Evolving Relational Hierarchical Classification Rules for Predicting Gene Ontology-Based Protein Functions

Ricardo Cerri
ICMC
Universidade de São Paulo
Campus de São Carlos
Caixa Postal 668 - 13560-970
São Carlos-SP-Brazil
cerri@icmc.usp.br
- - - - - - - -
Secretaria de Gestão
Estratégica
Empresa Brasileira de
Pesquisa Agropecuária
Pq. Estação Biológica - s/n
Brasília - DF
ricardo.cerri@embrapa.br

Rodrigo C. Barros
Faculdade de Informática
Pontifícia Universidade
Católica do Rio Grande do Sul
Av. Ipiranga, 6681,
90619-900, Porto
Alegre-RS-Brazil
rodrigo.barros@pucrs.br

Alex A. Freitas
School of Computing
University of Kent
Canterbury, Kent, CT2 7NF
A.A.Freitas@kent.ac.uk

André C. P. L. F. de Carvalho
ICMC
Universidade de São Paulo
Campus de São Carlos
Caixa Postal 668 - 13560-970
São Carlos-SP-Brazil
andre@icmc.usp.br

ABSTRACT

Hierarchical Multi-Label Classification (HMC) is a complex classification problem where instances can be classified into many classes simultaneously, and these classes are organized in a hierarchical structure, having subclasses and superclasses. In this paper, we investigate the HMC problem of assign functions to proteins, being each function represented by a class (term) in the Gene Ontology (GO) taxonomy. It is a very difficult task, since the GO taxonomy has thousands of classes. We propose a Genetic Algorithm (GA) to generate HMC rules able to classify a given protein in a set of GO terms, respecting the hierarchical constraints imposed by the GO taxonomy. The proposed GA evolves rules with propositional and relational tests. Experiments using ten protein function datasets showed the potential of the method when compared to other literature methods.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms*; I.2.m [**Computing Methodologies**]: Artificial Intelligence—*Miscellaneous*

Keywords

Hierarchical Multi-Label Classification, Gene Ontology, Propositional Rules, Relational Rules, Genetic Algorithms, Bioinformatics

1. INTRODUCTION

In conventional classification, an instance $\mathbf{x}_i \in X$ can be classified in only one class $c_j \in C$. However, there are more complicated classification problems, where an instance can be simultaneously classified into a set of classes $C_j \in C$. In Hierarchical Multi-Label Classification (HMC), an instance can be classified into a set of classes, and these classes are organized in a hierarchical taxonomy, having subclasses and superclasses. In this taxonomy, there is a partial order $\prec_h$ representing the superclass relationships, *i.e.*, for all $c_1, c_2 \in C, c_1 \prec_h c_2$ if and only if c_1 is a superclass of c_2.

Among the different applications of HMC, protein function prediction deserves to be highlighted. Proteins perform almost all functions in an organism. Their functions are related to cell activity, such as biochemical reactions, cell signaling, structural, and mechanical functions [6]. Protein functions are also hierarchically structured, which makes protein function prediction a typical HMC problem.

In this work, we investigate the protein function prediction problem using the Gene Ontology (GO) hierarchy. In the GO, the classes are organized in an hierarchy of terms,

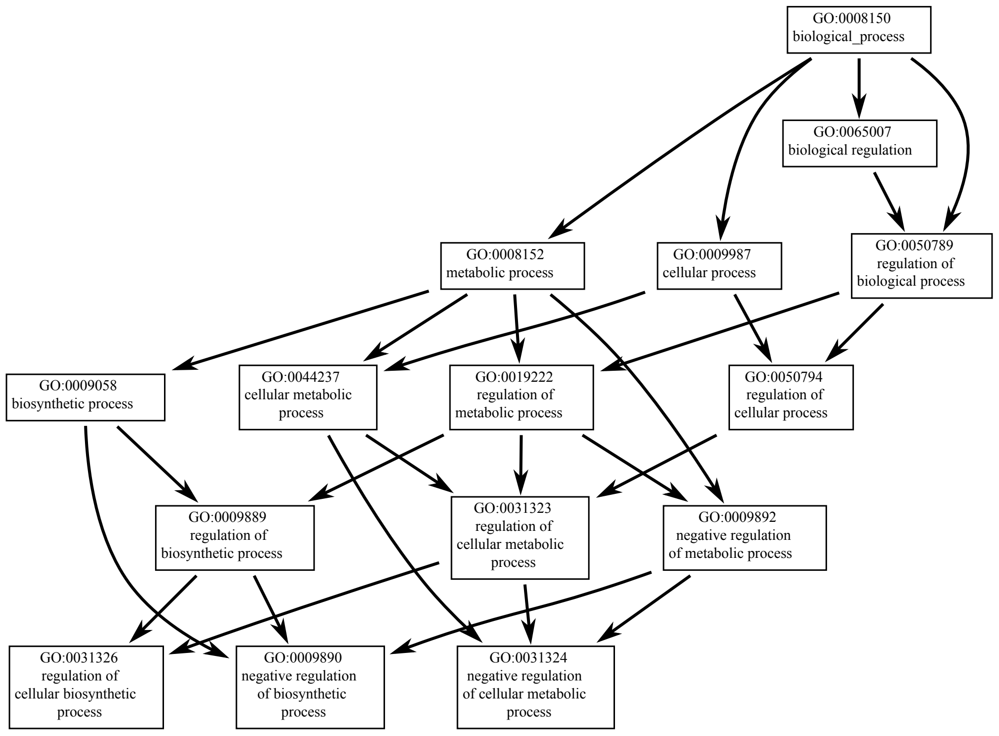
GO:0008150
biological_process
GO:0065007
biological regulation
GO:0008152
metabolic process
GO:0009987
cellular process
GO:0050789
regulation of biological process
GO:0009058
biosynthetic process
GO:0044237
cellular metabolic process
GO:0019222
regulation of metabolic process
GO:0050794
regulation of cellular process
GO:0009889
regulation of biosynthetic process
GO:0031323
regulation of cellular metabolic process
GO:0009892
negative regulation of metabolic process
GO:0031326
regulation of cellular biosynthetic process
GO:0009890
negative regulation of biosynthetic process
GO:0031324
negative regulation of cellular metabolic process

