A Grammatical Evolution Algorithm for Generation of Hierarchical Multi-Label Classification Rules

Ricardo Cerri,
Rodrigo C. Barros,
André C. P. L. F. de Carvalho
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, SP, Brazil
Email: {cerri,rcbarros,andre}@icmc.usp.br

Alex A. Freitas
School of Computing
University of Kent
Canterbury, Kent, UK
Email: A.A.Freitas@kent.ac.uk

***Abstract*—Hierarchical Multi-Label Classification (HMC) is a challenging task in data mining and machine learning. Each instance in HMC can be classified into two or more classes simultaneously. These classes are structured in a hierarchy, in the form of either a tree or a directed acyclic graph. Therefore, an instance can be assigned to two or more paths from the hierarchical structure, resulting in a complex classification problem with hundreds or thousands of classes. Several methods have been proposed to deal with such problems, including several algorithms based on well-known bio-inspired techniques, such as neural networks, ant colony optimization, and genetic algorithms. In this work, we propose a novel global method called GEHM, which makes use of grammatical evolution for generating HMC rules. In this approach, the grammatical evolution algorithm evolves the antecedents of classification rules, in order to assign instances from a HMC dataset to a probabilistic class vector. Our method is compared to bio-inspired HMC algorithms in protein function prediction datasets. The empirical analysis conducted in this work shows that GEHM outperforms the bio-inspired algorithms with statistical significance, which suggests that grammatical evolution is a promising alternative to deal with hierarchical multi-label classification of biological data.**

I. Introduction

Hierarchical multi-label classification is a challenging task in data mining and machine learning. It differs from the well-known flat single-label classification approach regarding the organization and assignment of classes.

In flat single-label classification, each instance is assigned to a single class. The classes are mutually-exclusive and there are no assumptions regarding their relationships. In hierarchical single-label classification (HC) [1], the classes are organized in a well-defined hierarchy, which dictates the relationship among classes — some classes are specializations of other classes. This hierarchy is usually found in the form of either a tree or a directed acyclic graph (DAG). Each instance in a HC scenario may be assigned to more than one class, though these classes must be located within the same hierarchical path. In hierarchical multi-label classification (HMC), the classes are also organized in a well-defined hierarchy, in the form of a tree or a DAG, though there are no constraints regarding the assignment of classes in different paths — each instance may be assigned to classes in distinct paths within the hierarchy of classes.

HMC problems, due to their own nature, are quite complex to be handled. They are much more challenging than flat single-label classification for the following reasons [2]:

- In most cases, it is more difficult to discriminate between classes located at the bottom of the hierarchy than classes at the top of the hierarchy, since the number of instances per class tends to be smaller at lower levels of the hierarchy as opposed to top levels of the hierarchy.
- Class predictions must satisfy hierarchical parent-child relationships, given that an instance associated with a class is automatically associated with all its ancestors classes.
- Multiple unrelated classes (classes which are not involved in ancestor/descendant relationship) may be predicted at the same time.

Well-known examples of these complex HMC problems are the tasks of text classification [3]–[5] and protein function prediction [6]–[8]. The latter is an increasingly important research field, bearing in mind the availability of unknown proteins for analysis and determination of their biological functions. Protein function prediction can be seen as a data mining problem in which each protein is a dataset instance, whereas different protein features are used as predictive attributes, and the objective is to classify these proteins according to different functions they can perform. A protein can perform multiple functions, and these functions are usually organized in a hierarchical structure (*e.g.*, the FunCat [9] and Gene Ontology [10] protein functional-definition schemes), characterizing the protein function prediction as a typical HMC problem.

The focus of this work will be the protein function prediction HMC problem, and therefore we highlight the importance of generating comprehensible classification models — a model which can be validated by biologists — in order to provide new insights about the correlation of protein features and their functions. For a detailed study regarding the importance of comprehensible models in protein function prediction, please refer to [11].

Well-known comprehensible classification models are decision rules, in which a set of `if-then-else` rules are employed for classifying instances according to their predictive attributes. In the context of HMC problems, a possible example of decision rule would be:

$$\texttt{if } A_1 \geq 0.75 \texttt{ AND } A_2 \leq 3.59 \texttt{ ... AND } A_n > 0.20$$
$$\texttt{then } x_i = \{C_1, C_{1.1}, C_2, C_{2.3}, C_{2.3.1}\}$$

where A_j is the j^{th} predictive attribute, x_i is the instance whose classes should be predicted, and C_l is a given class within the hierarchy of classes.

In this paper, we propose generating HMC rules with a novel *grammatical evolution* algorithm, namely GEHM (Grammatical Evolution for Hierarchical Multi-label classification). To the best of our knowledge, this is the first work to present a grammatical evolution approach for solving HMC problems. Grammatical evolution is the state-of-the-art in grammar-based evolutionary algorithms. It combines the advantages of grammar-based GP — flexibility and ability of incorporating prior knowledge to the problem being tackled — with the advantages of genetic algorithms — simplicity of breeding operations over vectors. In addition, grammatical evolution allows for neutral mutations, the so-called degenerative genetic code [12].

This paper is organized as follows. Section II briefly presents work related to our approach. Section III details GEHM, our novel algorithm for generating HMC rules. Section IV describes the methodology employed during the empirical analysis which is itself presented in Section V. We conclude this paper and point to future research in Section VI.

