Knowledge Discovery with Artificial Immune Systems for Hierarchical Multi-label Classification of Protein Functions

R. T. Alves, M. R. Delgado and A. A. Freitas

Abstract— **This work presents a system for knowledge discovery from protein databases, based on an Artificial Immune System. The discovered rules have the advantage of representing comprehensible knowledge to biologist users. This task leads to a very challenging problem since a protein can be assigned multiple classes (functions or Gene Ontology (GO) terms) across several levels of the GO's term hierarchy. To solve this problem we present two versions of an algorithm called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System), which is a sophisticated classification algorithm tailored to both multi-label and hierarchical classification. The first version of MHC-AIS builds a global classifier to predict all classes in the dataset, whilst the second version builds a local classifier to predict each class. The proposed versions and an algorithm chosen for comparison are evaluated on a protein dataset, and the results show that MHC-AIS outperformed the compared algorithm in general.**

I. Introduction

The field of data mining has attracted the attention of researchers in different areas [1], [2]. This is due to the fact that the volume of data stored in databases continues to become larger and larger. Hence, the manual analysis of such databases is in general infeasible, and data mining methods are often necessary to extract knowledge from data in a (partially-)automated fashion. The need for data mining is clear in biology, where the amount of data available in biological databases (such as protein databases) keeps increasing very fast.

Arguably, the main goal of data mining is to extract knowledge from data in order to support some human decision-making process. Among the several tasks (types of problems) addressed by data mining, this paper focuses on the classification task. This task essentially consists of using algorithms - usually derived from machine learning or multivariate statistics - to build classification models that are able to predict the class of an example (data instance, record) based on the values of predictor attributes describing that example. The criterion most used to evaluate the performance of a classification algorithm is its predictive accuracy, i.e. a measure of its generalization ability. However, another important criterion in many applications, including the bioinformatics application addressed in this paper (to be described below) is the simplicity, or comprehensibility, of the discovered knowledge [3]. In other words, it is desirable that the classification model be expressed in a representation easily interpretable by the user. One type of representation which usually can be intuitively interpreted by the user consists of rules of the form: IF (antecedent) THEN (consequent), where the antecedent typically consists of a conjunction of conditions on attribute values and the consequent consists of a class(es) to be predicted for the examples that satisfy the rule's antecedent.

This paper presents a new classification algorithm based on the paradigm of Artificial Immune System (AIS). The immune system as a biological complex adaptive system has provided inspiration for a range of innovative problem solving techniques [4], including techniques for classification [5]. The proposed AIS algorithm combines the adaptive global search of the AIS paradigm with advanced concepts and methods of data mining (hierarchical and multi-label classification), in order to solve a challenging bioinformatics problem (protein function prediction). By hierarchical classification it is meant that the classes to be predicted are arranged into a hierarchy (unlike the conventional flat classification problem), and by multi-label it is meant that multiple classes can be assigned to a single example (unlike the conventional single-label classification problem).

Bioinformatics is an inter-disciplinary field, involving the areas of computer science, mathematics, biology, etc [6]. Among many bioinformatics problems, this paper focuses on the prediction of protein functions from information associated with the protein's primary sequence (i.e., its sequence of amino acids). As proteins often have multiple functions which are described hierarchically, the use of multi-label hierarchical techniques for the induction of classification models in Bioinformatics is a promising research area. At present, the biological functions that can be performed by proteins are defined in a structured, standardized dictionary of terms called the Gene Ontology (GO) [7]. The GO consists of a dictionary that defines gene products independent from species. GO actually consists of 3 separate "domains" (very different types of GO terms): molecular function, biological process and cellular component. The GO is structurally organized in the form of a direct acyclic graph (DAG), where each GO term represents a node of the hierarchical structure.

The proposed AIS discovers classification rules for a hierarchical and multi-label classification problem, in the context of protein function prediction, where the classes to

R. T. Alves is with Instituto Federal de Educação, Ciência e Tecnologia do Paraná, Campus Paranaguá, Laboratório de Computação, IFPR, Rua Antonio Carlos Rodrigues, 453, Porto Seguro, CEP: 83215-750, Paranaguá, PR, Brazil (e-mail: roberto.t.alves@gmail.com).

M. R. Delgado is with Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, UFTPR , Av. Sete de Setembro, 3165, CEP: 80230-901, Curitiba, PR, Brazil (corresponding author phone: +55 41 33104688; fax: +55 41 33104683; e-mail: myriamdelg@utfpr.edu.br).

A. A. Freitas is with the Computing Laboratory and Centre for BioMedical Informatics, University of Kent , CT2 7NF, Canterbury, U.K (e-mail: A.A.Freitas@kent.ac.uk).

be predicted are hierarchically-related GO terms and multiple GO terms can be assigned to a single protein (example, data instance). The use of AIS to discover classification rules was previously investigated by the authors in [8]. In that work we used an AIS to discover a set of rules for non-hierarchical single-label classification problems. The reason why the fuzzy theory was not considered here is that the tackled multi-label classification data set consists of only two continuous features and 38 binary ones.

The AIS presented in this paper is based on our previous work [9], but it extends that work with several new procedures (described in Section II). In addition, it discovers knowledge interpretable by the user, in the aforementioned form of IF-THEN classification rules, unlike many other methods proposed in the literature, whose classification model is typically a "black box" which normally does not provide any insight to the user about interesting hidden relationships in the data [1]. The proposed AIS is evaluated mainly with respect to predictive accuracy, but the discovered rules are also evaluated with respect to their simplicity (size of the rule set built by the algorithm).