Figure 1: Part of the Gene Ontology Hierarchical Taxonomy. (Adapted from Ashburner et. al. [2])

where each term corresponds to a protein function. The GO taxonomy is organized as a Directed Acyclic Graph (DAG). It is formed by a set of three ontologies, each one covering a different domain. The domais covered are *cellular components*, *biological processes*, and *molecular functions* [2], each having thousands of classes. Figure 1 illustrates a small part of the GO taxonomy. By having thousands of classes, the protein function prediction task using the GO taxonomy is a very challenging problem. The prediction of the terms located in the deepest levels of the hierarchy is very difficult, since they have very few positive instances. In addition, an instance can be classified simultaneously in many paths of the hierarchy. When an instance is classified into a class c_j, it is classified into all superclasses of class c_j. This is the multiple inheritance interpretation, and is the correct interpretation when working with the Gene Ontology [2].

Two approaches have been used to deal with HMC problems, called local and global approaches. The local approach trains a set of classifiers, where each one is responsible for the prediction of one class or a small set of classes which are neighbors in the hierarchy. A classification for a new instance is then obtained combining the predictions provided by the individual classifiers. Any conventional classifier can be used in the local approach. In the global approach, only one classifier is trained to deal with all classes at the same time, and then the classification of a new instance is performed in just one step. Differently from the local approach, a conventional classifier cannot be used, unless adaptations are made to it in order to cope with the multi-label and hierarchical constraints [9].

In this paper, we propose a Genetic Algorithm (GA) to generate hierarchical multi-label classification rules. The method is called **R**elational **H**ierarchical **M**ulti-Label **C**lassification with a **G**enetic **A**lgorithm (RHMC-GA). It is a extended version of the method proposed in Cerri et. al. [5]. We use a new fitness function and selection operator to evolve antecedents of classification rules. The consequents of the rules are obtained using a deterministic procedure. They are represented as a class vector $\bar{\mathbf{v}}$, where each position corresponds to a class, and receives a real value interpreted as the probability of an instance being classified in the class. In addition, we evolve rules with two different kinds of tests. The first one traditionally evaluates if an attribute value A_k satisfies a test condition, *e.g.* $A_k \leq x_{i,k}$. These tests are known as propositional tests. Besides evolving rules with only propositional tests, RHMC-GA evolves rules that also compare the values of different attributes, *e.g.* $A_1 \leq A_2$. These tests are called relational tests [8].

The induction of rules containing relational tests comparing the values of two attributes is the main contribution of this paper. There are many algorithms to discover HMC rules, but none of them induce rules with relational tests. The motivation for inducing relational tests is the increasing of the expressiveness power of the rules. Also, these tests cope better with attribute interactions. However, they increase a lot the search space, which can make it more difficult to discover good rules.

The next sections are organized as follows. Section 2 reviews some works that generate HMC rules for protein function prediction; Section 3 presents our method for generating HMC rules; the experimental set up is presented in Section 4; Section 5 presents the experiments performed; finally, conclusions and future directions are presented in Section 6.

2. RELATED WORK

Not so many works proposed methods to induce HMC rules for protein and gene function prediction in the Gene Ontology taxonomy. This section discusses some of them.

