HCGA: A GENETIC ALGORITHM FOR HIERARCHICAL CLASSIFICATION

Rafael V. Carvalho
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
Email: rvieira@dcc.ufmg.br

Gustavo Brunoro
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
Email: brunoro@dcc.ufmg.br

Gisele L. Pappa
Computer Science Department
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
Email: glpappa@dcc.ufmg.br

***Abstract*—Hierarchical classification (HC) is a specialization of the well-known flat classification task. The main difference among them is that in HC examples have to be assigned to classes organized in a previously defined class hierarchy, while in traditional flat classification no class order is imposed. There are two main approaches commonly used to tackle HC: the top-down or local approach, which is classifier independent, and the big-bang or global approach, which usually is the product of a modification of a well-known flat classifier. Although evolutionary algorithms have been successfully applied to flat classification, they are underexplored in HC. In this direction, this paper proposes HCGA (Hierarchical Classification Genetic Algorithm), a method that takes both local and global information into account. HCGA uses a top-down approach for building a classification model and also for classifying new examples. This is in contrast with current top-down methods, which make use of this strategy only for test, using flat classifiers for training models. The method was applied to four GPCR (G protein-coupled receptor) activity datasets, obtaining results statistically equal or better than five baseline classifiers run using a top-down approach.**

Keywords: hierarchical classification, genetic algorithms, protein function prediction

I. INTRODUCTION

The problem of flat classification has been extensively studied in the areas of data mining, machine learning, and pattern recognition, with evolutionary computation being an effective and successful approach for solving the problem [1]. In the past decade, research efforts in these areas turned to the task of hierarchical classification [2]. In contrast with flat classification, which builds models from a set of categories which do not obey any order or structure, hierarchical classification has as its main goal to classify examples following a predefined hierarchy.

When developing methods for hierarchical classification, two points need to be addressed: (i) the structure of the class hierarchy, which can be organized as a tree (single-label classification) or a direct acyclic graph (multi-label classification); and (ii) the expected results for classification, which might either require each example to belong to a leaf node of the hierarchy, or to a class represented by an internal node in any hierarchical level.

Having addressed the aforementioned points, three main approaches can be used to generate hierarchical classification models [2]: (i) turn the hierarchical classification problem into a flat one, (ii) use a local approach, also known as top-down, or (iii) use a global approach, also called big-bang. The first strategy simply ignores the hierarchy, and considers the classes in the leaf nodes as flat classes. The top-down approach uses flat classifiers to generate models for either each node or sibling nodes in the hierarchy, and takes *local hierarchy information* into account when building these models (ending up with a set of models). For each new example, these models are used to predict their classes in a top-down fashion, as detailed later. Finally, the global approach usually modifies a traditional flat classification algorithm in order to add hierarchy information when building the classification model.

Both local and global methods have advantages and drawbacks. Local methods ignore hierarchy information which might degrade the models produced, but at the same time they produce more compact models and are classifier independent. Global methods, in turn, usually build more complex models (but note that here we have a single model, against a set of them produced by local methods) and are classifier-dependent, but deal better with global information [2]. In general, top-down methods are far more used in the literature.

Regarding evolutionary algorithms, to the best of our knowledge, they have not been yet used in the context of hierarchical classification. In order to fill this gap, this paper proposes HCGA (Hierarchical Classification Genetic Algorithm), a genetic algorithm that deals with hierarchical classification. One of the most interesting features of the algorithm, and which cannot be found in any other method proposed so far, is to *use the top-down approach for both building the model as well as classifying new examples*.

Hence, our main goal is to transform the *test* top-down approach from mainly a method for avoiding or correcting inconsistencies in class prediction at different levels [2] into an effective way of *generating models* in a top-down fashion. In order to achieve that, the GA is first used to produce a model for the classes in the first level of the hierarchy. In a second step, these models are used as seeds for evolving models for classes in the second level, and so on.