II. Multi-label Hierarchical Classification with an Artificial Immune System

The AIS algorithm used in this paper is called Multi-label Hierarchical Classification with an Artificial Immune System (MHC-AIS). As discussed in [9], MHC-AIS is based on the following natural immunology principles: clonal selection, immune network and somatic hypermutation [10], [11].

The training phase of MHC-AIS is performed by two major procedures, called Sequential Covering (SC) and Rule Evolution (RE) procedures, which will be detailed in the next sections. MHC-AIS can be considered the first AIS algorithm for multi-label hierarchical classification using such procedure.

Two versions of MHC-AIS are being proposed: the first version of the MHC-AIS builds a global classifier to predict all classes in the application domain, whilst the second version builds a local classifier to predict each class. The next Sections detail each version of the algorithm.

A. Global Version

Each antibody ab_j in the MHC-AIS represents an IF-THEN rule. The IF part is composed by a set given by:

$$\mathbf{z}_j = \langle z_{j1}, z_{j2}, \cdots, z_{jd}, \cdots, z_{j|\mathcal{D}|} \rangle, \text{for } 1 \le d \le |\mathcal{D}|;$$

where d is the d-th condition encoded in ab_j and each d is associated with a predictive attribute of the domain $\mathcal{D}$. Every condition is composed by a triple $z_{jd} = \langle OP_d^j, V_d^j, B_d^j \rangle$, with OP_d: $(=)$ or $(\neq)$ for categorical attributes and $(\geq)$ or $(<)$ for continuous attributes; V_d is a possible valuer for attribute $d \in \mathcal{D}$; B_d = 0 or B_d = 1, indicating if the condition will or will not be used by the rule to classify the examples. The inclusion of B_d is to turn inactive the condition whenever it is necessary.

The rule consequent in the global MHC-AIS is by the following set:

$$Y_j = \{y_j^1, y_j^2, \cdots, y_j^q, \cdots, y_j^m\}, \text{for } Y_j \subseteq \mathcal{C};$$

where $\mathcal{C}$ is the class domain to be predicted and m is the total number of classes that can be represented in the rule consequent (see IV-A.1 for details). Recall that, in the multi-label classification process being proposed, the rule consequent classes mean Gene Ontology (GO) terms.

As discussed before, the training phase of MHC-AIS is performed by two major procedures, called Sequential Covering (SC) and Rule Evolution (RE) procedures. The high-level description of the SC procedure is shown in Pseudo-code 1.

```
Input: full protein training set;
Output: set of discovered rules;
DiscoveredRuleSet = $\emptyset$
TS = set of all protein training examples;
TS' = HierarchicalStructure(TS)
TrainSet = TS';
WHILE $|TrainSet| > MaxUncovExamp$;
    BestRule = RULE-EVOLUTION(TrainSet, AIS);
    DiscovRuleSet = DiscovRuleSet $\cup$ BestRule;
    updateCoveredClasses(TrainSet, BestRule)
    TrainSet = remvExWithAllClassesCovered(TrainSet);
END WHILE
Computefitnes(DiscovRuleSet$_{Final}$,TS');
Eliminate, from all rule consequents,
classes with  fitness  $< \delta_{FT}$
```

Pseudo-Code 1: Sequential Covering (SC) procedure

First, SC procedure initializes the set of discovered rules with an empty set and initializes the training set with the set of all original training examples. Next, each example in the training set is extended to contain both the original class and all its ancestral classes in the GO hierarchy (see Appendix - part A - for more details). Thereafter, the algorithm starts a WHILE loop which, at each iteration, calls the Rule Evolution (RE) procedure. The latter receives, as parameters, the current training set and uses AIS algorithm to discover classification rules. The RE procedure returns the best classification rule discovered by the AIS for the current training set. Then the SC procedure adds that rule to the discovered rule set and removes the training examples covered by that rule. The process of removing examples from the training set is discussed in Appendix - part B.

The process is repeated until the size of Training Set drops below a given threshold($MaxUncovExamp$), indicating too few examples in the set. This stop criterion is used to prevent the algorithm from discovering too specific rules (i. e. rules covering too few examples). At the end, the fitness of all the rules are recomputed considering the original training set (with all the proteins) and the classes with fitness lower than a given threshold (δ_{FT}) are eliminated from all consequents (the "THEN parts") of all discovered rules. Computational experiments have shown that these two steps represent an improvement from the original work proposed in [9] since better results concerning accuracy and simplicity of the discovered rules have been achieved.

The high-level description of the RE procedure used to evolve each rule by means of an AIS algorithm is shown in Pseudo-code 2.

```
Input: current TrainSet;
Output: the best evolved rule;
$AG$ = current TrainSet;
$AB_{t=0}$ = Create initial population of
 antibodies at random;
ComputeFitness ($AB_{t=0}$,$AG$);
$CL$ = ProduceClones($ABt = 0$);
$CL^*$ = MutateClones($CL$);
$AB_{t=1} = AB_{t=0} \cup$ CL$^*$
FOR $t = 1$ to Number of Generations
    Computefitness($AB_t$,AG);
    Elitism($AB_t$);
    Pruning($AB_t$);
    LocalSearch($AB_t$);
    Suppresion($AB_t$);
    $CL$ = ProduceClones($AB_t$);
    $CL^*$ = MutateClones($CL$);
   $AB_{t+1} = AB_t \cup CL^*$
END FOR;
Return the antibody with the best fitness
 among all antibodies produced
 in all generations;
```

Pseudo-Code 2: Rule Evolution (RE) procedure based on AIS.