In Vens et al. [12], three methods based on the concept of Predictive Clustering Trees (PCT) were investigated. The authors proposed a global Clus-HMC method that induces a single decision tree to cope with the entire classification problem. They compared its performance with two local methods. The first method, 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. Still based on PCT, the study of Schietgat et al. [11] used an ensemble technique to combine the decision trees induced by Clus-HMC.

Alves et al. [1] proposed a global method using Artificial Immune Systems (AIS) for the generation of HMC rules. The method is divided into two basic procedures: Sequential Covering (SC) and Rule Evolution (RE). 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.

In the work of Otero et al. [9], the authors proposed a global method using Ant Colony Optimization (ACO). The method discovers classification rules in the format "IF ... THEN ...", where an ACO algorithm is employed to optimize the antecedents of the rules. Basically, a sequential instance covering procedure is applied to create classification rules that cover all (or almost all) training instances. The method is initialized with an empty set of rules, and a new rule is added to the set while the number of instances not covered by any rule is higher than a given threshold.

A problem transformation method was proposed in the work of Bi and Kwok [3]. Initially, the classes of the instances are projected to a space of smaller dimension ($\mathcal{R}^m$), using kernel principal component analysis. After that, m regressors are trained. To preserve the hierarchical relationships during the projection procedure, the hierarchical constraints were incorporated to the kernel function.

A KNN-based method was proposed by Pugelj and Džeroski [10]. The authors modified the original KNN algorithm to calculate the prototype of the k-nearest neighbours of an instance. This prototype is the final classification of the instance, just like done in the PCT-based methods [12].

In the experiments performed in this work, we used the global-based method Clus-HMC and its local variants Clus-HMC and Clus-SC. These methods were proposed in [12]. We chose them because they were all applied to the datasets used in our experiments. In addition, they produce the same type of output provided by RHMC-GA and have their code available for downloading. Therefore, we could compare the prediction performances in detail. Recall that none of the above methods learns rules with relational tests.

3. RHMC-GA

Relational Hierarchical Multi-Label Classification with a Genetic Algorithm (RHMC-GA) is a global-based method for the generation of HMC rules using a Genetic Algorithm (GA). The main pseudocode of the method is presented in Algorithm 1, where a sequential covering procedure is implemented to evolve antecedents of rules. In this procedure, instances covered by a rule are removed from the training set, so that new rules can be generated with the remaining instances. The consequent of a rule is generated using a deterministic procedure considering the classes of all instances covered by the rule.

3.1 Individual Representation

Figure 2 illustrates the individual representation in RHMC-GA. Each test of an individual is represented as a 4-tuple {`FLAG`, `OP`, Δ_1, Δ_2}, where each 4-tuple is associated to a dataset attribute A. The gene `FLAG` indicates if the test over an attribute is used in the rule. If the test is used, `FLAG` receives the value 1, and 0 otherwise. Gene `OP` is the integer index of the operator used in the test. Genes Δ_1 and Δ_2 will receive values that will depend on the operators used and in the type of test used (propositional or relational). Exactly how all values are assigned will be detailed explained in the next section.

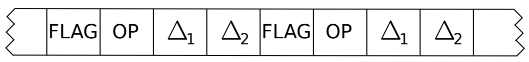

Figure 2: Representation of an Individual.

With the representation depicted in Figure 2, RHMC-GA is able to evolve rules of the form `IF` Antecedent `THEN` Consequent. The antecedent of a rule is thus formed by a conjunction of tests, and the consequent of a rule is formed by a set of GO classes, respecting the constraints of the hierarchical taxonomy. An example of rule is given below. In this example, only active tests of the rule are shown. As can be seen, a rule can be formed by both propositional and relational tests.

$$\texttt{IF (}A_1\ \texttt{OP}\ \Delta\texttt{) AND (}A_3\ \texttt{OP}\ A_5\texttt{) AND (}A_5\ \texttt{OP}\ \Delta\texttt{)}$$
$$\text{THEN}$$
$$\{GO_1, GO_2, GO_4, \ldots, GO_{|C|}\}$$

3.2 Population Initialization

The population in RHMC-GA is initialized using a seeding procedure, where an instance is randomly selected and transformed into a rule. Each test has a probability *pt* of being used. The operator used is randomly selected depending if the attribute is numeric or categoric.

After choosing the operator, the values to be put in the genes Δ_1 and Δ_2 depend on the operator chosen. For categorical attributes, the operators can be $=$, $\neq$, and *in*. The *in* operator verifies if a given attribute value is among a given set of values. If the operator chosen is $=$ or $\neq$, gene Δ_1 receives the index corresponding to the categoric value of the attribute in the instance being used as seed, and gene Δ_2 receives 0 (Δ_2 is not going to be used in the test). If the operator is *in*, gene Δ_1 receives the index corresponding to one of the sets of values which contain the value of attribute in the instance, and gene Δ_2 receives 0. As an example of this last procedure, if the attribute in the instance has the value `A`, and the possible values for this attribute in the dataset are `A`, `B`, and `C`, position Δ_1 receives the value corresponding to one of the sets of values which contain value `A`: {`A, B`}, {`A, C`}, and {`A, B, C`}. The set of values used is randomly chosen.

