolving model trees for mining data sets with continuous-valued classes, *Expert Syst. Appl.* 35 (2008) 1513–1532.
- [37] L. Rokach, O.Z. Maimon, Top-down induction of decision trees classifiers – a survey, *IEEE Trans. SMC, Part C* 35 (4) (2005) 476–487.
- [38] L. Rokach, O.Z. Maimon, Data mining with decision trees: theory and application, *Mach. Percept. Artif. Intell.* 69 (2008).
- [39] G. Schwarz, Estimating the dimension of a model, *Ann. Stat.* 6 (1978) 461–464.
- [40] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, Improvements to the SMO algorithm for SVM regression, *IEEE Trans. Neural Netw.* 11 (5) (2000) 1188–1193.
- [41] D.J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th ed., Chapman and Hall/CRC, 2007.
- [42] N.S. Sheth, A.R. Deshpande, A review of splitting criteria for decision tree induction, *Fuzzy Syst.* 7 (1) (2015) 1–4.
- [43] L. Torgo, *Regression DataSets Repository*, available at <http://www.dcc.fc.up.pt/ltorgo/Regression/DataSets.html> (accessed 16.02.16).
- [44] J.R. Quinlan, Learning with continuous classes, in: *Proc. of AI'92, World Scientific*, 1992, pp. 343–348.