First, the set of antigens (AG) is defined according to the current training set received as a parameter. The initial population of antibodies (candidate IF-THEN classification rules) $AB_{t=0}$ is randomly created, where the consequent of each rule contains all GO classes in the data being mined. After creation of the population AB, the global fitness (quality measure) of each antibody $ab_j^{t=0}$ of the initial population is calculated on the training set (AG), according to Equation 1 (see next subsection) where each example represents an antigen ag_i.

Next, the population AB is submitted to a clonal expansion process giving rise to a population of clones CL. The population of clones undergoes a process of somatic hypermutation just on the IF part of the rule. As will be discussed latter, the mutation rate applied to each clone cl is inversely proportional to the fitness of the antibody ab from which the clone was produced. The population CL^*, which is formed only by clones that underwent some mutation, is then inserted in the population AB.

Thereafter, the AIS starts to evolve the population of antibodies. Once the global fitness of the rule has been calculated for each ab_j in the population, the algorithm executes other procedures: elitism, pruning, local search and suppression of antibodies. Elitism, a mechanism quite common in evolutionary algorithms [12], selects the antibody with the best fitness to be included in the next-iteration population AB_{t+1}. The procedures pruning and local search are applied to the best rule found so far with the objective of producing some improvements concerning simplicity and precision. These procedures represent another improvement (observed in computational experiments) of the proposed new versions of the algorithm, by comparison with the original algorithm proposed in [9]. The suppression procedure, characteristic of AIS based on the immune network theory, removes from AB_t similar antibodies. These processes will also be detailed later.

1) **Computing the fitness of an antibody (rule):** The global fitness of ab_j is computed according to this equation:

$$\text{Fitness}(ab_j) = \frac{1}{nt}\sum_q \text{FitY}(y_j^q) \geq \delta_{FT}; \quad (1)$$

where nt specifies the number of terms y_j^q whose value of *fitness* $\text{FitY}(y_j^q) \geq \delta_{FT}$. Hence, the value of $\text{Fitness}(ab_j)$ represents the average *fitness* of the terms y_j^q whose fitness be greater or equal than threshold $\delta_{FT} \in [0, 1]$.

Note that the global fitness of a rule depends on the individual values of $\text{FitY}(y_j^q)$ for each class present in the rule consequent, where this individual fitness value is given by the F-measure, combining precision (P) and recall (R) values as follows:

$$\text{FitY}(y_j^q) = \frac{(\beta_{FT}^2 + 1) \times P \times R}{\beta_{FT}^2 \times P + R}, \beta_{FT} \in [0, \infty];$$

where

$$P = \frac{VP_{y_j^q}}{VP_{y_j^q} + FP_{y_j^q}}$$

and

$$R = \frac{VP_{y_j^q}}{VP_{y_j^q} + FN_{y_j^q}}$$

Hence, the values of P and R had to be adapted to the context of the hierarchical and multi-label classification task, and these values depend on the confusion matrix computed for each class, indicating the number of correct and wrong classifications associated with the term y_j^q [2], as illustrated in Table I.

TABLE I
CONFUSION MATRIX FOR $y_j^q \in Y_j$ IN THE ANTIBODY ab_j.

		Real Classes	
		y_j^q	$\neg y_j^q$
Predicted Classes	y_j^q	TP[a]	FP[b]
	$\neg y_j^q$	FN[c]	TN[d]

[a] True Positive: $\text{Aff}(ab_j, ag_i) \geq \delta_{AF}$ and $\mathbf{l}_i[q] = 1$;
[b] False Positive: $\text{Aff}(ab_j, ag_i) \geq \delta_{AF}$ and $\mathbf{l}_i[q] = 0$;
[c] False Negative: $\text{Aff}(ab_j, ag_i) < \delta_{AF}$ and $\mathbf{l}_i[q] = 1$.
[d] True Negative: $\text{Aff}(ab_j, a_i') < \delta_{AF}$ and $\mathbf{l}_i[q] = 0$;

In Table I the affinity $\text{Aff}(ab_j, ag_i)$ is calculated as:

$$\text{Aff}(ab_j, ag_i) = \frac{\#SatCond_j^i}{\sum_{\forall d \in \mathcal{D}} B_d^j};$$

where $\#SatCond_j^i$ measures the total of activated conditions in ab_j that were satisfied by the predictive attributes of ag_i. The threshold $\delta_{AF} \in [0, 1]$ is an user-specified parameter. The term $\mathbf{l}_i[q]$ is defined by Equation 5 in IV-A.1.

MHC-AIS maintains a set of consistent hierarchical classifications during the construction of the global classifier. Hence, if the fitness of some ancestral class y_j^{q*} is smaller than the fitness of its descendant class, then the fitness of y_j^q is assigned to its ancestral class y_j^{q*}.