It is important to point out that hierarchical problems have been explored mainly in the areas of text classification [3][4] and bioinformatics [5][6]. Lately, most research in hierarchical classification is concentrated on the protein function prediction

scenario, which presents many challenges, such as a huge number of classes, very unbalanced datasets, and very few examples to represent the leaf nodes of the class hierarchy. Considering this scenario, this paper deals with the problem of GPCR (G protein-coupled receptor) activity. Identifying these proteins is of extremely importance, once it is believed that 40–50% of current medical drugs target GPCR activity [7]. Results in four version of GPCR datasets showed that the proposed algorithm obtains results statistically equivalent or superior to those obtained by other five baseline algorithms run under the top-down framework.

The remainder of this paper is organized as follows: Section II reviews works of evolutionary algorithms in flat classification, and also presents the main concepts of hierarchical classification. Section III describes HCGA, while Section IV shows experimental results. Finally, Section V draws conclusions and suggests future research directions.

II. Related Work

As pointed out before, evolutionary algorithms (EAs) have been successfully used to solve flat classification problems for quite some time [1][8], but we did not find in the literature any EA specially designed to deal with the hierarchical classification problem. Hence, this section focuses on how things work in flat classification EAs, and how they can be adapted or improved to deal with class hierarchies.

Depending on the type of EA used to solve the flat classification task, different types of representations can be used. For instance, most genetic programming (GP) algorithms generate classification models represented by a mathematical function [9], while both GPs and GAs also commonly built models as a set of if-then rules [10]. We have tested the two aforementioned representations. However, as we tried the method in more complex and high dimensional datasets, runtime became prohibitive. A the size of the search space increased, the GA with more complex representations could not effectively explore it with a feasible number of generations and individuals. Hence, in order to simplify the already complex problem of HC, here we use a third type of representation, first proposed in [11] for flat text classification. This representation is simpler than the aforementioned ones, but has the drawback of being restricted to binary attributes.

Moreover, the representation proposed in [11] corresponds to the most common individual representation in the genetic algorithm community: a vector of binary values. Each of these vectors represents a rule associated with a predefined class, and is divided into two parts: the first part corresponds to the attributes that might be present in the example for it to belong to the class being predicted, and the second part represents attributes that must be absent from the examples belonging to the class being considered (see Figure 2). Given this representation, datasets with many attributes do not necessarily need to consider all of them. In this case, an effective attribute selection becomes an almost mandatory process. Note that, as in most flat classification EAs, the class is not encoded in the individual representation.

Having defined the the type of individual representation and consequent classification model, the population needs to be initialized. Usually GAs for flat classification employ intelligent initialization methods, in order to guarantee no rules covering no examples will be generated. This process was also incorporated to HGA, as defined in Section III-C. Then, the standard version of the GA is run, with a few modifications detailed in the next section. However, when applying genetic operators to improve search, classification-related operators are introduced, such as rule generalization and specialization [8]. Here we adapted these operators to respect the properties imposed by the representation used, as shown in Section III-E.

It is important to point out that the term hierarchical genetic algorithm (HGA) [12][13] was already defined in the literature in a completely different context. [13] studied how much we can increase the size of problems maintaining their feasibility to be solved with EAs. This feasibility depends on the properties of the problem, being one of the most important the variable dependency. When two variables are interdependent, the contribution of the first to the fitness function depends on the value of the second. In this context, a problem has hierarchical structure when its variables are organized to form levels, and each variable belongs or contributes to one or more levels. Variables in each level are divided into non-overlapping modules, which can be nested. A formal definition of these problems is given in [13], but note they have no relation with the problem discussed in this paper.

Regarding non-evolutionary hierarchical classification problems, two good surveys can be found in [2], [14]. Here we simply describe the two most common approaches used to tackle this problem: the local (or top-down) and the global (or big-bang) approach. In the top-down approach, one first chooses if binary classifiers will be generated for each node of the hierarchy tree or if classifiers will account for sibling nodes in the hierarchy. The top-down approach can use any type of classifier, and is only a wrapper that determines how models will be constructed and later applied.

After models are built, they are applied to the test set in a top-down fashion. Hence, given a new example, the classifiers of the nodes in the first level of the hierarchy are used to classify the example, and a first-level class c_1 is assigned to it. From there, only the models generated taking into account the children classes of c_1 are considered (i.e., if class c_2 in the first level was already disregarded, its children will also be). This process is repeated until a leaf-node is found (in the case of problems with mandatory leaf prediction) or until another stopping criterion is reached. Note that in this approach classifiers are built using only local information about the hierarchy.

