Hierarchical Classification Trees Using Type-Constrained Genetic Programming

Athanasios Tsakonas and George Dounias, *Member, IEEE*

***Abstract*—In this paper we investigate the capability of the genetic programming approach for producing hierarchical, rule-based, classification trees. These trees can be seen as an extension to the machine learning decision trees concept, where the predicates here can be complex expressions rather than just simple attribute-value comparisons. In order to improve the search ability and to produce meaningful results, type-constraints are applied to the genetic programming procedure, expressed in a BNF grammar. The model is tested in two well-known domains. In the Balance-Scale data, the system achieves in revealing the data creation rule. In the E-Coli Protein Localization Sites data, the system realizes a competitive to the literature classification score, retaining the solution comprehensibility. The training procedure is guided by an adaptive fitness measure. The overall performance of this system denotes its competitiveness to standard computational intelligent procedures.**

***Index Terms*—Classification Trees, Type-Constrained Genetic Programming, Balance-Scale, E-Coli Protein Localization Site**

I. Introduction

The classification procedures can be divided in two types, concerning the number of categories that are classified. The first classification type separates the data between only two classes (known as binary classification or two-class task), and the second type classifies the data between more than two classes (multi-class task). Today, there is a number of intelligent classification methods which handle efficiently the two-class task, such as the *AdaBoost* and the *support vector machines*. On the other hand, there are intelligent methodologies, such as the *inductive decision trees* or the *genetic algorithms*, which can handle efficiently both types of classification. In this paper, we are concerned, among the various classification approaches, with the rule-base production techniques for classification. Any multi-class problem can be substituted by, more than one, two-class problems. One such approach is to build independent classification rules for each of the classes and then run these -competitive- rules simultaneously [1]. However, it is not clear how the case of possible classification conflicts between some of the rules would be dealt with. Moreover, it is not clear, how the absence of a positive classification result, would be dealt with, among that kind of competitive crisp rule-bases. This problem is addressed in [2] where the construction of two-class (or two class-sets) rules is performed in a hierarchical way, creating a cooperative crisp rule-base -rather a competitive one-which always results in one class. A prime disadvantage, in both techniques, is that, each of the rules demands a separate run for the intelligent approach - in these cases, the genetic programming [3]. Thus, these methods can be case-driven and may involve heavy human interaction -and so, dubious results in different problems-, preventing us from characterizing them as generalizing approaches. The other way of handling multi-class tasks, in order to build a rule-base, is by using directly a multi-class approach. Among the methodologies of this type, inductive decision trees and genetic programming have been used with success in the past, although the complexity of the classification task often is increased when more than two classes are separated. These multi-class methods can be further divided in two types. The first type constructs fuzzy competitive rule-bases using, for example, heuristic [4] or genetic programming [5] techniques. The second type, constructs crisp cooperative and hierarchical classification rule-trees, such as Quinlan's inductive decision trees [6]. The popularity of these decision trees may be explained by the natural decision method that humans often follow. The latter conclusion is demonstrated for example, in the medical field, where physicians usually follow a form of a complex decision/classification tree [7], showing that, medical decision making often is similar to Quinlan's approach. Although fast and robust, this algorithm is however restricted in terms of each rule's (tree branch) premise set, where the expression evaluated is an inequality between an attribute (input variable) and a value (number). Apparently, a more generic methodology could involve, in the rules' premise sets, the incorporation of more complex comparisons, such as combinations of expressions including more than one attributes and values. The idea is addressed in [3,8] where the genetic programming approach is used to build a kind of decision trees. In general, the genetic programming search, found with a proper function set, has been proved capable in finding optimal solutions in a reasonable time for a variety of classification tasks [9]. Nevertheless, two problems arise, when dealing with the construction of hierarchical classification trees. First, as the number of classes addressed is usually more than two, and the search space is large, the methodology can be proved inefficient and slow. Second, the form of the solution is not derived necessarily in a comprehensive form. These problems are addressed in

A.Tsakonas is with the University of the Aegean, Dept. of Business Administration, 82100 Chios, Greece, (telephone 30-93-789-1399, e-mail: tsakonas@stt.aegean.gr)

G.Dounias is with the University of the Aegean, Dept. of Business Administration, 82100 Chios, Greece, (telephone 30-271-035165, e-mail: g.dounias@aegean.gr)

this paper by applying syntax constraints in the genetic programming trees, in order to improve the readability of the solutions and speed up the search by reducing the number of possible node combinations - thus the search space. This paper is organized as follows. In the next section, the design and implementation of the model is discussed. Section 3 contains the results from the application of the system in two databases, the Balance-Scale and E-Coli Protein Localization Sites. Finally, section 4 includes our conclusions and proposes further research in this domain.