*2) **Cloning and Hypermutating the antibodies:*** In the cloning process, each antibody ab_j produces $\#Cl_j$ clones of itself, where $\#Cl_j$ is proportional to the fitness of ab_j. The number of clones to be produced for each ab_j is defined as $\#Cl_j = \text{Int}(\text{Fitness}(ab_j) \times \#MaxCl \times ClRate), \#Cl_j \geq 1$, where $\#MaxCl$ represents the maximum number of clones which can be generated from ab_j and $ClRate$ is a parameter whose value is calculated at each generation in order to control the size of population AB, stimulating or not the clones generation. The value $ClRate$ is calculated as:

$$ClRate = \begin{cases} HyperClRate & \text{if } |AB| < nIP \\ 0 & \text{if } |AB| > nMaxP \\ 1 - \left(\dfrac{|AB| - nIP}{nMaxP - nIP}\right) & \text{otherwise} \end{cases}$$

where $HyperClRate$, nIP and $nMaxP$ are specified in the beginning of the execution of the algorithm and indicate, respectively, clonal hyper-expansion rate, initial antibody population size and maximum antibody population size. It is important to emphasize that the parameter $nMaxP$ does not represent the maximum size that the antibody population AB can take during the evolution. Rather, it indicates that, if the size of AB is greater than the value of that parameter, the generation of clones proportional to antibody fitness is dissimulated.

As discussed before, the process of somatic hypermutation is applied just to the antecedent of the rule. A mutation rate applied to each clone cl is inversely proportional to the fitness of the antibody ab from which the clone was produced. Such rate is determined by :

$$MtRt_{cl} = MtMin + (MtMax - MtMin)(1 - \text{Fitness}(cl));$$

where $MtMin$ and $MtMax$ indicate, respectively, the minimum and maximum mutation rates to be applied to a clone cl; and the function Fitness(cl) is presented in Equation 1. The $MtRt_{cl}$ represents the probability that each gene (rule condition in the antecedent) of clone cl will undergo mutation.

*3) **Suppressing Antibodies:*** The suppression procedure removes from the population, antibodies that are similar to each other. This mechanism aims at maintaining the diversity of the immune repertoire. The similarity between antibodies is computed as follows:

$$\text{Similarity}(ab_j, ab_{j'}) = \frac{\#CondIg}{\#MaxCondAt_\alpha}, \text{for all } ab_j \neq ab_{j'};$$

where $\#MaxCondAt_\alpha = \max\left[\sum_{\forall d \in \mathcal{D}} B_d^j, \sum_{\forall d \in \mathcal{D}} B_d^{j'}\right]$, and

- $\#MaxCondAt_\alpha$ – the maximum of two values, namely the number of active conditions in ab_j and in $ab_{j'}$;
- $\#CondIg$ – the number of active conditions that are equal in ab_j and $ab_{j'}$;
- B_d – a binary flag indicating whether the d-th condition is active or not.

If Similarity$(ab_j, ab_{j'}) > \delta_{SIM} \in [0,1]$, then either ab_j or $ab_{j'}$ must be suppressed (removed) from the population, where δ_{SIM} is a user-defined similarity threshold. The ab to be removed (out of two similar antibodies) is the one with smaller fitness - ties are broken at random.

*4) **Pruning and Local Search:*** In general, the antibody of AB_t with best fitness is selected to undergo pruning - i.e., having its irrelevant rule conditions (if any) removed. The selected antibody ab_j can undergo pruning only if it has at least two active conditions. Once this constraint is satisfied, active conditions are randomly selected from ab_j for the pruning procedure. For each of those conditions, the condition is tentatively removed from the rule antecedent and the fitness of the rule is recalculated. If the new fitness value (without the condition) is greater than or equal to the previous value (with the condition), then the condition is effectively removed from the rule - by changing the value of its flag to inactive. The loop choosing active conditions for potential pruning is repeated while the trials is less than two and the number of active conditions is greater than 1.

If the fitness after the tentative removal of a rule condition is worse than the previous fitness (with the rule condition), another active condition is randomly selected for potential pruning. If the fitness of the rule does not improve after two successive choices of active conditions, then the pruning process is terminated. It should be emphasized that a rule condition is removed only if this does not reduce the fitness of the antibody. A de-activated (pruned) condition can become active again only through the somatic hypermutation mechanism.

The local search procedure (an improvement of the MHC-AIS presented here) aims at performing a fine tuning of the rule antecedent, in order to improve the fitness of the rule. The local search used here is similar to a conventional hill climbing algorithm, which has no memory of previously generated candidate solutions [13]. The local search procedure works as follows. First, it selects the best antibody ab_b in the current population. Next, an active condition of ab_b is randomly chosen to undergo local search, and the current attribute value in that condition is replaced by a randomly chosen value (among the values in the domain of the attribute). Then the fitness of ab_b is re-computed. This process is iteratively performed for other randomly chosen attribute values, again re-computing the fitness of ab_b with each new value. When the fitness does not improve after two consecutive changes of attribute value, the local search process for this condition is terminated, and a new rule condition is randomly chosen to undergo local search as described above. When the fitness does not improve after local search has been applied to two consecutively chosen rule conditions, the local search for ab_b is terminated.

B. Local Version

The antibody of the local version is similar to the antibody of global version described in section II-A. The only difference occurs in the consequent. In the local MHC-AIS the consequent is represented by:

$$y_j^l = \begin{cases} 1 & \text{if the rule predicts the class } l \\ 0 & \text{otherwise} \end{cases};$$

where l represents the class for which the local classifier was trained to predict.

Like the global MHC-AIS, the local MHC-AIS consists of the SC (see Pseudo-code 1) and RE procedures (see Pseudo-code 2) described in Section II-A, but with some differences.

