Mixed Decision Trees: An Evolutionary Approach

Marek Kretowski and Marek Grześ

Faculty of Computer Science, Białystok Technical University
Wiejska 45a, 15-351 Białystok, Poland
{mkret, marekg}@ii.pb.bialystok.pl

Abstract. In the paper, a new evolutionary algorithm (EA) for mixed tree learning is proposed. In non-terminal nodes of a mixed decision tree different types of tests can be placed, ranging from a typical univariate inequality test up to a multivariate test based on a splitting hyperplane. In contrast to classical top-down methods, our system searches for an optimal tree in a global manner, i.e. it learns a tree structure and tests in one run of the EA. Specialized genetic operators allow for generating new sub-trees, pruning existing ones as well as changing the node type and the tests. The proposed approach was experimentally verified on both artificial and real-life data and preliminary results are promising.

1 Introduction

Decision trees [18] are one of the most frequently applied data mining approaches. There exist many induction algorithms which can differ in several more or less important elements, like for example the way for tree construction (i.e. top-down versus global) or the way for test selection. From a users point of view, one of the most important features of a decision tree is a test representation in the internal nodes. In typical univariate trees two types of tests are usually permitted. For a nominal attribute, the mutually exclusive sets of feature values are associated with each branch, whereas for a continuous valued feature inequality tests are applied. In case of multivariate trees more than one feature can be used to create a test. Linear (oblique) tests based on a splitting hyper-plane are specific and the most widely used form of the multivariate test. It should be noticed that most of the DT-based systems are homogeneous, which means that they take advantage of only one type of test (i.e. univariate or oblique).

The term *mixed decision trees* was proposed by Llora and Wilson in [15] to describe trees in which different types of tests can be exploited. One of the first and best-known examples of such an approach is the *CART* system [3]. This system is able to search for a linear combination of non-nominal features in each node and it compares the obtained test with the best univariate test. However, it should be noted that *CART* has a strong preference to simpler tests and it results in very rare use of more elaborate splits. Another form of the hybrid classifier is proposed by Brodley in [4]. Her *MCS* system combines univariate tests, linear machines and instance-based classifiers (*k-NN*) and during the top-down generation of a tree classifier it recursively applies automatic bias selection. Recently, a fine grain parallel model *GALE* [15] was applied to generate decision trees which employ inequality and oblique tests.

Evolutionary techniques [16] are known to be useful in many data mining tasks [9]. They were successfully applied to learning univariate (e.g. [10,19,20]) and linear trees (e.g. [6,2,5]). Regardless of the tree types there are two main approaches to the induction: top-down and global. The first one is based on a greedy recursive procedure of test searching and sub-node creation until a stopping condition is met. In contrast to this classical method, the global algorithm searches for both the tree structure and tests at the same time.

The global approach based on evolutionary algorithms for decision tree induction was investigated in our previous papers. We showed that homogeneous trees (univariate [12] or oblique [13,14]) can be effectively induced and we demonstrated that globally generated classifiers are generally less complex with at least comparable accuracy. In this paper, we want to merge the two developed methods in one system, which will be able to induce mixed trees.

The rest of the paper is organized as follows. In the next section our global system for induction of mixed decision trees is presented. Preliminary experimental validation of the approach on both artificial and real-life datasets are presented in section 3. The paper is concluded in the last section.

2 Global Induction of Mixed Decision Trees

The general structure of the proposed algorithm follows a typical evolutionary framework [16]. As the presented approach is a continuation and unification of our work on the global induction of homogeneous decision trees [12,13,14], in this section we described only these issues that are specific to mixed trees.

Representation and initialization. A mixed decision tree is a complicated tree structure, in which the number of nodes, test types and even the number of test outcomes are not known in advance for a given learning set. Moreover additional information, e.g. about feature vectors associated with each node, should be accessible during the induction. As a result, decision trees are not specially encoded in individuals and they are represented in their actual form.

There are three possible test types in internal nodes: two univariate and one multivariate. In case of univariate tests, a test representation depends on the considered attribute type. 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. For continuous-valued features typical inequality tests with two outcomes are used. In order to speed up the search process only boundary thresholds[1] as potential splits are considered and they are calculated

[1] A boundary threshold for the given attribute is defined as a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of the attribute, in which the examples belong to two different classes.

before starting the EA. Finally, an oblique test with binary outcome can be also applied as a multivariate test. A splitting hyperplane is represented by a fixed-size table of real values corresponding to the weight vector and the threshold. The inner product is calculated to decide where an example is routed.