II. Design and Implementation

Genetic Programming is an extension to the original concept of genetic algorithms where the existence of chromosomes is substituted by variable length, tree-like programs. The main genetic operators -such as *crossover* and *mutation*- are applied in genetic programming too. The hierarchical structure of a genetic programming candidate solution, known as *individual*, enables to each independent node to carry a functional code - thus, the solution can behave like a program. These nodes may represent a primitive operator -such as addition or multiplication- a number, a variable, etc. Most favorite genetic programming implementations are the *steady-state* ones, where only one population is preserved and, usually, a *tournament selection* between a small number of individuals replace the worst of these in every iteration. Although powerful in its definition, the genetic programming procedure may be proved greedy in computational and time resources. Therefore, when the syntax form of the desired solution is already known, it is useful to restrain the genetic programming from searching solutions with different syntax forms [10,11]. The most advantageous method to implement such restrictions, is to apply syntax constraints to genetic programming trees, usually with the help of a grammar, often declared in the *Backus-Naur-Form* (BNF) [12]. The BNF-grammar consists of *terminal* nodes and *non-terminal* nodes and is represented by the set {N,T,P,S} where *N* is the set of non-terminals, *T* is the set of terminals, *P* is the set of production rules and *S* is a member of *N* corresponding to the starting symbol. Here, the use of the terms terminal and non-terminal in a BNF-grammar, is not corresponding to what Koza defines as terminal and function. Rather, a function -a non-terminal node in terms of the GP tree architecture- is expressed as terminal in a BNF grammar. To avoid confusion, the use of the terms GPFunction and GPTerminal -instead of the ambiguous terms function and terminal- has been proposed [13] and is adapted throughout this paper. The construction of the production rules is the most critical point in the creation of a BNF grammar, since they express the permissible structures of an individual. In Table I, we show the production rules for the implementation of hierarchical rule-based classification trees into the genetic programming architecture. A legal tree according to this grammar is shown in Figure 1.

As seen by the grammar definition, a <clause> node may be either a <class> node or a <if> (<if_less>, <if_equal> etc.) node. If we examine more carefully this rule of the

TABLE I
BNF Grammar Production Rules for Hierarchical Classification Trees in Genetic Programming

Symbol	1. Quantity
<CLAUSE>	::= <CLASS>|<IF_LESS>| <IF_EQUAL>|....
<IF_LESS>	::=**IF_LESS**<EXPR> <EXPR> <CLAUSE><CLAUSE>
<EXPR>	::=<NUMB> | <OPER>
<OPER>	::=<PLUS>|<MINUS>|....
<PLUS>	::= **+** <EXPR> <EXPR>
<NUMB>	::=**1** | **2** | **3** ....
<CLASS>	::=**CLASS1**|**CLASS2**|**CLASS3**|...

The trees are described in a prefix notation. Words in bold denote program nodes (terminals).

```
IF <
           *
                   X1
                   4
           8
then       CLASS1
else       IF =
                   X2
                   X3
           then    CLASS2
           else    CLASS3
```

Fig. 1. Tree representation, according to our grammar, for the example expression: IF ((X1*4)<8) then CLASS1 else (IF (X2=X3) then CLASS2 else CLASS3)

grammar, we may notice that this design enables the existence of one-node solutions, for example CLASS3. Although this case is absolutely normal, it was proved in our experiments that it can delay the solution search, by creating many single-class individuals (as one such individual may have good fitness in the beginning of the run) and thus reducing the population diversification very early, leading to local optima. To encounter this phenomenon, we selected to apply two measures. First, we tuned the selection probability for operations, between the <clause> representatives (i.e. we gave more selection probability to <if> rather than to <class> nodes). Second, we modified the fitness measure as shown in the following equations, in order not to promote very small-sized solutions. This modified fitness measure performs a penalty to small-sized solutions adapting the size of the solution examined. Although these measures do not guarantee that the run will not converge early, it was shown in our experiments that the algorithm performed statistically better (i.e. we did not encounter early convergence to local optima). This fitness measure is given by the equations:

$$F = a\phi \tag{1}$$

$$\phi = \sum_{t=0}^{n-1} (1 : f_t = Y_t \mid 0 : f_t \neq Y_t) \tag{2}$$

where *F* is the program fitness, *t* is a record in the training set, *n* is the number of training records, f_t is the program output for the record *t*, and Y_t is the value of the record *t*. The *a* factor, is given by the following equation:

$$a = \left(1 : S < 0.95 \middle| 1 - \frac{2(S-0.5)-0.94}{0.05} : S \geq 0.95\right) \quad (3)$$

where S is the simplicity factor. This factor is presented in [1] and its value is:

$$S = \frac{M - 0.5N - 0.5}{M - 1} \quad (4)$$

where M stands for the maximum size of trees allowed in our application (in nodes), N stands for the examined solution's size (in nodes). This value ranges from 0.5 to 1, producing 0.5 when the expression has the maximum size and 1 when the expression has only one node (the simpler case). This fitness measure will reduce the actual fitness value, when the simplicity's value of the solution is greater than 0.95, in a linear manner, finally producing zero fitness value when the simplicity's value is one.

III. Results and Discussion

A. Balance-Scale Data

The first data set we selected to apply the system is the Balance-Scale [14]. This data was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance.

The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight).

TABLE II
GP parameters for Balance-Scale

Parameters	Values
Population:	12,000 individuals
GP implementation:	Steady-state Type-Constrained GP
Selection:	Tournament with elitist strategy
Tournament size:	6
*Crossover Rate:	0.6
*Overall Mutation Rate:	0.4
*Node Mutation Rate:	0.4
*Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed formula size:	250 tree-nodes per individual
GPFunction set	IfGreater(x1,x2,y1,y2), IfLess(x1,x2,y1,y2), IfEqual(x1,x2,y1,y2), Multiplication (*)
GPTerminal set	Class Values, Attribute Values
* operations are subject to type-constraints	

If they are equal, it is balanced. The data set has 625 examples (49 balanced, 288 left and 288 right). As we expect no numbers to appear in the solution formula and no other operators except multiplication, we adapted the function set shown in Table II. We separated the data set into training and test sets with 313 and 312 examples accordingly. Odd examples were assigned to training set and even examples to test set. By applying this selection we resulted into having 45 examples of balanced records in the training set and only 4 of them in the test set. While this kind of selection was not deliberate, in Figure 2, where the training phase is shown, it is seen that, in the test set validation, the genetic programming process achieved almost accurate results earlier than in the training set. This unusual outcome is however due to the coverage of the extracted formula of the two multitudinous classes in test data set, that are these of Left and Right class.

The extracted solution, achieved after 6,757,670 iterations (~281 generations), classified correctly both the 100% of the training set examples and the 100% of the testing set examples. As it is seen from Figure 3, the formula is rather complex as compared to the original

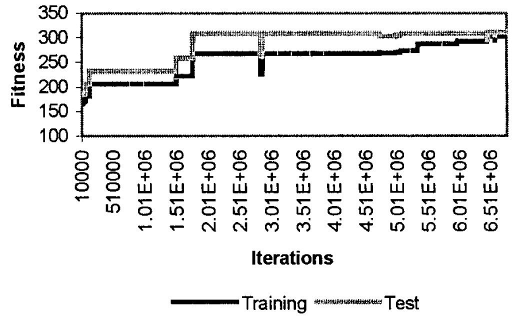

Fig. 2. Best individual during run for Balance-Scale.

```
IF RW*RD < LW*LD
 then L
 else IF RW > RW
    then R
    else IF RD*RW = LW*LD
       then B
       else IF LW < RD
          then R
          else IF RW > LD
             then R
             else IF LW < RD
                then IF LD > RW
                   then R
                   else L
                else IF LW > LW
                   then B
                   else IF LD < RD
                      then IF LW > LW
                         then IF LW > LD
                            then L
                            else R
                         else IF LW*LW > RD
                            then R
                            else R
                      else IF RW > RW
                         then R
                         else IF RD*RW*RD > LW
                            then B
                            else L
```

Fig. 3. Hierarchical classification tree for the Balance-Scale produced by our model

determining formula of this data set. This is due to the fact that the penalties concerning the solution size, that are included in the algorithm -as discussed in the previous paragraph- abet the generation of middle or large-sized solutions. Nevertheless, this formula can easily be simplified into the determining one as many of the formula parts have constant sub-result -i.e. they have no effect to the formula's value-, parts known as *introns*. This example demonstrates that if a deterministic generation formula exists for the data involved, the proposed genetic programming configuration, found with a proper function set, will probably reveal it in an efficient amount of computational effort. The prime advantage of our solution remains though, that it is easily comprehensive by humans, and underlies the knowledge extraction.