In order to consider more global information when creating hierarchical classification models, the big-bang approach was proposed. This approach produces only one model for the whole class hierarchy. Most algorithms following this approach are based on the Bayesian Theory [15] or Support Vector Machines [16], and are usually developed considering the text classification task. Besides, some extension of well-

Algorithm 1 HCGATrain(Node,committee)

1: $chNode =$ children of $Node$
2: **if** $chNode = \emptyset$ **then**
3: **return**
4: **end if**
5: $pop \leftarrow committee \cup$ CreateSpecies($chNode$)
6: Evolve(pop)
7: $committee =$ SelectIndividuals(pop)
8: **for each** i in $chNode$ **do**
9: HCGATrain(i, $committee_i$)
10: **end for**

know flat classification algorithms were proposed, such as C4.5 [5] and Ripper [17].

The use of an evolutionary approaches to solve this problem, to the best of our knowledge, was not tried yet, despite the success of these approaches for solving flat classification problems. However, some natural computation methods, such as PSO and ACO working in a top down fashion, were proposed in [18][19].

III. A Hierarchical Classification Genetic Algorithm

This section presents HCGA (Hierarchical Classification Genetic Algorithm), a genetic algorithm developed to solve the hierarchical classification problem. The algorithm follows a top-down approach, but presents one main difference regarding other classification methods in the model building process. While all of them create independent models for classes/levels of the hierarchy, HCGA uses the models obtained in level l as a starting point for models in level $l + 1$.

Hence, we can say that HCGA follows a *top-down approach both during model construction and classification* of new examples. The rationale behind a top-down building model process is that the model is refined level by level. One might argue that this process can lead to serious error propagation. However, this problem is tackled by allowing HCGA to work with both new generated models as well as those created by the HCGA in the superior levels.

A. Overview

As any classification method, HCGA works in two phases. In the first phase, it creates a set of classification models from the training set, each considering a different hierarchy node. In a second phase, it applies the models previously built to predict the classes of new examples, whose classes are unknown.

Alg. 1 illustrates the first part of the process: the training phase. Regarding the basic GA flow [20], HCGATrain presents two particularities. First, it works with speciation, where each class is represented by species [21]. Second, as previously mentioned, the classification models are built following a top-down approach. Hence, given a node $Node$, which is initially the root of the hierarchy tree and covers all examples, and an empty $committee$, the method works as follows.

Initially, the algorithm lists all children of $Node$ (line 1), and then creates an initial population of individuals based on the $committee$, as described in Section III-C. This population is composed of species, each of them representing a child class of $Node$ (line 5). Each individual is evaluated according to its $f1$ measure, in order to consider both the precision and recall obtained. A fitness sharing method is used to maintain population diversity, and is described in Section III-D.

After that, the population is evolved using the traditional crossover and mutation operators, plus classification-specific operations, showed in Section III-E. When the evolution process finishes, a committee of individuals is selected (line 7), and used to generate a subset of the population for the next level. The selection process is described in Section III-F.

When HCGATrain is first called, the committee is empty. From the second generation onwards, the committee holds the best rules found to classify examples in level l. Hence, rules in level $l + 1$ start to be evolved using traditional methods as well as variations of the best rules found so far. Next, for each children node of $node$, we start this process again.

At the end of the entire process, the number of models generated is $1 + \sum_{i=2}^{l} \#ofchildren_i$. The models generated are highlighted in Fig. 1. Consider the hierarchy in Fig. 1, where "All" level 0. The first model is created for all classes in the first level. After that, for each node in the first level, a new model is produced for its children. Hence, for the presented hierarchy, four models would be produced.