Before starting the actual evolution, the initial population is created. All initial trees are homogeneous, but half of the population is initialized with univariate tests and the other part with oblique tests. A simple top-down algorithm is applied to generate all individuals. In each potential internal node it chooses randomly a pair of objects from different classes and searches for a test which separates them to distinct sub-trees. In case of a univariate tree, such a test can be directly constructed for any feature with different feature values. When an oblique test is necessary, the splitting hyperplane is perpendicular to the segment connecting the two drawn objects and placed in a halfway position.

The algorithm terminates when the fitness of the best individual does not improve during a fixed number of generations (default value is equal 1000) or the maximum number of generations (default value: 10000) is reached.

Genetic operators. There are two specialized genetic operators corresponding to the classical mutation and cross-over. Application of both operators can result in changes of the tree structure and tests in non-terminal nodes.

A mutation-like operator is applied with a given probability to a tree (default value is 0.5) and it guarantees that at least one node of the selected individual is mutated [14]. Modifications performed by this operator depend on the node type (i.e. if the considered node is a leaf node or an internal node). For a non-terminal node a few possibilities exist:

- a completely new test of the same or different type can be drawn; new tests are created in the same way as described for the initialization,
- the existing test can be altered by shifting the splitting threshold (continuous-valued feature), by re-grouping feature values (nominal features) or by shifting the hyperplane (oblique test); these modifications can be purely random or can be performed according to the adapted dipolar operator [11],
- the test can be replaced by another test or tests can be interchanged,
- one sub-tree can be replaced by another sub-tree from the same node,
- the node can be transformed into a leaf.

Modifying a leaf makes sense only if it contains objects from different classes. The leaf is transformed into an internal node and a new test is randomly chosen. The search for effective tests can be recursively repeated for all descendants.

There are also several variants of cross-over operators (applied with a default probability 0.2). One node is randomly chosen in each of two affected individuals and an exchange encompasses sub-trees or is limited only to nodes (their tests). The order of sub-trees can be also altered during the cross-over.

The application of any genetic operator can result in a necessity for relocation of the input vectors between parts of the tree rooted in the modified node. Additionally the local maximization of the fitness is performed by pruning lower parts of the sub-tree on the condition that it improves the value of the fitness.

Fitness function. A fitness function drives the evolutionary search process and is the most important and sensitive component of the algorithm. When concerning a classification task it is well-known that the direct optimization of the classifier accuracy measured on the learning set leads to an over-fitting problem. In a typical top-down induction of decision trees, the over-specialization problem is mitigated by defining a stopping condition and by applying a post-pruning [8]. In our approach, the search for an optimal structure is embedded into the evolutionary algorithm by incorporating a complexity term in the fitness function. The fitness function is maximized and has the following form:

$$Fitness(T) = Q_{Reclass}(T) - \alpha \cdot (Comp(T) - 1.0), \tag{1}$$

where $Q_{Reclass}(T)$ is a reclassification quality and α is the relative importance of the classifier complexity (default value is 0.005). In the simplest form the tree complexity $Comp(T)$ can be defined as the classifier size which is usually equal to the number of nodes. The penalty associated with the classifier complexity increases proportionally with the tree size and prevents classifier over-specialization. Subtracting 1.0 eliminates the penalty when the tree is composed of only one leaf (in majority voting).

This simple complexity definition is surely adequate for a homogeneous tree composed of only univariate tests. However, when linear tests are also considered, it seems that a more elaborate solution is necessary. It is rather straightforward that an oblique split based on a few features is more complex than a univariate test and that we should apply preference to simpler tests as an inductive bias. As a consequence the tree complexity should also reflect the complexity of the tests. However it is not easy to definitely decide how to balance different test complexities because it depends on the problem solved and user preferences. In such a situation we decided to define the tree complexity $Comp(T)$ in a flexible way and allow the user to tune its final form:

$$Comp(T) = |N_{leaf}(T)| + \sum_{n \in N_{int}(T)} (1 + \beta \cdot (F(n) - 1)), \tag{2}$$

where $N_{leaf}(T)$ and $N_{int}(T)$ are sets of leaves and internal nodes correspondingly, $F(n)$ is the number of features used in the test associated with the node n and $\beta \in [0, 1]$ is the relative importance of the test complexity (default value 0.2). The complexity of the tree is defined as a sum of the complexities of the nodes and it is assumed that for leaves and internal nodes with univariate tests the node complexity is always equal to 1.0. It can be also observed that when $\beta = 1$ the number of features used in a test is applied as the test complexity, whereas when $\beta = 0$ the complexity of a test is completely ignored.