If the operator chosen corresponds to an operation over a numeric attribute, the assignment of values to genes Δ_1 and

Algorithm 1: A Genetic Algorithm to generate HMC rules.

```
procedure RHMC-GA($D$,$G$,$p$,$minCov$,$maxCov$,$maxNotCov$,$cr$,$mr$,$t$,$e$,$pt$)
Input: training set D
       number of generations G
       size of population p
       minimum number of instances covered by a rule minCov
       maximum number of instances covered by a rule maxCov
       maximum number of not-covered instances maxNotCov
       crossover rate cr
       mutation rate mr
       tournament size t
       number of individuals selected by elitism e
       probability of using a test in a rule pt
Output: set of rules InducedRules
$inducedRules \leftarrow \emptyset$
while $|D| > maxNotCov$ do
    $initialPopulation \leftarrow generatePopulation(D, p, pt)$
    $calculateFitness(initialPopulation, D)$
    $currentPopulation \leftarrow initialPopulation$
    $bestRule \leftarrow$ best rule of $currentPopulation$ according to $fitness$
    $j \leftarrow G$
    repeat
        $newPopulation \leftarrow \emptyset$
        $newPopulation \leftarrow newPopulation \cup elitism(currentPopulation, e)$
        $parental \leftarrow tournamentSelection(initialPopulation, t, e, p)$
        $offspring \leftarrow uniformCrossoverDistance(parental, cr)$
        $newPopulation \leftarrow newPopulation \cup offspring$
        $newPopulation \leftarrow mutation(newPopulation, mr, pt)$
        $newPopulation \leftarrow localOperator(newPopulation, minCov, maxCov)$
        $currentPopulation \leftarrow newPopulation$
        $calculateFitness(currentPopulation, D)$
        $bestRule \leftarrow getBestRule(initialPopulation, bestRule)$
        $j \leftarrow j - 1$
    until $j > 0$ OR $ruleConvergence()$;
    $inducedrules \leftarrow inducedRules \cup bestRule$
    remove from $D$ all instances covered by $bestRule$
return $inducedRules$
```

Δ_2 is simpler, because numeric attribute values do not need to be indexed. In the case of operator $\geq$, gene Δ_1 receives the attribute value in the instance, and gene Δ_2 receives 0. If the operator is $\leq$, gene Δ_2 receives the attribute value in the instance, and gene Δ_1 receives 0. We use Δ_1 and Δ_2 differently depending on the operator used because we consider Δ_1 and Δ_2, respectively, as the lower and upper bounds of the attribute value. This facilitates the use of an operation testing if an attribute value is between two given values ($\Delta_1 \leq A_i \leq \Delta_2$). In this case, the values for genes Δ_1 and Δ_2 are randomly chosen in order to make the attribute value A_i satisfy the test condition.

Up to now, we have explained how the propositional rules are encoded. To encode relational rules, a modification in the above indexation scheme must be done. When comparing two attributes A_1 and A_2, we always use the operator $\leq$. However, we index it with another value, to differentiate it from the $\leq$ operator used in propositional rules. Also, the genes Δ_1 and Δ_2 receive the indexes of the attributes in the instance. Thus, if attributes A_1 and A_2 are being compared, Δ_1 and Δ_2 receive, respectively, the values 1 and 2. Recall that only numeric attributes can be compared.

The indexation of categorical values and operators is done according to Figure 3. In the Figure, a dataset with four attributes is considered. When verifying if a rule covers an instance, appropriate operations are executed according to the type of attribute (numeric and categoric), and also the index of the operation.

With the indexation scheme presented in Figure 3, a rule can be formed by any kind of test, propositional or relational, and there is no specific amount of each one of them in the rule, since the operators are randomly chosen. Still, in relational tests, all possible pairs of attributes can be compared. The indexation scheme also allows different kinds of rules to be generated by a very simple modification in the scheme. If one wishes to generate only propositional rules, he only needs to remove the operator indexed as 3 (relational operator) in the numeric indexation scheme, so that index 3 will never be chosen. On contrary, if only relational rules are desired, one needs to remove operators 0, 1 and 2 from the numeric indexation scheme, so that only index 3 will be chosen.