II. Related Work

Several previous studies found in the literature have investigated new alternatives to solve HMC problems for protein and gene function prediction. Some of them make use of "black-box" approaches, which do not provide any means to understand the reasons that have led to a certain prediction. Others make use of comprehensible approaches, such as decision trees and decision rules, allowing for user interpretation and hypothesis' creation. These two categories are used for dividing the existing work in HMC.

Recall that, in this paper, we propose a comprehensible approach for HMC. It generates rules that are easily interpreted, allowing the user (biologist, bioinformatician, etc.) to understand the process that has led to a given prediction. To the best of our knowledge, our approach is the first to employ the Grammatical Evolution (GE) technique to hierarchical multi-label classification.

A. Algorithms Producing Black-Box Models

In Kiritchenko [13], [14], a black-box strategy was proposed for the classification of Gene Ontology (GO) [10] genes based on the classification of documents from the MedLine repository that describe these genes. This method expands the sets of classes by including their ancestral classes and then applies the AdaBoost algorithm [15] in the modified dataset.

Barutcuoglu et al. [6] proposed a black-box strategy that employs a hierarchy of SVM [16] classifiers for the prediction of gene functions structured according to the GO. The classifiers are trained separately for each class, and the predictions are then combined using Bayesian Networks [17], aiming at finding the most probable consistent set of predictions.

An ensemble of classifiers, named True Path Rule Ensemble (TPR), was proposed by Valentini [18]. In this method, each trained classifier estimates the local probability $\hat{p}_j(\mathbf{x}_i)$ that a given instance $\mathbf{x}_i$ belongs to a given class c_j. A combination phase estimates the global consensual probability $p_j(\mathbf{x}_i)$. In Valentini and Re [19] and Valentini [20], the authors modified this method in order to modulate the relationship between the prediction of a class and the prediction of its descendants.

Cerri et al. [21] proposed a black-box approach that employs a sequence of connected artificial neural networks for protein function prediction. Each network is associated to a hierarchical level, and the output of the network in level l is used as the input of the network in level $l + 1$. A strategy for avoiding inconsistent predictions is employed, since a given neural network may predict a class whose superclass had not been predicted before.

B. Algorithms Producing Comprehensible Models

One of the first HMC methods was proposed by Clare and King [22]. This method, named HMC4.5, is based on decision-tree induction algorithms. It is a variant of C4.5 [23] with modifications in the calculation of class entropy. In the original C4.5 algorithm, the entropy is used to decide the best data split in the decision tree, *i.e.*, the best attribute to be placed in a tree node. The proposed modification uses the sum of the number of bits needed to describe membership or non-membership of each class, and also the information related to the size of the tree rooted by a given class.

Vens et al. [8] proposed three methods based on the concept of Predictive Clustering Trees (PCT). The authors proposed the global Clus-HMC method [24] that induces a single decision tree to cope with the entire classification problem. They compared its performance with two local methods. The first one, Clus-SC, induces an independent decision tree for each class of the hierarchy, ignoring the relationships between classes. The second one, Clus-HSC, explores the hierarchical relationships between the classes to induce a decision tree for each class. The authors applied the methods to hierarchies structured as trees and DAGs, and discussed the modifications needed so the algorithms could cope with both types of hierarchical structures.

Alves et al. [25] proposed a global method using Artificial Immune Systems (AIS) [26] for the generation of HMC rules. The method, named Multi-label Hierarchical Classification with an Artificial Immune System (MHC-AIS), is divided into two basic procedures: Sequential Covering (SC) and Rule Evolution (RE). These procedures produce candidate classification rules whose antecedent (IF part) is represented by a vector of attribute-value conditions and the consequent (THEN part) is represented as a set of predicted classes. The SC procedure iteratively calls the RE procedure until all (or almost all) training instances (antigens) are covered by the discovered rules. The RE procedure evolves classification rules (antibodies) that are used to classify the instances. The best antibody is added to the set of discovered rules. The authors extend their work by proposing new procedures to improve the algorithm's prediction performance [27].

Sangsuriyun et al. [28] proposed a comprehensible method based on rule sets, named Hierarchical Multi-Label Associative Classification (HMAC), which can be applied to both tree and DAG structured hierarchies. The method uses the so-called negative rules, which consider important the absence of a given attribute for the classification of an instance. The method also takes into account the rules that predict a negative set of classes, *i.e.*, rules that indicate that an instance does not belong to a given set of classes.

Otero et al. [29] proposed *h*Ant-Miner, a global method for hierarchical single-label classification using Ant Colony Optimization (ACO) [30]. Two ACO algorithms are employed to construct the rules in an cooperative manner, one for optimizing the antecedents and the other for rule consequent. The authors extended this method proposing *hm*Ant-Miner, which allows multi-label classification [2], considering both tree and DAG structured hierarchies. The *hm*Ant-Miner discovers hierarchical multi-label classification rules in the format `if-then`.

Cerri et al. [31] proposed a Genetic Algorithm (GA) to produce HMC rules. The GA evolves the antecedents of classification rules in order to optimize the level of coverage of each antecedent. The fitness function employed in that work gives a better reward to rules with the antecedents which cover a higher number of instances. The set of optimized antecedents is selected to build the corresponding consequent of the rules (set of classes to be predicted).