3 Experimental Results

The proposed approach to learning mixed decision trees is assessed on both artificial and real life datasets and is compared to the well-known top-down

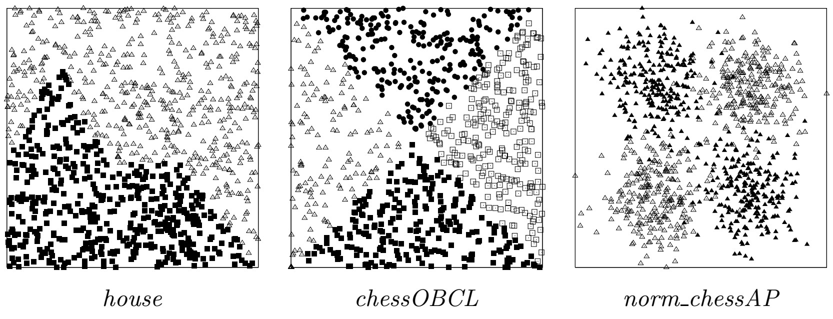

Fig. 1. Examples of artificial datasets

univariate ($C4.5$ [21]) and oblique ($OC1$ [17]) decision tree systems. It is also compared to two homogenous versions of our global *GDT* system: univariate - *GDT-AP* [12] and oblique *GDT-OB* [14]. All prepared artificial datasets comprise training and testing parts. In case of data from a UCI repository [1] for which testing data is not provided, 10-fold stratified cross-validation was employed. Each experiment on all stochastic algorithms (i.e. all except $C4.5$) was performed 10 times and the average result of such an evaluation was presented. The $OC1$ system was run with different values of the seed that initializes the random number generator. Our system is initialized by the system time.

A statistical analysis of the obtained results was done by the Friedman test with the corresponding Dunn's multiple comparison test (significance level equal to 0.05) as recommended by Demsar [7].

Artificial datasets. A range of artificial datasets suited to axis-parallel or oblique tests was generated to assess the universality of the proposed approach. Most of the datasets have two continuous-valued features (see examples in Fig. 1) and only the $LS10$ (Linearly Separable) dataset has 10 features. The number of examples was varied and depends on the number of distinct regions. In the training part it ranged from 1000 (for simple 2-dimensional problems) to 4000 (for $LS10$). The testing part is twofold larger in each case.

There was also prepared a special dataset (see Fig. 2a) to validate the performance of the proposed mixed decision tree system. This dataset is the three class problem that contains three descriptive features. Two of them (x and y) are continuous-valued and the last one (z) is nominal with two binary values. This experiment was intended to check whether our system can deal with such a problem in which the best separability of classes can be achieved by incorporating all three types of splits. A hyperplane and an inequality test separate observations on two planes and one nominal test provides additional separation between those planes. In Fig. 2b the decision tree learned by the *GDT-Mix* system is presented. It is the best solution to this problem. The most important thing is that the algorithm was able to select correctly different types of tests and apply them to build the optimal tree structure. This experiment shows

Table 1. Results on artificial data

	C4.5		OC1		GDT-Mix		GDT-AP		GDT-OB	
Dataset	**size**	**quality**	**size**	**quality**	**size**	**quality**	**size**	**quality**	**size**	**quality**
chess2x2	1	50	10.1	89.3	4.0	99.7	4	99.75	4	99.34
chess2x2x2	1	50	23.8	71.0	8.0	98.5	8	99.72	8.2	97.00
chess3x3	9	99.7	21.1	73.7	9.0	98.8	9	99.73	9.9	97.08
chessOB2CL	33	95.6	7	77.3	4.3	98.0	17.9	92.64	4.7	99.05
chessOB4CL	35	94.6	4.3	49.8	4.0	97.9	18	92.14	4.4	98.41
house	21	97.4	8.2	92.8	4.0	96.0	13.3	96.62	4	96.71
ls10	284	77.3	7.3	95.3	2.0	95.7	18.8	70.68	2	97.20
ls2	22	97	2	99.7	2.0	99.8	14	95.68	2	99.93
normal	5	90	7.3	87.9	3.6	89.5	25.7	86.85	4	90.01
norm_chessAP	1	50	11.2	85.5	4.0	95.4	4.2	95.53	4	95.42
norm_chessOB	19	93	11	83.3	4.0	93.7	9.3	92.59	4	93.58
norm_wave	15	94	8.4	90.3	4.0	94.5	9.1	93.45	4	94.87
zebra1	25	95.3	3	83.5	3.3	99.1	15.3	94.64	3	99.28
zebra2	2	59.5	4.8	94.1	4.0	98.5	21.4	91.63	4.4	98.70
zebra3	57	91.2	8.2	24.3	8.8	95.4	31.5	88.80	8.8	96.76