3.3 Evolution

The evolutionary process starts by saving the best e rules of the current population (elitism). Then, $p - e$ rules are

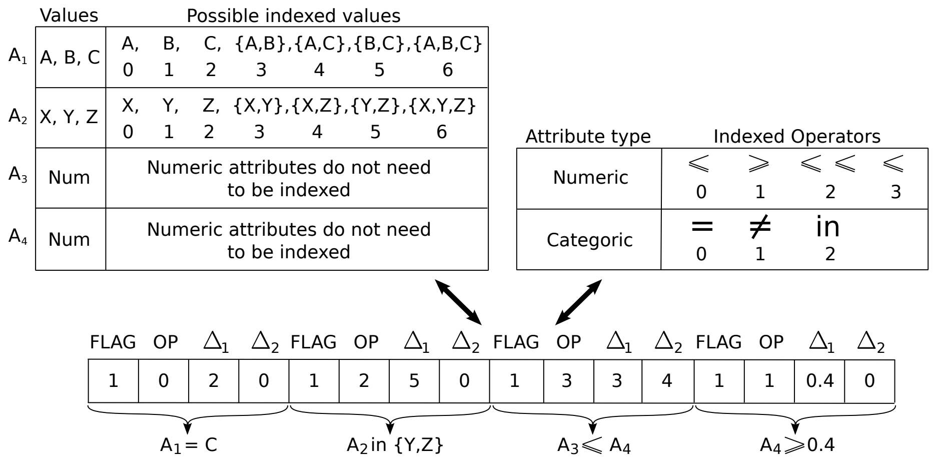

Figure 3: Indexation of operators and categorical values.

selected to be submitted to a uniform crossover operation, in order to generate an offspring. The crossover operation exchanges entires 4-tuples between the individuals. This means that crossover points are allowed to fall only in the boundaries between two 4-tuples. In order to specialize the rules in the classification of a set of instances, the crossover operation considers the distances between the consequents of the rules. The consequent of a rule represents a vector of class probabilities, and each vector position value is given by Equation 1.

$$\overline{v}_{r,j} = \frac{|S_{r,j}|}{|S_r|} \tag{1}$$

In Equation 1, $S_{r,j}$ is the set of all training instances covered by rule r, which are classified in class c_j. The set S_r contains all training instances covered by rule r. Thus, each position $v_{r,j}$ contains the proportion of instances covered by rule r, which are classified in class c_j. This can be interpreted as the probability of an instance covered by r to be classified in class c_j. Since GO terms located at levels closer to the root have more positive instances than GO terms located in deeper levels, positions in $\overline{\mathbf{v}}$ associated to terms closer to the root receive higher probability values than positions associated to deeper terms. Thus, the hierarchical constraint is guaranteed not to be violated.

Our crossover operator receives as input a list of $p-e$ rules. A rule is then removed from the list, and the weighted Euclidean distances between the consequent of this rule, and the consequents of all the other rules in the list, are calculated. The lower the Euclidean distance value between the consequents of two rules, the nearer the rules are considered to be in the search space. The two nearest rules are then removed from the list, and their antecedents are submitted to a uniform crossover to generate two child rules. The objective is to apply the crossover operator in rules that cover instances that are near in the search space, *i.e.*, instances that are classified in a similar or equal set of classes. Equation 2 gives the calculation of the weighted Euclidean distance (WED) between the consequents of two rules.

$$WED(\overline{\mathbf{v}}_1, \overline{\mathbf{v}}_2) = \sqrt{\sum_{j=1}^{|C|} w_i \times (\overline{v}_{1,j} - \overline{v}_{2,j})^2} \tag{2}$$

In Equation 2, w_j corresponds to the weight associated to the j^{th} class in the hierarchy. Weights were associated to each class because, in the context of hierarchical classification, similarities between classes located in levels closer to the root are more important than similarities between classes located in deeper levels [12].

The weighting scheme used in RHMC-GA is the same used in the PCT-based methods [12]. After trying different schemes, the authors found out that the best one is given by Equation 3. The weight w_0 associated to a class in the first level is defined as 0.75, and the weight of a class c_j is recursively defined as the multiplication of w_0 by the mean weight of all its ancestor classes P_j.

$$w_j = w_0 \times \sum_{k=1}^{P_j} w(p_k)/P_j \tag{3}$$

After the generation of new rules, a mutation operator is applied to a percentage mr of them, randomly chosen. Each of the rules have a chance of 50% to suffer a `FLAG` mutation and a chance of 50% to suffer a restriction or generalization.

In the `FLAG` mutation, each test in the antecedent of the rule has a probability pt of being not used (gene `FLAG` exchanged from 1 to 0), or used (gene `FLAG` exchanged from 0 to 1). In the restriction/generalization operation, each used test in the rule is randomly restricted or generalized, having their values modified by using a randomly generated factor in $[0, 1]$. The restriction/generalization procedure is applied in order to make the tests cover a smaller/larger number of instances.