In the local version, a classifier is trained for each node (class) of the GO's DAG. So, the SC procedure re-labels for each class the training examples as positive or negative. Positive examples represent examples associated with the class of the current node of the GO's DAG, denoted class Y, whilst examples that do not have the class Y are labeled as negative examples. MHC-AIS is an algorithm for constructing hierarchical classifiers, and therefore the hierarchical structure has to be coped with like in the global version. Hence, all training examples labeled with any descendant class of the current class Y are labeled as positive class.

In this local version, MHC-AIS first discovers as many classification rules as necessary in order to cover the positive examples. Next, the algorithm discovers as many rules as necessary to cover the negative examples. Every time that a given rule is discovered, all the examples correctly covered by that rule (i.e. examples satisfying the conditions in the rule antecedent and having the class predicted by the rule consequent) are removed from the current training set, as usual in rule induction algorithms. This iterative process of rule discovery and removal of training examples is repeated until the number of examples in the current training set becomes smaller than a user-defined threshold $MaxUncovExamp$. The other procedures of the local MHC-AIS are the same as in the global version of the algorithm, described in II-A.

III. Experiments and Results

This section describes the experiments performed to compare the two proposed versions of MHC-AIS with a traditional method to solve classification problems.

A. Compared Methods

The two versions of the MHC-AIS are compared with the PART[1] algorithm. The Partial Decision Tree algorithm (PART) was proposed by Frank and Witten [14] and it builds classifiers consisting of rules of the form IF <*antecedent*> THEN <*consequent*> from "partial" decision trees – see [14] for details. In the context of this work PART, builds local (binary) classifiers, so that a classifier is generated for each class (GO term).

Table II presents the values used in the experiments for each parameter of MHC-AIS in the global and local versions.

B. The Protein Data Base Considered

All the compared methods (MHC-AIS - local and global, and Part) were evaluated on a dataset of proteins created from information extracted from the well-known UNIPROT database [15]. This dataset contains two protein families: DNA-binding proteins (which are involved in gene expression as transcription activators) and ATPase proteins (which are enzymes that catalyze the hydrolysis of ATP and as a result release energy that is used by the cell) [16]. Both types of protein families consist of a large number of proteins, with a correspondingly large number of associated classes (GO terms), leading to challenging hierarchical and multi-label classification problems. The dataset used in the experiments contains 7877 proteins, where each protein (example) is described by 40 predictor attributes, 38 of which are PROSITE1 patterns (a well-known type of protein motif or signature) and 2 of which are continuous attributes (molecular weight and the number of amino acids in the primary sequence). Each of the 38 attributes representing PROSITE1 patterns are binary attributes, indicating whether or not the protein contains the corresponding pattern. In total, the dataset contains 214 classes (GO terms) to be predicted.

TABLE II
Parameters used by the two MHC-AIS versions.

Parameter	Definition	Global	Local
δ_{AF}	*matching* threshold	$\{0.8, 0.9, 1.0\}$	
δ_{FT}	*fitness* threshold	0.9	-
δ_{SIM}	Similarity for *ab*'s	0.7	0.7
β_{FT}	parameter of *f-measure*	0.05	1
$MaxUncovExamp$	number of non-covered examples in the train. set	10	10
$HyperClRate$	hypermutation rate	2.0	2.0
$MtMin$	min of mutation rate	0.01	0.01
$MtMax$	max of mutation rate	0.5	0.5
$\#MaxCl$	max of number of clones	10	10
$\#MaxIter$	itert per evolut period	50	50
nIP	size of initial pop	100	100
$nMaxP$	max size of pop	500	500

C. Predictive Accuracy and Simplicity in the Test Set

As previously discussed, in data mining the discovered knowledge should be not only accurate, but also comprehensible to the user [1], [2]. In this spirit, the results can be evaluated according to two criteria, viz. the predictive accuracy and simplicity of the discovered rule set. In this paper, simplicity will be measured in terms of the size of the discovered rule set, an approach which is not ideal but is still used in the literature. The predictive accuracy is evaluated by the F-measure (adapted to the scenario of multi-label hierarchical classification), which involves computing the precision and recall of the discovered rule set on the test set (unseen during training).

In the global version, the set of GO terms predicted for a test example t, denoted $PredGO(t)$, consists of the union of all GO terms in the consequent of all rules covering t - i.e. all rules ab_j whose conditions are satisfied by t's attribute values ($\text{Aff}(ab_j, t) \geq \delta_{AF}$).

In the local version of MHC-AIS, each test example t is submitted to the $|T_H|$ trained classifiers (T_H is given by Equation 4). Each classifier consists of a set of discovered rules. The class predicted by each classifier is the class represented in the consequent of the rule with the greatest fitness value (computed during training) out of all

[1]The PART algorithm is a well-known rule induction algorithm included in the freely available data mining tool WEKA: *http://www.cs.waikato.ac.nz/ml/weka/*.

rules discovered by that classifier that cover the example t. Hence, $PredGO(t)$ consists of all GO terms whose trained classifiers predicted their corresponding positive class for the example t.

In both cases (local and global), if no discovered rule covers the example t, the latter is classified by the default rule, which predicts the majority class in the training set.