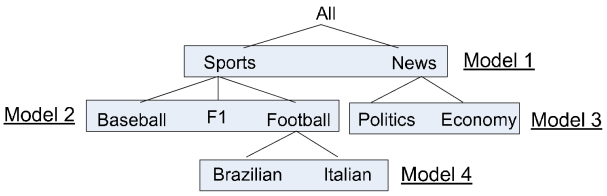


Fig. 1. A hierarchy of classes and the four models generated by the HCGA

After a set of models is built from the hierarchy, the HCA-Test phase starts. It follows the traditional hierarchical top-down approach. Each example is submitted to all rules predicting classes in the first hierarchical level. If more than one rule covers the example, and these rules disagree on the final class of the example, the rule with the highest confidence determines the final class. If we have $r1$ rules predicting class 1 and $r2$ rules predicting class 2, the confidences of all rules are summed up, and the class which obtains the highest confidence is considered the final class.

After the class c_1 in the first level is predicted, only the models of the children classes of C_1 are applied. Children of other nodes are disregarded. This process goes on while the confidence of the classifier is above a predefined threshold t. Note that this version of HCGA works with the same threshold for all classes, but this can be easily change in order to relax the threshold for very difficult classes.

B. Individual Representation

Each GA individual represents a classification rule, associated with a pre-defined class. These rules follow a very

simple representation, inspired by the one proposed in [11]: individuals are binary strings, where the first k bits represent attributes present in the example and the last k represent the absent attributes. The positive part of the individual requires that at least one of the attributes is present in the example (it implements the *or* logical operation), while the negative part works as an *and*, requiring that all of the attributes set are absent from the example.

Fig. 2 shows an example of an individual representing 5 attributes which might appear or not in ana example for it to belong to a specific class. Note that the value of k is fixed, and for big datasets a feature selection phase is performed to identify the most informative attributes, which are than considered during the search. In the case of this individual, it classifies examples which present attributes 3 *or* 4, and do not present attributes 2 *and* 5.

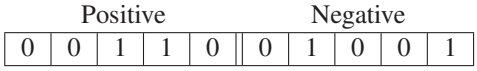


Fig. 2. Example of an Individual

C. Population Initialization

In classification problems, in general, a random initialization of the population is not recommended, once many of the generated individuals might not classify any examples in the dataset. Hence, here the population is initialized in two different ways.

First, as in traditional flat classification, an example is randomly chosen from the training set, and its attributes considered to generate a new rule. Second, an attribute selection is performed using χ^2, generating an attribute rank. Then, the probability of turning a bit on or off is proportional to its position in the attribute rank. In other words, the better an attribute is to separate examples in different classes, the greater the probability of it being set. The procedures just described can be used in isolation or in conjunction. Experiments showed that mixing up both improves the diversity of the individuals.

However, one of the most interesting features of HCGA is population initialization from the second level on. As illustrated in Algorithm 1, once models for classes in the first level are created, a committee with a set of individuals is selected to be a starting point for classifiers in the next level. The committee selection scheme is described in Section III-F. This allows us to perform refinements over the high-hierarchical level results, instead of initializing the search all over again.

Hence, for all hierarchical levels except the first, apart from generating individuals using the aforementioned methods, the populations are also based on the committee or some variations of the committee. These variations are done by (un)setting a predefined number of bits chosen according to the χ^2-based probabilities, explained earlier.

D. Fitness Function and Fitness Sharing

As the classifiers are created following a top-down approach, no special metric for evaluating hierarchical classification is used. Instead, all evaluation are based on $\mu f1$. $\mu f1$ is a standard metric to evaluate classifiers, derived from the $f1$ metric, which is the harmonic mean of precision and recall. However, it weights the $f1$ of each class according to the number of examples in each class. Eqs. 2 and 3 define the micro-precision (μPr) and micro-recall (μRe), for a problem with m classes. TP_i corresponds to the number of examples of class i correctly classified, FP_i corresponds to the number of examples in classes different from i but classified as i, and FN_i corresponds to the number missclassified examples in class i.

$$\mu f1 = \frac{2 \times \mu Pr \times \mu Re}{\mu Pr + \mu Re} \tag{1}$$

$$\mu Pr = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} (TP_i + FP_i)} \tag{2}$$

$$\mu Re = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} (TP_i + FN_i)} \tag{3}$$

We observed in preliminary experiments that the diversity within species was not being preserved. Hence, a niching scheme based on the standard fitness sharing method [21] was implemented. The method penalizes the fitness of similar individuals so the new fitness distribution in the population is dispersed in niches. For that, we need to define a way to calculate the distance between two individuals (rules) based on how different are the examples they cover. Given two individuals ind_1 and ind_2, where ind_1 covers the $covSet_1$ examples and ind_2 covers $covSet_2$, then they are considered similar if the distance given by the Eq. 4 is less than a θ_{share} value.