After the mutation operation, a local operator is applied in order to try to guarantee that the rules cover a minimum and maximum number of instances. This is performed to make the rules not too specific neither too general.

After the generation of a new offspring, the fitness of all rules is calculated, and the best rule is saved. This procedure is executed until the maximum number of generations is reached, or until rule convergence, *i.e.* the best rule remains the same after 10 generations. After this complete evolutionary cycle is performed, the best rule found so far is saved, and its covered instances are removed from the training data. A new population is then generated, and a new evolutionary cycle is executed. This is performed until all, or almost all, training instances are covered.

3.4 Fitness Calculation

RHMC-GA uses the variance gain [9, 12] of a rule as its fitness. The variance gain value is higher for rules which cover a more homogeneous set of instances, *i.e.*, rules that partition the training set in more homogeneous sets. In addition, the variance gain can directly cope with hierarchical multi-label data, considering the relationships between the classes [9]. The variance gain (VG) calculation is presented in Equation 4.

$$VG(r,S) = var(S) - \frac{|S_r|}{|S|} \times var(S_r) - \frac{|S_{\neg r}|}{|S|} \times var(S_{\neg r}) \quad (4)$$

As observed in Equation 4, the set S of all training instances is divided into two subsets: the set of instances covered by rule r, denoted S_r, and the set of instances not covered by rule r, denoted $S_{\neg r}$. The variance gain of a rule is obtained considering the set S, and also involves the variance (var) of the sets S_r and $S_{\neg r}$. The variance of a set of instances is defined by the sum of the mean quadratic distances between the class vector of each instance ($\mathbf{v}_i$), and the mean class label vector of all instances in the set ($\overline{\mathbf{v}}$). The variance of a set of instances S is presented in Equation 5. The distance used is the weighed Euclidean distance presented in Equation 2.

$$var(S) = \frac{\sum_{i=1}^{|S|} WED(\mathbf{v}_i, \overline{\mathbf{v}})^2}{|S|} \quad (5)$$

4. EXPERIMENTAL SET UP

In this section we present the datasets, evaluation measures, and RHMC-GA parameters used in the experiments.

4.1 Datasets

We used ten datasets where protein functions are structured according to the Gene Ontology. Details about the datasets can be found in Vens et. al. [12]. Table 1 presents the characteristics of the datasets. As can be seen, we are dealing with thousands of classes in each dataset, and all instances are assigned to more than one class simultaneously. This is, therefore, a very difficult classification task.

The datasets are available at `http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets.html`, and are related to issues like phenotype data and gene expression levels.

4.2 Evaluation Measure

As explained in Section 3, the final classification of an instance is given by a vector $\overline{\mathbf{v}}$, where each position $\overline{v}_j$ corresponds to the probability of the instance being classified in class c_j. This is also true for the PCT-based methods. Thus, a threshold value must be employed in order to obtain the final prediction for an instance. If the corresponding output value $\overline{v}_j$ for a class c_j is equal to or larger than the threshold, the instance is classified into c_j. Otherwise, it is not classified into c_j. The final prediction for an instance is then given by a vector $\mathbf{v}$, where $v_j = 1$ is the instance is classified into c_j, and 0 otherwise.

The question that arises now is how to choose the optimal threshold value in order to obtain the best classification. This is a difficult task, since low threshold values lead to many classes being assigned to each instance, resulting in high recall and low precision. On the other hand, large threshold values lead to very few instances being classified, resulting in high precision and low recall.

In order to avoid the choice of a threshold, we used precision-recall curves (PR-curves) to compare the different methods. To obtain a PR-curve, different thresholds between [0,1] are applied, and thus different values of precision and recall are obtained. Each threshold then represents a point within the PR-space. We connect these points to form a PR-curve, and the area under the curve is calculated. Different methods can be compared based on their areas under the PR-curves.

In this work, we used the area under the average PR-curve ($AU(\overline{PRC})$). Given a threshold value, a precision-recall point ($\overline{Prec}, \overline{Rec}$) in the PR-space can be obtained through Equations (6) and (7). In these equations, i iterates over all classes of the hierarchy.

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad (6) \qquad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \quad (7)$$

To verify the significance of the results, we employed the non-parametrical tests Friedman and Nemenyi [7]. We adopted a confidence level of 95% in the statistical tests. We used the same data partitions suggested in [12]: 2/3 of each dataset were used for inducing the rules and 1/3 for test.

4.3 RHMC-GA Parameters

The parameter values used in RHMC-GA are listed in Table 2. These parameters were obtained based on the work of Carvalho et. al. [4], and no attempts to optimize them were made. The work developed in [4] is a local-based GA to evolve rules for hierarchical, but not multi-label, problems.