MHC-AIS computes the hierarchical multi-label Precision and Recall for a test example t - denoted $P(t)$ and $R(t)$, respectively - as per Equations 2 and 3, where $TrueGO(t)$ is the set of true GO terms for example t.

$$P(t) = |PredGO(t) \cap TrueGO(t)|/PredGO(t) \quad (2)$$

$$R(t) = |PredGO(t) \cap TrueGO(t)|/TrueGO(t) \quad (3)$$

Thus, precision is the proportion of true classes among all predicted classes, whilst recall is the proportion of predicted classes among all true classes. The hierarchical multi-label F-measure for a test example t is given by the harmonic mean of $P(t)$ and $R(t)$ as:

$$F(t) = (2 \times P(t) \times R(t))/(1 + P(t) + R(t))$$

Finally, once $P(t)$ and $R(t)$ have been computed for each test example t, the system computes the overall F-measure over the entire test set $\mathbf{T}$ as

$$\text{Predictive Accuracy} = F(\mathbf{T}) = \left(\sum_{t \in \mathbf{T}} F(t)\right) /|\mathbf{T}|$$

where $|\mathbf{T}|$ denotes the cardinality of the test set $\mathbf{T}$.

D. Results

Table III shows the predictive accuracy (precision, recall and F-measure) for the proposed MHC-AIS algorithm (global and local versions) compared with the PART algorithm.

TABLE III
PREDICTIVE ACCURACY OF MHC-AIS VERSUS PART.

MHC-AIS Global			
δ_{AF}	*Precision*	*Recall*	*F-Measure*
0.8	95.36 ± 0.3	79.89 ± 0.4	83.93 ± 0.4
0.9	96.58 ± 0.4	77.86 ± 0.3	83.41 ± 0.3
1.0	96.17 ± 0.2	77.44 ± 0.2	82.92 ± 0.1
MHC-AIS Local			
0.8	90.91 ± 0.2	87.32 ± 0.3	$\mathbf{87.96 \pm 0.2}$
0.9	89.69 ± 0.3	87.13 ± 0.4	87.27 ± 0.3
1.0	84.64 ± 0.2	87.09 ± 0.4	84.63 ± 0.5
PART - Weka			
-	83.08 ± 0.6	81.85 ± 0.6	82.78 ± 0.5

In Table III, the numbers after the $\pm$ symbol represent the standard deviations associated with a well-known 10-fold cross-validation procedure [2]. In the column F-measure, the best result (out of all methods being compared) is shown in bold. Table III shows results for different affinity (matching) threshold δ_{AF} values for both versions of MHC-AIS, to evaluate the predictive performance of the algorithms using partial matching ($\delta_{AF} < 1.0$) or total matching ($\delta_{AF} = 1.0$).

Table III shows that the local MHC-AIS obtained the best results for F-measure with all affinity threshold values. Note that, in both versions of MHC-AIS, as the value of the affinity threshold δ_{AF} increases the value of F-measure is reduced, showing a disadvantage in the use of total matching. These results show that both versions of MHC-AIS outperformed PART in terms of F-measure value, for all values of the δ_{AF} threshold. The result of the Wilcoxon signed rank test (a non-parametric statistical test often used in data mining research) confirmed that the differences in the F-measure values of the local version of MHC-AIS and of PART are statistically significant (with 95% of confidence) for all values of δ_{AF}. When comparing the global version of MHC-AIS with PART, the differences in F-measure values are statistically significant (again, with 95% confidence using the Wilcoxon signed rank test) for $\delta_{AF} = 0.8$, but not for $\delta_{AF} = 0.9$ and $\delta_{AF} = 1.0$. These results confirm that the proposed MHC-AIS is a good alternative to solve our target hierarchical multi-label classification problems.

Table IV shows the results with respect to the simplicity of the discovered rule set. This simplicity was measured by the number of discovered rules and total number of rule conditions (in all rules) – whose values are shown in the first and second columns in the table, respectively. Recall the values reported in Table IV are average values computed by a 10-fold cross-validation procedure.

TABLE IV
RULE SET SIMPLICITY (SIZE) OF MHC-AIS VERSUS PART.

MHC-AIS Global		
δ_{AF}	#Rules	#TCond
0.8	104.3 ± 3.6	909.9 ± 24.2
0.9	64.7 ± 2.2	409.6 ± 17.6
1.0	$\mathbf{46.9 \pm 1.0}$	$\mathbf{122.6 \pm 5.46}$
MHC-AIS Local		
0.8	738.3 ± 4.4	8394.7 ± 59.8
0.9	752.5 ± 4.4	7165.3 ± 82.9
1.0	734.9 ± 7.3	4610.3 ± 51.2
PART - Weka		
	4759.3 ± 12.6	1820.6 ± 6.8

Note that, as shown in Table IV the global MHC-AIS obtained much better results concerning rule set simplicity (i.e. much smaller rule sets) than the local MHC-AIS and PART, in all experiments. This advantage of the global MHC-AIS is due to the fact that it builds a single set of rules predicting all classes in a single run of the algorithm. This allows the classifier to capture some relationships among classes, leading to a more compact classification model. In contrast, local MHC-AIS and PART have to build a rule set for each class, and the size of the entire classification model is given by the union of the rule sets built by all the classifiers, leading to much larger models, probably involving considerable redundancy between rules built by different but related classifiers (e.g. parent and child classifiers). When comparing local MHC-AIS with PART, the former discovered fewer rules, whilst the latter built rule sets with fewer rule conditions. The considerably smaller number of rule conditions discovered by PART is due to the fact that, in many of the local rule sets discovered by PART, the only rule produced by the algorithm was a default rule, i.e, a rule with no conditions in its antecedent, and simply predicting the majority class in the training set for all examples in the test set covered by the rule. This is the reason why the number of rules discovered by PART is greater than the total number of conditions in all the discovered rules.