III. GEHM

Grammatical Evolution for Hierarchical Multi-label classification (GEHM) is a grammatical evolution (GE) algorithm aimed at generating hierarchical multi-label classification rules. Given that GEHM is a grammar-based evolutionary algorithm (EA), it differs from traditional EAs in the use of a grammar G in the evolutionary process. The grammar G defines a language L whose terms are the functions and terminals.

It is important to understand the differences between grammar-based genetic programming (GGP) and GE. In GGP, the individual that undergoes evolution is a derivation tree. Both genotype and phenotype are represented as the same structure (the derivation tree). In GE, genotype and phenotype have different encoding structures. The genotype is represented as a linear string of codons (sequence of 8 bits) of variable size, which is then decoded into a vector of integers. The phenotype, which is the structure that defines fitness, is represented by a derivation tree (resulting from the application of a grammar). Hence, a mapping from genotype to phenotype is required in a GE algorithm. The mapping process to decode the string chromosomes into derivation trees is called *genotype-phenotype mapping* (GPM). Among the alleged benefits of GE are the unconstrained search of the genotype while ensuring phenotype validity, and enhancing genetic diversity by allowing mutations which are neutral with respect to the phenotype (various genotypes can represent the same phenotype) [12]. Figure 1 depicts the typical GE scheme.

For performing the genotype-phenotype mapping, the following procedure is executed. First, each codon is converted

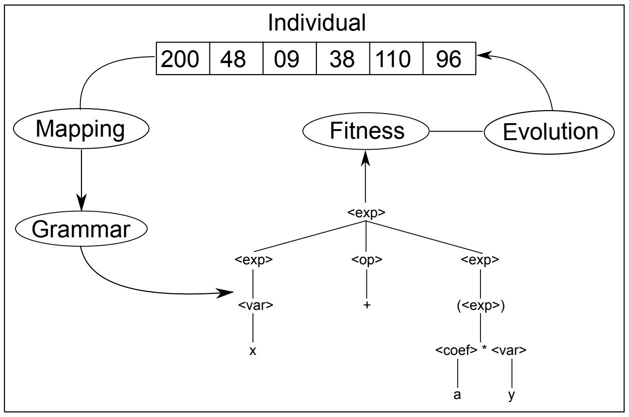

Fig. 1. Grammatical evolution typical scheme.

into an integer value int. Then, the GPM function is given by:

$$GR_i = int \bmod nr_i \tag{1}$$

where GR_i is the index of the grammar rule to be selected in non-terminal i, int is the codon integer value and nr_i is the number of rules available for derivation in non-terminal i. The operator mod returns the remainder of the division of one number by another. For exemplifying this procedure, considering the following rule in a context-free grammar:

```
<exp>::= <exp> + <exp> | <exp> - <exp> | <number>
```

Now, consider that the codon responsible for deriving grammar rule `<exp>` represents the integer 79. Since `<exp>` offers three options for further derivation, the value of nr_i is 3. Hence, solving Equation (1) results in:

$$GR_i = 79 \bmod 3 = 1 \tag{2}$$

Since each derivation option is assigned an index — `<exp> + <exp>` is assigned index 0, `<exp> - <exp>` is assigned index 1, and `<number>` is assigned index 2 —, the second derivation option is chosen (`<exp> - <exp>`) because $79 \bmod 3 = 1$.

A. Grammar

In GEHM, each individual of the evolutionary algorithm corresponds to the antecedent of a HMC rule. For mapping the string of codons into a derivation tree, we make use of the grammar presented in Figure 2.

This grammar provides typical Boolean operators such as $\geq, \leq$, and also a double operator such as $num1 \leq x \leq num2$. Non-terminal `<att>` selects randomly one of the dataset's attributes (an attribute index), and non-terminal `<num>` generates random continuous values in the range [0,1] from a uniform distribution (an ephemeral random constant). Operators `AND` and `OR` perform their respective Boolean operations. We believe this simple grammar is capable of generating a thorough search-space of HMC rules' antecedents.

```
1) <start>::= <exp>
2) <exp>::= <exp2> AND <exp2> | <exp> AND
<exp> | <exp2> OR <exp2> | <exp> OR <exp>
3) <exp2>::= <att> ≥ <num> | <att> ≤ <num>
4) <exp2>::= <num> ≤ <att> ≤ <num> | <exp>
5) <att>::= randomly chosen dataset attribute
6) <num>::= U(0,1)
```

Fig. 2. Grammar for deriving antecedents of HMC rules.

B. Genetic Operators

In the beginning of the evolutionary process, the individuals of the population are randomly initialized. They have variable length, with a minimum size of five codons each, and a chance of 85% that new codons will be incrementally added to the individual. Recall that each codon is comprised of 1 byte (8 bits), which is randomly generated.

To evolve the current generation of individuals, the following mutually-exclusive genetic operations can be performed: crossover, mutation, and duplication. Crossover has a chance of 90% of being performed, and both duplication and mutation have a chance of 5% of being applied. These operations are executed until all individuals of the new population are generated.