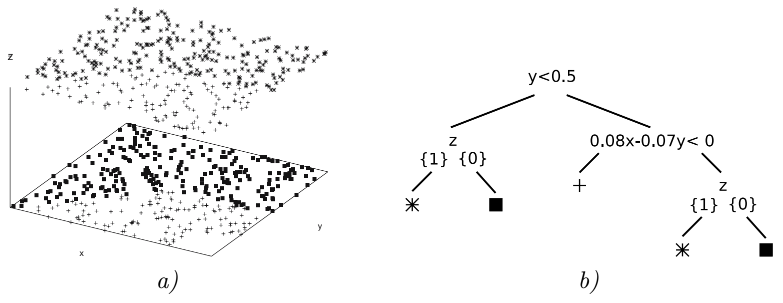

Fig. 2. A graphical representation of the dataset which can be optimally separated only with all three test types and the tree obtained by *GDT-Mix* system

that when such compound relationships will exist in the real data our algorithm may tackle them successfully revealing invaluable information for specialists in a certain domain to which it might have been applied.

The results on the range of datasets designed for this investigation are collected in Table 1. Because we analyze artificial data in this experiment, we know how, in terms of the type of tests used in the tree, the optimal solution can be represented. There are certain classification tasks, like for instance the classical chessboard problem, that suit very well univariate decision trees. There are also linearly separable datasets (like e.g. *LS*10) for which splits based on hyperplanes are highly recommended to avoid a staircase-like structure. The main aim of our endeavor in this work is to show that *GDT-Mix* can easily adjust to the specific problem. The analysis of Table 1 proved that the *GDT-Mix* inducer performs better on axis parallel data while compared to oblique systems

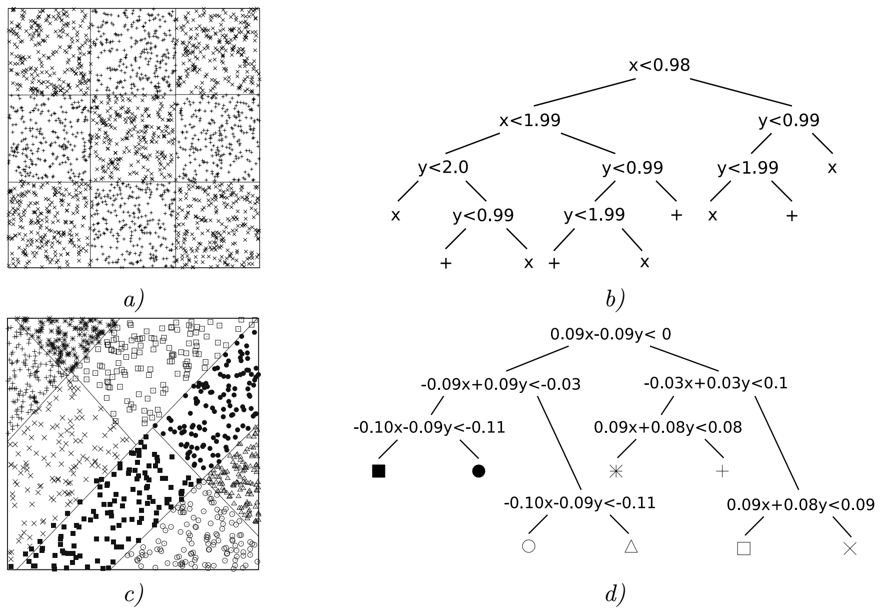

Fig. 3. Decision trees obtained by *GDT-Mix* system for *chess3x3* and *zebra3* datasets (*b)* and *d)*) and the corresponding dataset scatterplots with drawn splits (*a)* and *c)*)