Table IV also shows that, for each of the two versions of MHC-AIS, the simplest (smallest) rule set is built when $\delta_{AF} = 1.0$. Hence, considering the results shown in Tables III and IV, in summary the use of partial matching leads to higher predictive accuracy, whilst the use of total matching leads to the discovery of a simpler rule set.

An example of a rule discovered by global MHC-AIS in the aforementioned protein data set is presented below:

IF (PS00636 == 1) AND (MOLECULAR-WEIGHT < 54885) THEN (5488, 5515, 31072)

The biological interpretation of this rule is: if a protein presents the Prosite pattern "SJ-protein family domains signature and profiles"and "molecular weight is less than 54885" then the predicted classes (biological functions) are: "binding" (GO term 5488) and "protein binding" (GO term 5515) and "heat shock protein binding" (31072). Note that the GO hierarchy was considered, i.e. the true hierarchical path is 5488 - 5515 - 31072 (from shallower to deeper nodes).

IV. Conclusion

This work presented a new artificial immune system (MHC-AIS) for the difficult problem of hierarchical multi-label classification in data mining, in the context of protein function prediction – where the classes to be predicted are protein functions corresponding to terms in the Gene Ontology (GO). Two versions of the MHC-AIS were proposed, a global version, where a single global classifier is built predicting all classes of the application domain; and a local version, where a local classifier is built for each node of the hierarchical GO classes. Both versions have the advantage of discovering IF-THEN classification rules, constituting a type of knowledge representation that can, in principle, be easily interpretable by biologist users.

The local and global versions were compared with a traditional classification method - the PART algorithm. The results showed that overall the proposed algorithm outperformed PART in the two evaluation criteria considered: predictive accuracy (F-measure) and simplicity (size) of the discovered classification model (rule set). More precisely, in all 3 experiments (with different parameter values for local MHC-AIS) comparing the local MHC-AIS with PART, the local MHC-AIS achieved significantly higher predictive accuracy and significantly fewer rules than PART, although PART discovered significantly smaller rules. Also, in all 3 experiments (with different parameter values for global MHC-AIS) comparing the global MHC-AIS with PART, global MHC-AIS discovered significantly fewer and smaller rules than PART; and in one of those 3 experiments the predictive accuracy obtained by MHC-AIS was significantly higher than the accuracy obtained by PART (with no statistically significant difference in the other two cases). These results suggest that both versions of MHC-AIS are very competitive with PART.

Future work will involve: (a) analyzing the biological relevance of the discovered rules; (b) evaluating the proposed MHC-AIS in datasets of other protein families and with other types of predictor attributes; and (c) comparing the results with other approaches, e.g. CLUS algorithm proposed in [17].

Appendix

A. Applying GO Hierarchical Structure to the Set AG

In biological databases a protein is annotated only with its most specific GO term. Given the semantics of the GO's functional hierarchy, this implicitly means the protein also contains all the functional classes of its ancestral GO terms in the GO's DAG. Hence, in a data preprocessing step, MHC-AIS explicitly assigns to each antigen (protein) both its most specific class(es) (GO term(s)) and all its ancestral classes.

Hence, the hierarchical structure $\mathcal{H}$ of the terms (classes) of the GO is also provided as input to the algorithm. The GO structure is defined as $\mathcal{H} = \langle \mathcal{C}, \preceq \rangle$, where $\mathcal{C}$ represents the set of terms defined in the GO and the relation $\preceq$ determines the hierarchical structure of the GO graph (where each GO term is a node in the graph) in the form of a partially-ordered set of terms.

The total set of classes to be predicted by the classifier is defined by Equation 4.

$$T_H = \left(\bigcup_{\forall ag_i} L_i \right) \bigcup \left(\bigcup_{\substack{\forall l_{ik} \in L_i \\ \forall ag_i}} \text{Ancestors}(l_{ik}) \right) ; \quad (4)$$

where L_i represents the set of classes directly annotated for (associated with) the i-th antigen of the data set, $l_{ik} \in L_i$ the k-th class associated with ag_i and Ancestors(l_{ik}) the set of terms (classes) which are ancestors of l_{ik} with the exception of the root node.

1) MHC-AIS Global: The set of classes associated with the examples ag_i considering the hierarchical structure $\mathcal{H}$ is defined as

$$T_{H_i} = L_i \bigcup \left(\bigcup_{\forall l_{ik} \in L_i} \text{Ancestors}(l_{ik}) \right) ;$$

These classes are represented by a binary vector $\mathbf{l}_i$ of length $m = |T_H|$, where each of those m vector components indicates whether or not the corresponding class is associated with ag_i, as given in Equation 5.

$$\mathbf{l}_i[q] = \begin{cases} 1 & \text{if } l_q \in T_{H_i} \\ 0 & \text{otherwise} \end{cases} ; \quad (5)$$

where l_q represents the label associated with the q-th element of T_H.

So, MHC-AIS also considers the semantics of the GO's functional hierarchy when creating classification rules - i.e., it guarantees that, if a rule predicts a given GO term, all its ancestral GO terms are also predicted by the rule.

2) MHC-AIS Local: Classifiers are built for each GO term (class) $l \in T_H$ of the set of classes to be predicted and its ancestors (Ancestors(l)).