For crossover to be performed, two individuals are chosen via tournament selection. After the individuals are selected, they take part in a standard one-point crossover operation, generating two children. In the duplication process, one individual is also selected using tournament selection, and then two codons of the individual are randomly selected, copying all codons located between these two selected codons to the end of the individual. Finally, mutation requires one individual to be selected via tournament selection. This individual is traversed codon by codon, where each codon has a 10% probability of having its value replaced by a randomly-generated 8-bit value. The three operators are illustrated in Figure 3.

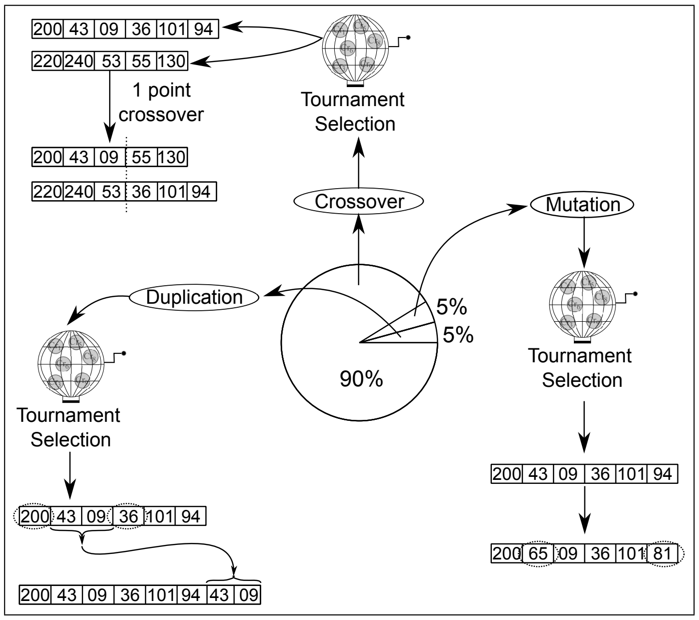

Fig. 3. GEHM's genetic operators.

C. Sequential Covering

GEHM is both generational EA and sequential covering rule algorithm. In a sequential covering algorithm, instances which are covered by a rule are removed from the training set, so the new rules generated can fit the remaining uncovered instances. Algorithm 1 shows this sequential covering strategy.

GEHM runs a full evolutionary cycle (lines 3-34), and then saves the best rule from the last generation. Furthermore, all instances which are covered by a rule are removed from the training set (line 35). Next, GEHM starts a full new evolutionary cycle, and this process is repeated until only a few instances (user-defined parameter) are left uncovered, or until there are no instances left in the training set.

Algorithm 1 GEHM.

```
Require: Training dataset $X$, number of generations $N$,
    population size $p$, maximum number of uncovered instances
    $maxUncovered$, grammar $G$.
 1: $Rules \leftarrow \emptyset$
 2: while ($|X| > maxUncovered$) do
 3:   Randomly generate $InitialPop$
 4:   $Phenotype \leftarrow mapping(InitialPop, G)$
 5:   $calculateFitness(Phenotype, X)$
 6:   $CurPop \leftarrow InitialPop$
 7:   $j \leftarrow N$
 8:   while ($j > 0$) do
 9:     $NewPop \leftarrow \emptyset$
10:     $NewPop \leftarrow NewPop \cup CurPop.elite()$
11:     repeat
12:       Switch genetic operator
13:       Case crossover:
14:       $Parents \leftarrow selection(CurPop, 2)$
15:       $Children \leftarrow crossover(Parents)$
16:       $NewPop \leftarrow NewPop \cup Children$
17:       end case
18:       Case Duplication:
19:       $Parent \leftarrow selection(CurPop, 1)$
20:       $Child \leftarrow duplication(Parent)$
21:       $NewPop \leftarrow NewPop \cup Child$
22:       end case
23:       Case mutation:
24:       $Parent \leftarrow selection(CurPop, 1)$
25:       $Child \leftarrow mutation(Parent)$
26:       $NewPop \leftarrow NewPop \cup Child$
27:       end case
28:     until ($|NewPop| = p$ )
29:     $CurPop \leftarrow \emptyset$
30:     $CurPop \leftarrow NewPop$
31:     $Phenotype \leftarrow mapping(CurPop, G)$
32:     $calculateFitness(Phenotype, X)$
33:     $j \leftarrow j - 1$
34:   end while
35:   $Rules \leftarrow Rules \cup CurPop.bestRule()$
36:   Remove instances from $X$ covered by $Rules$
37: end while
38: return $Rules$
```

D. Consequent Generation and Fitness Function

GEHM performs the same deterministic procedure for generating the consequent of a rule as HMC-GA [31] and *hm*Ant-Miner [2], as follows. Given the set of instances X_r covered by a rule r, the consequent of a rule is a vector of length k (where k is the number of class labels in the class hierarchy), which means each position of the vector corresponds to a given class of the hierarchy. The value for each component of the consequent vector for rule r is given by:

$$consequent_{r,i} = \frac{|X_r \wedge C_i|}{|X_r|} \tag{3}$$

where $|X_r \wedge C_i|$ is the number of instances covered by rule r that belong to class C_i. Thus, the consequent of a rule is given by the mean class label vector of all instances that are covered by that rule.