The parameter *pt* (probability of using an attribute in initialization) has a different value according to the number of attributes in the dataset. It is given by $|A| \times pt = 5$, where $|A|$ is the number of attributes. Thus, the parameter value is set in order to activate on average 5 tests in the rule. We chose to start with small rules in order not to have an initial population with too many restricted rules, covering none or very few instances. This same logic is used when applying the mutation operator, with probability *pt*.

5. RESULTS AND DISCUSSION

In Table 3, we present the experiments performed using the two different types of rules induced by RHMC-GA: rules using only propositional tests and rules using propositional and relational tests.

To generate rules with relational tests, we used only the datasets whose attributes are microarray data, because it only makes sense to compare attributes of the same domain. We standardized the attributes in these datasets (mean equals 0 and standard deviation equals 1). We also executed the other methods in the standardized datasets.

The values showed in Table 3 for RHMC-GA are the mean and standard deviations after ten executions. As Clus-HMC, Clus-HSC and Clus-SC are deterministic methods, only one execution is needed. As can be seen, the $AU(\overline{PRC})$ values obtained by all methods are very low. This is considered a normal result in HMC domains, specially in the GO, where

Table 1: Summary of datasets: number of attributes ($|A|$), number of classes ($|C|$), total number of instances (Total) and number of multi-label instances (Multi).

Dataset	$\|A\|$	$\|C\|$	Training		Valid		Test	
			Total	Multi	Total	Multi	Total	Multi
Cellcycle	77	4122	1625	1625	848	848	1278	1278
Church	27	4122	1627	1627	844	844	1278	1278
Derisi	63	4116	1605	1605	842	842	1272	1272
Eisen	79	3570	1055	1055	528	528	835	835
Expr	551	4128	1636	1636	849	849	1288	1288
Gasch1	173	4122	1631	1631	846	846	1281	1281
Gasch2	52	4128	1636	1636	849	849	1288	1288
Pheno	69	3124	653	653	352	352	581	581
Seq	478	4130	1692	1692	876	876	1332	1332
Spo	80	4116	1597	1597	837	837	1263	1263

Table 2: Parameters used in RHMC-GA

Parameters	Values
Size of population (p)	100
Elitism rate (e)	1%
Mutation rate (mr)	40%
Crossover rate (cr)	90%
Probability of using an attribute in initialization (pt)	$\|A\| \times pt = 5$
Number of Generations (G)	100
Tournament size (t)	2
Maximum number of not-covered instances ($maxNotCov$)	10
Minimum number of instances covered by a rule ($minCov$)	10
Maximum number of instances covered by a rule ($maxCov$)	300

thousands of classes are involved, making the classification problem much more challenging.

According to Table 3, when evolving classification rules with only propositional tests, RHMC-GA obtained better results than Clus-SC in all datasets, and also outperformed Clus-HSC in some datasets. In comparison with Clus-HMC, RHMC-GA obtained competitive results in some datasets, but was, in general, outperformed.

When comparing the results using the classification rules with both propositional and relational tests, RHMC-GA still obtained better performances than Clus-SC. However, it was outperformed by Clus-HMC and Clus-SC. We can also see that the standardization of data seemed to harm the performances of the PCT-based methods, specially the local versions Clus-HSC and Clus-SC.

We can observe that the $AU(\overline{PRC})$ values obtained by RHMC-GA, when evolving these new kind of rules, were inferior to the values obtained when evolving rules with only propositional tests. This can be explained by the fact that, when comparing different attributes among themselves, the search space is largely increased. This potentially augments the possibility of finding good solutions, but also can make the classification problem more difficult, since it is more difficult to find good solutions in larger search spaces.

The fitness function employed also has a characteristic that may have harmed the RHMC-GA performance in some situations. According to Equation 4, the variance gain of a rule is maximized when the difference between the variance of S_r and $S_{\neg r}$ is minimized. However, if a very homogeneous set of training instances (instances classified in the same, or in a very similar, set or classes) is left to be covered, the fitness value can be reduced to 0. This happens when a rule which covers all instances is induced. In this case, it is a very good rule, since it covers all remaining instances belonging to a same/similar set of classes. However, its fitness value will be 0, since $\frac{|S_{\neg r}|}{|S|} \times var(S_{\neg r})$ will be 0, and the values of $var(S)$ and $\frac{|S_r|}{|S|} \times var(S_r)$ will be the same.

The individual representation is another characteristic that may have harmed the RHMC-GA performance when using relational tests. Because each 4-tuple test is associated to an attribute, when an attribute A_i associated to a given 4-tuple position i is used in a relational test, it cannot be used anymore in a propositional test.

In Table 4, we show the results of the statistical tests applied. We mark with an $*$ the comparisons where the classifier in the column obtained statistically better results than the classifier in the row.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed RHMC-GA, a Genetic Algorithm to generate propositional and relational Hierarchical Multi-Label Classification Rules. We used datasets structured according to the Gene Ontology taxonomy, a directed acyclic graph (DAG) hierarchy having thousands of classes.