TABLE III
E-COLI PROTEIN LOCALIZATION SITE ATTRIBUTE INFORMATION

Symbol	Value
MCG	McGeoch's method for signal sequence recognition
GVH	von Heijne's method for signal sequence recognition
LIP	von Heijne's Signal Peptidase II consensus sequence score
CHG	Presence of charge on N-terminus of predicted lipoproteins
AAC	score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins
ALM1	score of the ALOM membrane spanning region prediction program
ALM2	score of the ALOM program after excluding putative cleavable signal regions from the sequence

B.Protein Localization Sites : the E-Coli database

TABLE IV
E-COLI PROTEIN LOCALIZATION SITE CLASS DISTRIBUTION

Symbol	Value	Number of Examples
CP	cytoplasm	143
IM	inner membrane without signal sequence	77
PP	periplasm	52
IMU	inner membrane, uncleavable signal sequence	35
OM	outer membrane	20
OML	outer membrane lipoprotein	5
IML	inner membrane lipoprotein	2
IMS	inner membrane, cleavable signal sequence	2

The second data set we applied our system is the E-Coli Protein Localization Site [15]. The aim is to classify the localization site of the protein. There are seven predictive, real-valued, attributes (see Table III) and eight different localization sites.

The data set has 336 examples and the class distribution is not uniform (see Table IV).

We separated the data in two halves. The first half was used to compare the classification ability of our system to previous work such as in [15], where a 81% classification accuracy was achieved with an ad hoc structured probability model. Furthermore, we tested the extracted prediction formula of our system to unknown data (the second half of the data set), in order to reveal the generalization abilities of the formula. As with the Balance-Scale data, in separating the data set in two halves we selected for the first data set the odd examples and for the second data set the even examples.

The genetic programming parameters are shown in Table V. The solution presented in Figure 4 was obtained after 44,010,000 iterations (~1833 generations). It succeeded in classifying correctly 143 out of the 168 data examples given (accuracy: 85.12 %). Furthermore, when this formula

```
IF ALM1>CHG
 then IF 92%(-29)>ALM2
    then IM
    else IF MCG>CHG
       then IF LIP<120
          then IF 60%(-29)>ALM2
             then CP
             else IF GVH>ALM1
                then OM
                else IF ALM1<MCG
                   then IMU
                   else IF 85<CHG
                      then PP
                      else IF AAC>CHG
                         then IM
                         else IF ALM1>LIP
                            then IF GVH>CHG
                               then PP
                               else IF PP=-57
                                  then IM
                                  else IMU
                            else CP
          else IF GVH>CHG
             then PP
             else IF MCG>CHG
                then PP
                else IML
       else IF ALM1<MCG
          then IMU
          else IF AAC>CHG
             then IM
             else IF MCG>GVH
                then CP
                else IM
 else IF MCG>CHG
    then IF MCG>AAC
       then IF MCG=GVH
          then IF GVH>CHG
             then PP
             else CP
          then IF MCG=GVH
             then IF OML<-57
                then OML
                else OM
             else IF GVH>CHG
                then PP
                else CP
       else OM
    else IF MCG>CHG
       then IMU
       else IF ALM1>LIP
          then PP
          else CP
```

Fig. 4. Hierarchical classification tree for the the E-Coli Protein Localization Site produced by our model

TABLE V
GP PARAMETERS FOR THE E-COLI PROTEIN LOCALIZATION SITE

Parameters	Values
Population:	12,000 individuals
GP implementation:	Steady-state Type-Constrained GP
Selection:	Tournament with elitist strategy
Tournament size:	6
*Crossover Rate:	0.6
*Overall Mutation Rate:	0.4
*Node Mutation Rate:	0.4
*Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed formula size:	250 tree-nodes per individual
GPFunction set	IfGreater(x1,x2,y1,y2), IfLess(x1,x2,y1,y2), IfEqual(x1,x2,y1,y2), Addition (+), Subtraction (-), Multiplication (*), Protected Division (%)
GPTerminal set	Class Values, Attribute Values, Numbers

* operations are subject to type-constraints

was applied to unknown data it correctly classified 119 out of 168 examples (accuracy: 70.83 %).