According to Equation (3), each position of the consequent vector is a continuous value in the range [0,1], instead of a Boolean value indicating whether or not the particular class is being predicted. As a result, the value in the i^{th} component of the consequent vector of a rule represents the probability of an instance that satisfies its antecedent to belong to class C_i of the hierarchy. In order to obtain the class label predictions from a rule, one needs to select a classification threshold. If the value of the i^{th} component of the consequent is greater than or equal to the classification threshold, the C_i class label is predicted. Therefore, GEHM is a probabilistic method whose predictions vary according to the "degree of certainty" the user defines for predicting classes in the hierarchy. Lower thresholds lead to a larger number of classes being predicted, whereas higher thresholds lead to a smaller number of predicted classes.

Once the HMC rules have their consequent built according to the previously described procedure, the rule can be evaluated through a fitness function. We have chosen the *variance gain* [2] as the fitness function for GEHM. Variance gain measures the decrease of variance achieved among instances that are covered by the rule and instances that are not. The variance gain is given by:

$$\begin{aligned} varianceGain(X, r) = {} & variance(X) \\ & -\frac{|X_r|}{|X|} \times variance(X_r) \\ & -\frac{|\neg X_r|}{|X|} \times variance(\neg X_r) \end{aligned} \tag{4}$$

where $variance(\cdot)$ is the variance of the class vectors of all instances belonging to the set that has been passed as argument, $|X_r|$ is the number of instances that are covered by rule r, and $|\neg X_r|$ is the number of instances that are not covered by rule r. Note that the smaller the variance within the covered and uncovered sets, the larger the variance gain, and thus the better the rule. An ideal rule would provide the best discrimination between the covered and not covered instances (high variance gain).

IV. Methodology

In this section, we present the methodology employed during the empirical analysis. We present the baseline algorithms (Section IV-A), the datasets (Section IV-B), and the evaluation measures (Section IV-C) employed for assessing the performance of GEHM in the context of protein function prediction.

A. Baseline Algorithms and Parameters

Three of the methods reviewed in Section II are employed as the baseline methods for the experiments performed in this work. We make use of the Ant Colony Optimization (ACO) based method *hm*Ant-Miner [2], which is a method that employs ACO for generating HMC rules. GEHM is also compared to HMC-GA [31], which is a genetic algorithm that also evolves the antecedent of HMC rules. The third algorithm used as a baseline is HMC-LMLP [21], which is a local method that creates a sequence of connected neural networks throughout the levels of the hierarchy of classes. We chose these methods because they are all based on bio-inspired techniques: ant colony optimization, genetic algorithms, and neural networks. In addition, they produce the same type of output provided by GEHM — a probabilistic class vector — allowing the computation of the evaluation measures presented in Section IV-C.

Table I shows the user-defined parameter values used in GEHM. No attempt to tune these parameter values was made, since we do not try to optimize the parameters of the baseline methods either. Parameter optimization is a topic left for future research.

TABLE I. GEHM PARAMETERS.

Parameter	Description	Value
maxUncovered	Maximum number of uncovered instances	10
G	Number of generations	100
p	Population size	200
lp	Probability of adding codon in initialization (rate)	85%
cr	Crossover rate	90%
mr	Mutation rate	5%
dr	Duplication rate	5%
mp	Mutate gene probability	10%
t	Tournament size	3
e	Elitism rate	0.5%

B. Datasets

Ten freely available[1] datasets related to protein function prediction are used in the experiments. These datasets are related to issues like phenotype data and gene expression levels. They are organized according to two different class hierarchy structures: tree structure (FunCat data sets) and directed acyclic graph structure (Gene Ontology data sets). The directed acyclic graph (DAG) structure represents a complex hierarchical organization, where a particular node of the hierarchy can have more than one parent, in contrast to only one parent in tree structures.

Table II presents the main characteristics of the training, validation, and test datasets employed in the experiments. Usually, the training set is used to generate the prediction model for a set of algorithmic parameters, whereas the validation set is used to choose the models that yielded the best predictive performance among those generated by different parameter settings. Finally, the test set is used to evaluate the model in unseen cases, in order to provide a realistic assessment of performance. In the particular case of the rule-based algorithms employed in the experimental analysis (including GEHM), the training and validation datasets are merged and used together to generate the predictive models, and then the resulting set of rules is used to classify the instances that belong to the test set. For the case of HMC-LMLP, the validation set is used to evaluate the neural networks across different epochs (for more details, please refer to [21], [32]).

Note that these are very imbalanced datasets, with very few positive instances for each class. A description of each dataset can be found in [8]. The datasets used in this paper have only numeric attributes, since the current version of GEHM's

[1]http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets.html

TABLE II. SUMMARY OF DATASETS: NUMBER OF ATTRIBUTES ($|A|$), NUMBER OF CLASSES ($|C|$), TOTAL NUMBER OF INSTANCES (TOTAL) AND NUMBER OF MULTI-LABEL INSTANCES (MULTI).