and on linearly separable data while compared to axis parallel systems. It is also important that statistical analysis does not show significant differences between the *GDT-Mix* algorithm and the systems specialized for certain problems when a comparison is made on such problems. Our universal system performs as well as the specialized systems, that is its very strong point. The statistical test indicates that *GDT-OB* is significantly better in terms of quality than *C4.5*, *OC1* and *GDT-AP*. As for *GDT-Mix*, it is statistically better than *OC1*. The comparisons based on the second measure (the tree size) are also favorable. Statistical analysis reveals that *GDT-Mix* produces significantly smaller trees than *C4.5* and *GDT-AP*. This score is easily justified, because in the case of problems which require oblique splits our *GDT-Mix* system takes advantage of such splits.

Discussed results show that the proposed algorithm is more flexible in terms of representation which can be modified during the induction. For that reason more detailed analysis of these results aims at investigating the obtained decision trees. Such trees for relatively complex axis parallel and linearly separable classification tasks are presented in Fig. 3. These trees present a promising result because the *GDT-Mix* system managed to find the type of tests that suit the data in the best way and was able to apply them to build trees that perform very competitively while comparing to results of specialized systems. In Fig. 3a and 3c splits from decision trees are additionally drawn to present how the input space is partitioned by the global inducer. These splits show that trees obtained

Table 2. Results on real datasets with only continuous attributes

	OC1		**GDT-Mix**		**GDT-OB**	
Dataset	**size**	**quality**	**size**	**quality**	**size**	**quality**
balance-scale	5.4	90.0	2.8	89.3	3.2	89.1
bcw	4.7	91.2	2.0	97.1	2.0	96.9
bupa	5.8	65.6	3.6	69.5	3.0	71.3
glass	4.5	55.7	13.4	69.9	11.6	68.8
page-blocks	15.6	96.6	3.0	94.9	3.0	95.3
pima	6.5	69.6	2.4	75.0	2.1	75.3
sat	58.3	78.9	6.0	83.0	7.0	83.1
vehicle	21.6	66.4	8.8	67.7	7.7	65.7
waveform	10.5	77.4	4.2	81.2	4.2	82.2
wine	3.2	87.0	4.2	89.3	4.8	90.9

Table 3. Results on real datasets with both continuous and nominal attributes

	C4.5		**GDT-Mix**		**GDT-AP**	
Dataset	**size**	**quality**	**size**	**quality**	**size**	**quality**
australian	39	87	2.0	86.5	22.8	84.6
cars	31	97.7	3.6	97.9	4.0	98.7
cmc	136.8	52.2	4.0	55.1	13.1	53.8
german	77	73.3	3.8	72.0	16.5	73.4
golf	5	60	4.9	70.5	4.7	72.5
heart	22	77.1	6.1	75.3	44.9	74.2
solar	20	73.1	5.8	70.7	33.7	73.6
vote	5	97	2.0	97.0	13.5	95.6

in this experiment have an optimal structure (the empirical superiority of global induction).

Real-life data. Results on real-life data are divided into two groups. In the first one, $GDT - Mix$ is compared with $OC1$ and $GDT - OB$ systems, which are designed for applications where the instances have only numeric (continuous) feature values (Table 2). In the second group, the proposed system is compared with univariate tree induction algorithms on datasets that have both nominal and continuous attributes (Table 3).

The analysis of these results shows that there are no statistically significant differences in the quality between compared algorithms on all datasets. It is very favorable result. It means that the *GDT-Mix* system, which is designed to be universal to different kinds of tasks, performs as good as specialized counterparts. In the case of the size of the tree the same statistical analysis of the Table 3 indicates that *GDT-Mix* produces significantly smaller trees than *C4.5*. A detailed inspection of this table shows that there are some datasets (e.g. *heart*) for which there is evident difference in the tree size which shows the superiority

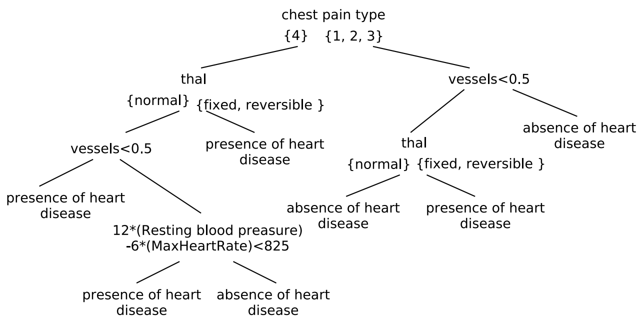

Fig. 4. The decision tree for *heart* data found by *GDT-Mix*

of the *GDT-Mix* algorithm. This is a very useful feature of mixed trees. As for the Table 2 there is a statistical difference in terms of the tree size ($p = 0.046$) but Dunn's test failed to detect it.