RHMC-GA uses a sequential covering procedure to evolve the antecedents of the rules. The consequents are deterministically obtained using the classes of the instances covered by the rules. It also tries to evolve rules specialized in the classification of a group of classes, by applying the crossover operation in rules considered near in the search space.

We showed how the RHMC-GA fitness function could have harmed the algorithm's performance in some situations. Also, we argued that the new kind of generated rules may have largely increased the search space, which can make more difficult to find good solutions.

According to the experiments, RHMC-GA obtained competitive results if compared with other methods in the literature. Although RHMC-GA was outperformed by the state-of-the-art Clus-HMC, the difference in their predictive performance was not statistically significant.

As future work, we plan to develop a better fitness function, which tries to verify if the classes of the instances were

Table 3: $AU(\overline{PRC})$ values obtained

Dataset	Only propositional tests				Dataset	Propositional and relational tests			
	RHMC-GA	Clus-HMC	Clus-HSC	Clus-SC		RHMC-GA	Clus-HMC	Clus-HSC	Clus-SC
Cellcycle	0.341 ± 0.009	0.357	0.371	0.252	Cellcycle	0.319 ± 0.016	0.358	0.344	0.243
Church	0.341 ± 0.005	0.348	0.397	0.289	Church	0.329 ± 0.008	0.340	0.374	0.267
Derisi	0.344 ± 0.003	0.355	0.349	0.218	Derisi	0.298 ± 0.015	0.353	0.346	0.208
Eisen	0.380 ± 0.004	0.380	0.365	0.270	Eisen	0.351 ± 0.017	0.389	0.338	0.271
Gasch1	0.361 ± 0.003	0.371	0.351	0.239	Gasch1	0.335 ± 0.017	0.370	0.325	0.236
Gasch2	0.351 ± 0.007	0.365	0.378	0.267	Gasch2	0.322 ± 0.022	0.369	0.352	0.266
Pheno	0.331 ± 0.002	0.337	0.416	0.316	-	-	-	-	-
Spo	0.345 ± 0.008	0.352	0.371	0.213	Spo	0.295 ± 0.028	0.351	0.315	0.214
Expr	0.361 ± 0.007	0.368	0.351	0.249	Expr	0.327 ± 0.016	0.370	0.334	0.251
Seq	0.359 ± 0.007	0.386	0.282	0.197	-	-	-	-	-

Table 4: Statistical Significance Analysis

	Only propositional tests					Propositional and relational tests			
	RHMC-GA	Clus-HMC	Clus-HSC	Clus-SC		RHMC-GA	Clus-HMC	Clus-HSC	Clus-SC
RHMC-GA					RHMC-GA				
Clus-HMC					Clus-HMC				
Clus-HSC					Clus-HSC				
Clus-SC		*	*		Clus-SC		*	*	

correctly predicted. The current fitness function only considers the variance of the sets of instances covered and not covered by the rules, but does not verify if the instances are being correctly classified. We also intent to investigate a way of evolving the consequents of the rules, and not only the antecedents. In this way, we will not only try find good antecedents, but antecedents with good consequents.

7. ACKNOWLEDGMENTS

The authors would like to thank the Brazilian research agencies FAPESP and CNPq. We would also like to thank Dr. Leander Schietgat and Dr. Celine Vens for providing support with the PCT-based methods. Ricardo Cerri is funded by grant 2009/17401-2, São Paulo Research Foundation (FAPESP).

8. REFERENCES

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

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25(1):25–29, May 2000.

[3] W. Bi and J. Kwok. Multi-Label Classification on Tree- and DAG-Structured Hierarchies. In L. Getoor and T. Scheffer, editors, *International Conference on Machine Learning*, ICML'11, pages 17–24, New York, NY, USA, June 2011. ACM.

[4] R. Carvalho, G. Brunoro, and G. Pappa. Hcga: A genetic algorithm for hierarchical classification. In *IEEE Congress on Evolutionary Computation*, pages 933–940, June 2011.

[5] 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*, SAC '12, pages 250–255, New York, NY, USA, 2012. ACM.

[6] E. P. Costa, A. C. Lorena, A. C. P. L. F. Carvalho, and A. A. Freitas. Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In *Brazilian Symposium on Bioinformatics*, volume 5167 of *Lecture Notes in Bioinformatics*, pages 35–46. Springer-Verlag, 2008.

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

[8] S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer, 2001.

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

[10] M. Pugelj and S. Džeroski. Predicting structured outputs k-nearest neighbours method. In *Proceedings of the 14th international conference on Discovery science*, pages 262–276, Berlin, Heidelberg, 2011. Springer-Verlag.

[11] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Dzeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11:2, 2010.

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