Structure	Dataset	$\|A\|$	$\|C\|$	Training		Valid		Test	
				Total	Multi	Total	Multi	Total	Multi
Tree	Cellcycle	77	499	1628	1323	848	673	1281	1059
	Derisi	61	499	1608	1309	842	671	1275	1055
	Eisen	79	461	1058	900	529	441	837	719
	Gasch1	173	499	1634	1325	846	672	1284	1059
	Gasch2	52	499	1639	1328	849	674	1291	1064
DAG	Cellcycle	77	4125	1625	1625	848	848	1278	1278
	Derisi	61	4119	1605	1605	842	842	1272	1272
	Eisen	79	3573	1055	1055	528	528	835	835
	Gasch1	173	4125	1631	1631	846	846	1281	1281
	Gasch2	52	4131	1636	1636	849	849	1288	1288

grammar cannot cope with nominal attributes. For executing GEHM in these datasets, all missing values were replaced with the mean value of the respective attribute.

Also note that the only baseline method that provided results for the DAG-based datasets is *hm*Ant-Miner [2], so we present a comparison of GEHM with the three baseline methods only for the five tree-based datasets, whereas GEHM is only compared with *hm*Ant-Miner [2] in all datasets (the five tree-based datasets plus the five DAG-based datasets).

C. Evaluation Measures

Since every algorithm tested in this work outputs a vector of class probabilities for each instance being predicted, we make use of the area under the average PR-curve ($AU(\overline{PRC})$) as the evaluation measure to compare them. To obtain a PR-curve for a given algorithm, different thresholds ranging within [0,1] are applied to the outputs of the methods, and thus different values of precision and recall are obtained, one for each threshold value. Each threshold value then represents a point within the PR space. The union of these points form a PR-curve, and the area below the curve is calculated. In order to calculate the area below the PR-curve, the PR-points must be interpolated [33]. This interpolation guarantees that the area below the curve is not artificially increased, which would happen if the curves were constructed just connecting the points without interpolation. Given a threshold value, a precision-recall point ($\overline{Prec}, \overline{Rec}$) in the PR-space can be obtained through Equations (5) and (6). They correspond to the micro-average of precision and recall,

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad (5) \qquad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \quad (6)$$

where i ranges from 1 to $|C|$, and TP, FP, and FN stand, respectively, for the number of true positives, false positives, and false negatives.

In addition to the $AU(\overline{PRC})$, the methods were also compared considering the size of the model created. For GEHM, HMC-GA, and *hm*Ant-Miner, the model size is defined as the number of rules discovered. Note that HMC-LMLP does not generate rules, and thus we do not provide such information.

V. RESULTS AND DISCUSSION

Since only the *hm*Ant-Miner algorithm is capable of dealing with DAG-based datasets, we divided the experiments into two parts. In the first part, we compare GEHM with HMC-GA and HMC-LMLP regarding the five tree-based datasets. In the second part, we compare GEHM with *hm*Ant-Miner in both tree-based and DAG-based datasets. The GEHM results reported are the means and standard deviations obtained after 10 executions of the algorithm varying the random seed. The results presented for the other methods are those provided in their respective references ([2], [21], [31]), since we are using exactly the same training, validation, and test sets.

Table III shows the results of the first part of the experiments. Note that GEHM outperforms HMC-GA regarding both average $AU(\overline{PRC})$ and model size, for all five protein function datasets. It also outperforms HMC-LMLP regarding the average $AU(\overline{PRC})$ in all datasets. The average values of $AU(\overline{PRC})$ provided by GEHM are much larger than the respective values provided by HMC-GA and HMC-LMLP. GEHM presents a larger standard deviation than HMC-GA, but it is considerably more stable than HMC-LMLP.

Despite the higher variance when compared to HMC-GA, GEHM is capable of generating rules that are much more accurate than HMC-GA in all five datasets. The fact that GEHM also generates a smaller set of HMC rules is also encouraging, since a small set of rules is much easier to interpret than a large set of rules. However, it should be noticed that the rules generated by GEHM may eventually be more complex than those generated by HMC-GA, considering the possibility of aligning several conjunctions and disjunctions, whereas HMC-GA only generates rules whose terms are connected by conjunctions.

Table IV shows the results of the second part of the experiments. In this table, we present the result for the ten datasets provided by GEHM and *hm*Ant-Miner. Note that GEHM outperforms *hm*Ant-Miner in the majority of the datasets (9 out of 10), and it generates a smaller set of rules in all datasets (though possibly more complex rules). Even though *hm*Ant-Miner seems to be more stable across different runs (smaller variance), GEHM is capable of achieving much larger values of $AU(\overline{PRC})$.

The next step of this experimental analysis is to assess the statistical significance of the results achieved by our method. Since we do not have a large number of datasets (five datasets structured as trees and five datasets structured as DAGs), and

TABLE III. Comparison of GEHM with HMC-GA and HMC-LMLP. Values are the average $AU(\overline{PRC})$ and model size obtained in the five datasets organized as trees.

	GEHM		HMC-GA		HMC-LMLP
	$AU(\overline{PRC})$	$\|Rules\|$	$AU(\overline{PRC})$	$\|Rules\|$	$AU(\overline{PRC})$
Cellcycle	**0.165 ± 0.005**	**5.10 ± 1.52**	0.150 ± 0.001	66.70 ± 13.4	0.144 ± 0.009
Derisi	**0.164 ± 0.006**	**4.30 ± 1.34**	0.152 ± 0.001	37.00 ± 10.2	0.138 ± 0.008
Eisen	**0.187 ± 0.007**	**3.70 ± 1.34**	0.165 ± 0.005	63.80 ± 7.7	0.173 ± 0.009
Gasch1	**0.176 ± 0.006**	**5.40 ± 1.35**	0.159 ± 0.004	136.8 ± 20.41	0.133 ± 0.018
Gasch2	**0.174 ± 0.008**	**4.50 ± 2.01**	0.151 ± 0.001	42.8 ± 8.77	0.127 ± 0.012

TABLE IV. Comparison of GEHM with *hm*Ant-Miner. Values are the average $AU(\overline{PRC})$ and model size obtained in the ten datasets organized both as trees and DAGs.