Usually, in order to build local classifiers, hierarchical and multi-label classification problems are transformed into flat single-label ones. In this latter case, each example in the dataset is associated to just one class, but the class hierarchy must be considered in some way. MHC-AIS represents the class hierarchy as follows:

$$l_i = \begin{cases} 1 & \text{if } l_{ik} \equiv l \vee l_{ik} \in \text{Descendants}(l) \\ 0 & \text{else} \end{cases};$$

where l represents the class which the classifier will created for predicting and $l_{ik} \in L_i$ is the k-th class annotated in the i-th example of the dataset. Therefore, when building a classifier to predict a given class l, positive examples are those annotated with class l or their descendants; the other examples are considered negative examples.

B. Removing Examples from the Training Set

In a single-label classification process based on a procedure for sequentially discovering rules from data [2], the removal of examples from the dataset is very simple, as follows. If an example is classified (matching partial or total) by the best rule discovered in iteration t and the example's class is the same as the class present in the rule consequent, then the example is removed from the training set. In multi-label classification based on a procedure for sequentially discovering rules, the process is more complicated, because a discovered rule can predict just some (rather than all) of the classes associated with an example, so that the example cannot be removed based just on that rule.

In the global version of MHC-AIS, every $ag_i \in AG$ is associated with a binary vector $\mathbf{q}_i$ that indicates the classes predicted by the candidate rule set CR up to the current iteration t. The vector $\mathbf{q}_i$ has the same number m of elements as $\mathbf{l}_i$.

Initially, in $t = 0$, the values of the components of $\mathbf{l}_i$ (Equation 5) are assigned to $\mathbf{q}_i$, since no class was predicted yet for ag_i. Hence, $\mathbf{q}_i^{t_0}[q] = \mathbf{l}_i[q], q \mid q = 1, \ldots, m$. For each discovered rule($BestRule$) in t, $\mathbf{q}_i$ is updated for the next iteration $t+1$. This updating is done only for the ag_i that are correctly classified by $BestRule$. An example is said to be correctly classified by a rule if the example satisfies the conditions in the rule antecedent and the example has the class(es) predicted by the rule. This updating is performed as follows:

$$\mathbf{q}_i^{t+1}[q] = \begin{cases} 0 & \text{if FitY}(y_b^q) \geq \delta_{FT} \wedge \mathbf{l}_i[q] = 1 \\ \mathbf{q}_i^t[q] & \text{otherwise} \end{cases},$$

$$\text{for Affinity}(ab_b, ag_i) \geq \delta_{AF};. \tag{6}$$

where $ab_b = BestRule$ and y_b^q is the q^{th} class in the consequent of ab_b.

Once $\mathbf{q}_i$ has been updated, the examples for which all classes have been predicted by CR are removed from AG. The elimination of an example fromAG depends on the total of non-covered classes it has in the consequent, which is calculated as:

$$\text{NcovClass}(ag_i) = \sum_{q=1}^{m} \mathbf{q}_i^{t+1}[q].$$

If NcovClass$(ag_i) = 0$, all classes of ag_i have been covered by CR, and so that example must be removed from AG.

The examples that must remain in the training set AG for the next iteration t are obtained as follows:

$$AG_{t+1} = \{ag_i \in AG_t \mid \text{NcovClass}(ag_i) > 0\}$$

where NcovClass$(ag_i) > 0$ indicates that there are classes still not covered by the classifier.

Acknowledgment

This work was supported in part by the CNPq under grant 307735/2008-7 and Fundação Araucária under grant no.233/8331.

References

[1] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[2] , I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition. 2005.

[3] A. A. Freitas and J. D.C. Wieser and R. Apweiler. "On the importance of comprehensible classification models for protein function prediction". *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, vol. 7(1), 2010, pp. 172-182.

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

[5] A. A. Freitas and J. Timmis. "Revisiting the foundations of artificial immune systems for data mining". *IEEE Trans. on Evolutionary Computation*, vol. 11(4), 2007, pp. 521–540.

[6] G. B. Fogel and D. W. Corne. *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann Publishers, 2003.

[7] . The Gene Ontology Consortium. "The Gene Ontology (GO) Database and Informatics Resource". *Nucleic Acids Research*, vol 32(1), 2004, pp. 258–261.

[8] R. T. Alves and M. R. Delgado and H. S. Lopes and A. A. Freitas "An Artificial Immune System for Fuzzy-Rule Induction in Data Mining". *Lecture Notes in Computer Science*, vol. 3242, 2004, pp. 1011-1020.

[9] R. T. Alves and M. R. Delgado and A. A. Freitas "Multi-Label Hierarchical Classification of Protein Functions with Artificial Immune Systems". *Proc. 3rd Brazilian symposium on Bioinformatics: Advances in Bioinformatics and Computational Biology*, 2008, pp. 1–12.

[10] G. A. Ada and G. V. Nossal. "The Clonal Selection Theory".*Scientific American*, vol 257, 1987, pp 50–57.

[11] N. K. Jerne. "Towards a Network Theory of Immune System". *Ann. Immunol. (Inst. Pasteur)*, vol 125C, 1974, pp. 373–389.

[12] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley Reading, 1989.

[13] , S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[14] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization". *Proc. Proceedings of the 15th International Conference on Machine Learning*, 1998, pp. 144–151.

[15] . The UniProt Consortium. "The Universal Protein Resource (UniProt)". *Nucleic Acids Res.*, vol. 35, 2007, pp D193–D197.

[16] B. Alberts and A. Johnson L. Lewis and M. Raff and K Roberts and P. Water. *Molecular Biology of the Cell*. Garland Science, 4th Edition, 2002.

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