These outcomes denote that this intelligent model achieves competitive results in classification scores and at the same time preserves the solution comprehensibility. Moreover, this formula can be further simplified, while they exist parts of the classification tree that produce constant values.

IV. Conclusions and Further Research

This paper addressed the creation of hierarchical classification trees using genetic programming. These trees are composed by comprehensible cooperative rules and achieve a competitive classification score. Moreover, each rule's premise set is allowed to be arbitrary complex. In order to implement such a system, we embedded constraints' grammar into the common genetic programming procedure. The model was tested in two domains.

The first test showed that the system is capable of extracting underlying knowledge in a comprehensive form and the second test showed that the learning capability of this system is high. The overall results are encouraging, denoting the effectiveness of our system. Further research could involve the application of this system to other real-world data as well as the incorporation and testing of different function sets. On the other hand, tuning investigation of the algorithm can be proved valuable in terms of speed efficiency.

References

[1] Célia C.Bojarczuk, Heitor S.Lopes, Alex A.Freitas, "Genetic Programming for Knowledge Discovery in Chest-Pain Diagnosis", in IEEE Engineering in Medicine and Biology, pp. 38-44, July 2000

[2] G. Dounias, A.Tsakonas, J.Jantzen, H. Axer, B. Bjerregaard, D. G. v.Keyserlingk, "Genetic Programming for the Generation of Crisp and Fuzzy Rule Bases in Classification and Diagnosis of Medical Data", in Proc. of NF-2002, Habana, Cuba, 2002

[3] John R.Koza, "Genetic Programming - On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992

[4] Nauck Detlef and Kruse Rudolf, "NEFCLASS – a Neuro-Fuzzy approach for the classification of data", In K. M. George, J. H. Carrol, Ed. Deaton, D. Oppenheim, J. Hightower, (eds.), Applied Computing, ACM Symposium on Applied Computing, Nashville, Feb. 26-28, pp. 461-465, ACM Press, New York, February 1995.

[5] E.Alba, C.Cotta, J.M.Troya, "Type-Constrained Genetic Programming for Rule-Base Definition in Fuzzy Logic Controllers", in John R. Koza, David E. Goldberg, David B. Fogel, Rick L. Riolo (eds.), Proc. of the 1st Ann. Conf. on Genetic Programming, Stanford Univ., Cambridge, MA. The MIT Press, pp. 255-260, 1996.

[6] Quinlan, J.R., "Induction of Decision Trees", Machine Learning, Vol. 1, pp. 81-106, 1986.

[7] Elstein A. S., Shulman Lee S., Sprafta S. A.,"Medical problem solving: an analysis of clinical reasoning", Cambridge, Harvard University, 1978

[8] John R.Koza, "Genetic Programming II - Automatic Discovery of Reusable Programs", The MIT Press, 1994

[9] John R.Koza, Forrest H.Bennett III, David Andre, Martin A. Keane, "Genetic Programming III", Morgan Kaufmann Publishers, Inc., 1999

[10] F.Gruau, "On Using Syntactic Constraints with Genetic Programming", in P.J.Angeline, K.E.Jinnear,Jr. (eds.), "Advances in Genetic Programming", MIT,1996

[11] D.J.Montana, "Strongly Typed Genetic Programming", in Evolutionary Computation, vol. .3, no. 2, 1995

[12] C.Ryan, J.J.Collins, M. O'Neil, "Grammatical Evolution: Evolving Programs for an Arbitrary Language", in W.Banzhaf, R.Poli, M.Schoenauer, T.C.Fogarty (eds.), "Genetic Programming", Lecture Notes in Computer Science, Springer, 1998

[13] P.Whigham, "Search Bias, Language Bias and Genetic Programming", in Genetic Programming 1996, pp. 230-237, MIT Press, 1996

[14] Siegler, R.S., "Three Aspects of Cognitive Development", Cognitive Psychology, vol. 8, pp. 481-520, 1976

[15] Paul Horton and Kenta Nakai, "A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins", Intelligent Systems in Molecular Biology, pp. 109-115, St.Louis, USA, 1996