		GEHM		*hm*Ant-Miner	
		$AU(\overline{PRC})$	$\|Rules\|$	$AU(\overline{PRC})$	$\|Rules\|$
Tree	Cellcycle	**0.165 ± 0.005**	**5.10 ± 1.52**	0.154 ± 0.001	28.67 ± 1.62
	Derisi	**0.164 ± 0.006**	**4.30 ± 1.34**	0.161 ± 0.002	19.33 ± 1.66
	Eisen	**0.187 ± 0.007**	**3.70 ± 1.34**	0.180 ± 0.003	19.00 ± 0.98
	Gasch1	**0.176 ± 0.006**	**5.40 ± 1.35**	0.175 ± 0.003	24.87 ± 1.70
	Gasch2	**0.174 ± 0.008**	**4.50 ± 2.01**	0.152 ± 0.0006	32.33 ± 1.52
DAG	Cellcycle	**0.338 ± 0.012**	**5.40 ± 1.74**	0.332 ± 0.002	35.40 ± 1.59
	Derisi	**0.343 ± 0.006**	**5.40 ± 1.26**	0.334 ± 0.003	22.53 ± 1.94
	Eisen	**0.378 ± 0.003**	**4.40 ± 1.78**	0.376 ± 0.002	18.20 ± 0.82
	Gasch1	0.353 ± 0.005	**3.80 ± 0.92**	**0.356 ± 0.002**	27.93 ± 0.92
	Gasch2	**0.351 ± 0.004**	**4.30 ± 1.42**	0.344 ± 0.002	34.20 ± 1.63

TABLE V. Critical values for the two-tailed sign test at $\alpha = 0.05$.

#Datasets	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\alpha_{0.05}$	5	6	7	7	8	9	9	10	10	11	12	12	13	13	14	15	15	16	17	18	18

we want to perform pairwise comparisons of the classifiers, we compared the classifiers counting the number of datasets where one classifier is better than another. If they perform similarly (null hypothesis), each classifier should win in approximately $N/2$ of N datasets. For N up to 25 datasets, the number of wins is distributed according do the binomial distribution [34]. This test is known as sign test [35], and the critical number of wins is shown in Table V.

According to Table III, GEHM obtained the largest $AU(\overline{PRC})$ values and also the smallest set of rules in all datasets that were investigated. This observation alone is already enough to conclude that GEHM is statistically superior to HMC-GA and HMC-LMLP. Indeed, any non-parametric ranking-based statistical test would suggest that the difference between the results is statistically significant, given that we are performing pairwise comparisons and GEHM will always be ranked #1 whereas the other classifier will be ranked #2. Indeed, according to the sign test (Table V), we can see that GEHM is statistically superior with $\alpha = 0.05$, because it outperforms both HMC-GA and HMC-LMLP in five of the five datasets involved in the comparison.

The sign test also suggests that GEHM outperforms *hm*Ant-Miner with statistical significance regarding $\alpha = 0.05$, because GEHM wins in nine of the ten datasets investigated (column 7 of Table V). Considering that GEHM also provides the smaller set of HMC rules in all the ten datasets, we can also conclude that the difference between the results is statistically significant in favor of GEHM.

VI. Conclusion and Future Work

In this work, we proposed a method called Grammatical Evolution for Hierarchical Multi-label classification (GEHM). GEHM is a grammatical evolution algorithm aimed at generating hierarchical multi-label classification rules.

The method differs from traditional evolutionary algorithms in the use of a grammar during the evolutionary process. This grammar defines a language whose terms are functions and terminals. Productions of the grammar are encoded in the genotype of the individuals, and are mapped to derivation trees.

We investigated the use of GEHM in datasets related to the problem of protein function prediction. We made use of datasets structured as both trees and directed acyclic graphs (DAGs). Whereas in the tree structure, each class may have only one superclass, in DAGs a class may have more than one superclass, making the classification task of hierarchies structured as DAGs much more challenging.

As GEHM is a bio-inspired algorithm, we compared it with other bio-inspired algorithms in the literature. We employed three baseline methods based on genetic algorithms, ant colony optimization, and neural networks. Throughout the experimental analysis, we showed that GEHM outperforms all the baseline bio-inspired methods with statistical significance. Since every method in the experiments predicts a vector of real numbers for each instance, we used the area under the precision-recall curves ($AU(\overline{PRC})$) to evaluate the methods' prediction performance, which allows a threshold independent evaluation.