More detailed analysis of decision trees obtained for real-life *heart* data is presented in Figure 4. This dataset was chosen for investigation because it contains both nominal and continuous attributes and represents a quite easily understood problem (at least in terms of outcomes of the classifier). Figure 4 presents one of decision trees (there were 10 runs of the algorithm on each dataset) that were gained in our experiment for *heart* data. In the presented tree all three types of possible tests are used. This example underlines the advantage of decision trees of being self explanatory and easy to understand. In mixed decision trees we can have tests both on nominal and continuous attributes what present interesting features of this system in terms of practical applications.

4 Conclusion and Future Works

In the paper a new evolutionary algorithm for global induction of mixed decision trees is proposed. In the unified framework both univariate and oblique tests are searched and applied in not-terminal nodes for optimal data splitting. The flexible defined fitness function enables the controlling of the inductive biases. Even preliminary validation shows that the algorithm is able to adapt to the problem being solved and to locally choose the most suitable test representation.

The presented approach is still under development and currently we are working on introducing more specialized mutation variants. They will allow the system e.g. to switch from an oblique hyper-plane to the closest axis-parallel test and analogously to slightly incline an original univariate test. We also consider introducing additional test types, especially multivariate tests. Furthermore, the fitness function and especially the impact of the definition of the complexity term on the resulting decision tree will be studied in more detail.

Acknowledgments. This work was supported by the grant W/WI/5/05 from Białystok Technical University.

References

1. Blake, C., Keogh, E., Merz, C.: *UCI repository of machine learning databases*, Irvine, CA: University of California, Dept. of Computer Science (1998).
2. Bot, M., Langdon, W.: Application of genetic programming to induction of linear classification trees. In *EuroGP 2000*. Springer LNCS 1802 (2000) 247–258.
3. Breiman, L., Friedman, J., Olshen, R., Stone C.: *Classification and Regression Trees*. Wadsworth Int. Group (1984).
4. Brodley, C.: Recursive automatic bias selection for classifier construction, *Machine Learning* 20 (1995) 63-94.
5. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **7**(1) (2003) 54–68.
6. Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J.: Piecewise-linear classifiers using binary tree structure and genetic algorithm. *Pattern Recognition* **29**(11) (1996) 1905–1917.
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7 (2006) 1–30.
8. Esposito, F., Malerba, D., Semeraro, G.: A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(5) (1997) 476–491.
9. Freitas A.: *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer (2002).
10. Koza, J.: Concept formation and decision tree induction using genetic programming paradigm, In *Proc. of PPSN 1.*, Springer LNCS 496 (1991) 124–128.
11. Krętowski, M.: An evolutionary algorithm for oblique decision tree induction, In: *Proc. of ICAISC'04*, Springer LNCS 3070 (2004) 432–437.
12. Krętowski, M., Grześ, M.: Global learning of decision trees by an evolutionary algorithm, In: *Information Processing and Security Sys.*, Springer, (2005) 401–410.
13. Krętowski, M., Grześ, M.: Global induction of oblique decision trees: an evolutionary approach, In: *Proc. of IIPWM'05.*, Springer, (2005) 309–318.
14. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection, In: *Proc. of ICAISC'06*, Springer LNCS 4029 (2006).
15. Llora, X., Wilson, S.: Mixed decision trees: Minimizing knowledge representation bias in LCS, In: *Proc. of GECCO'04*, Springer LNCS 3103 (2004) 797–809.
16. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3^{rd} edn. Springer (1996).
17. Murthy, S., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2 (1994) 1–33.
18. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery* 2 (1998) 345–389.
19. Nikolaev, N., Slavov, V.: Inductive genetic programming with decision trees. *Intelligent Data Analysis* 2 (1998) 31–44.
20. Papagelis, A., Kalles, D.: Breeding decision trees using evolutionary techniques. In: *Proc. of ICML'01.*, Morgan Kaufmann (2001) 393–400.
21. Quinlan, J.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).