Besides obtaining the larger $AU(\overline{PRC})$ values, GEHM also provided the smaller sets of HMC rules for all datasets. Smaller sets are usually easier to interpret than larger sets, though we highlight the fact that GEHM is capable of generating more complex rules than either HMC-GA or *hm*Ant-Miner. This is because GEHM's grammar allows aligning disjunctions and conjunctions of Boolean operators, instead of the traditional sequence of conjunctions adopted by both HMC-GA and *hm*Ant-Miner. We leave as future work a more detailed analysis on the comprehensibility of the rules generated by GEHM. Also as future work, we want to apply GEHM to other HMC domains such as text categorization [13], [36]. Finally, we want to extend GEHM's grammar in order to allow the use of datasets with categorical attributes. A parameter optimization procedure can also be applied to improve the method's performance.

ACKNOWLEDGMENT

The authors would like to thank Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, for funding this research.

REFERENCES

[1] A. A. Freitas and A. C. P. F. L. Carvalho, "A tutorial on hierarchical classification with applications in bioinformatics," in *Research and Trends in Data Mining Technologies and Applications*. Idea Group, 2007, ch. VII, pp. 175–208.

[2] F. Otero, A. A. Freitas, and C. Johnson, "A hierarchical multi-label classification ant colony algorithm for protein function prediction," *Memetic Computing*, vol. 2, pp. 165–181, 2010.

[3] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *Fourth IEEE International Conference on Data Mining*, 2001, pp. 521–528.

[4] A. Sun, E.-P. Lim, W.-K. Ng, and J. Srivastava, "Blocking Reduction Strategies in Hierarchical Text Classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1305–1308, 2004.

[5] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[6] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, pp. 830–836, 2006.

[7] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare, "Decision trees for hierarchical multilabel classification: A case study in functional genomics." in *Knowledge Discovery in Databases*, 2006, pp. 18–29.

[8] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, pp. 185–214, 2008.

[9] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter, and H. W. Mewes, "The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucleic Acids Research*, vol. 32, no. 18, pp. 5539–5545, October 2004.

[10] M. Ashburner *et al.*, "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." *Nature Genetics*, vol. 25, pp. 25–29, 2000.

[11] A. A. Freitas, D. C. Wieser, and R. Apweiler, "On the importance of comprehensible classification models for protein function prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 7, pp. 172–182, January 2010.

[12] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349 –358, aug 2001.

[13] S. Kiritchenko, S. Matwin, and A. Famili, "Functional annotation of genes using hierarchical text categorization," in *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, 2005.

[14] S. Kiritchenko, S. Matwin, R. Nock, and A. Famili, "Learning and evaluation in the presence of class hierarchies: Application to text categorization," in *Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2006, vol. 4013, pp. 395–406.

[15] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Machine Learning*, vol. 37, no. 3. Hingham, MA, USA: Kluwer Academic Publishers, 1999, pp. 297–336.

[16] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1999.

[17] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[18] G. Valentini, "True path rule hierarchical ensembles," in *International Workshop on Multiple Classifier Systems*, 2009, pp. 232–241.

[19] G. Valentini and M. Re, "Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction," in *1st Workshop on Learning from Multi-Label Data (MLD) held in conjunction with ECML/PKDD*, 2009, pp. 132–145.

[20] G. Valentini, "True path rule hierarchical ensembles for genome-wide gene function prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 3, pp. 832–847, May 2011.

[21] R. Cerri and A. Carvalho, "Hierarchical multilabel protein function prediction using local neural networks," in *Brazilian Symposium on Bioinformatics*, 2011, pp. 10–17.

[22] A. Clare and R. D. King, "Predicting gene function in saccharomyces cerevisiae," *Bioinformatics*, vol. 19, pp. 42–49, 2003.

[23] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[24] H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, and J. Struyf, "Hierarchical multi-classification," in *Workshop on Multi-Relational Data Mining*, 2002, pp. 21–35.

[25] R. Alves, M. Delgado, and A. A. Freitas, "Multi-label hierarchical classification of protein functions with artificial immune systems," in *III Brazilian Symposium on Bioinformatics*, ser. Lecture Notes in Bioinformatics, vol. 5167. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–12.

[26] L. N. de Castro and J. I. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. London: Springer-Verlag, 2002.

[27] R. Alves, M. Delgado, and A. A. Freitas, "Knowledge discovery with artificial immune systems for hierarchical multi-label classification of protein functions," in *International Conference on Fuzzy Systems*, 2010, pp. 2097–2104.

[28] S. Sangsuriyun, S. Marukatat, and K. Waiyamai, "Hierarchical Multi-label Associative Classification (HMAC) using negative rules," in *International Conference on Cognitive Informatics*. IEEE, july 2010, pp. 919–924.

[29] F. Otero, A. A. Freitas, and C. Johnson, "A hierarchical classification ant colony algorithm for predicting gene ontology terms," in *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Springer, 2009, pp. 68–79.

[30] M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, IT, 1992.

[31] R. Cerri, R. C. Barros, and A. C. P. L. F. Carvalho, "A genetic algorithm for hierarchical multi-label classification," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 250–255.

[32] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho, "Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks," in *Intelligent Systems Design and Applications (ISDA)*, nov. 2011, pp. 337 –343.

[33] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *International Conference on Machine Learning*, 2006, pp. 233–240.

[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[35] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th ed. Chapman & Hall/CRC, 2007.

[36] A. Esuli, T. Fagni, and F. Sebastiani, "Boosting multi-label hierarchical text categorization," *Inf. Retr.*, vol. 11, no. 4, pp. 287–313, 2008.