$$dist(ind_1, ind_2) = 1 - \frac{||covSet_1 \cap covSet_2||}{||covSet_1 \cup covSet_2||} \tag{4}$$

E. Genetic Operators

HCGA uses the traditional uniform crossover and single-point mutation operators, but also implements classification-specific ones, namely the generalization and specialization operations. As their name suggests, the generalization operator is applied to rules that are too specific while the specialization operator is applied to too general rules. Considering the representation of the individuals, the specialization operator removes attributes from the positive side of the chromosome or inserts new attributes to the negative side. The generalization operator works in the opposite way, adding attributes to the positive side and removing from the negative side.

F. Committee Selection

The committee selection can be done in three different ways:

1) Using the best k individuals based on the fitness function (same used in elitism);
2) Using the best individual based on the fitness function, followed by the $k-1$ individuals which are more distant from it, according to the distance measure defined in Eq. 4;

TABLE I
CHARACTERISTICS OF THE GPCR DATASETS: NUMBER OF CLASSES PER LEVEL, NUMBER OF ATTRIBUTES, NUMBER OF EXAMPLES AND AVERAGE NUMBER OF ATTRIBUTES PER EXAMPLE

Name	#Classes per lvl	#Atts.	#Ex.	Avg. atts. per ex.
GPCRPrints	8/46/76/49 (180)	281	5422	2,2
GPCRPfam	12/52/79/49 (193)	73	7077	1,1
GPCRInterpro	12/54/82/50 (199)	448	7461	2,3
GPCRProsite	9/50/79/49 (188)	127	6261	2,1

TABLE II
MAIN HCGAPARAMETERS

Parameter	Value
Generations	50
Individuals per species	50
Crossover Rate	0.9
Mutation Rate	0.4
Specia/General Rate	0.4
Tournament Size	2
Elitism	0.1

3) Using the best k individuals based on fitness sharing.

The first method assumes that the best individuals are the better predictors, while the second recognizes that being the best is not sufficient. Selected individuals need to be different enough to cover a greater variety of examples. Finally, the third method tries selecting individuals in different niches to better predict examples. All three operations were tested, and the best one selected in order to run the final experiments.

Note that the selection of a committee instead of a single individual, with the same simple HCGA individual representation, substitutes the need of evaluating more complex rules for classification, as we can see in the results on the next section.

IV. EXPERIMENTAL RESULTS

This section presents the results of applying HCGA in a set of four bioinformatics datasets regarding GPCR activity. Identifying GPCR proteins and their families is particularly important for medical applications, since it is believed that 40–50% of current medical drugs target GPCR activity [7]. We first describe the datasets, then report the results obtained compared to other five state of the art algorithms.

A. Datasets

GPCRs are essentially transmembrane proteins (i.e., they cross the cell's membrane) that transmit signals received from outside the cell to within the cell. Different signals activate different biological processes within the cell, and GPCRs are involved in the transmission of many different types of signals.

In all datasets, described in Table I, each example represents a protein, and each protein is described by a set of "motifs." A motif is a pattern or a "signature" typically found in some proteins. It is basically a sequence or partial sequence of amino acids that can be used to identify the function and/or the family of a protein [22]. Each motif is represented by a binary attribute, which indicates the presence or absence of the motif in each protein. This type of protein representation implicitly incorporates a lot of background knowledge about proteins available in the literature, since protein motifs have been created and refined by expert biologists and bioinformaticians for a long time.

In the GPCR datasets, the single class to be predicted is the function of the GPCR, which depends on which type of molecule binds to the part of the GPCR outside the cell and triggers the signal transmission into the cell. The classes are organized in four hierarchical levels. The predictor attributes vary depending on the dataset, but all the three types of motifs used as attributes (Prints, Pfam, Interpro, and Prosite motifs [22]) represent protein signatures. Note that these datasets are very sparse (fourth column in Table I). The creation of the GPCR datasets is described in [23].

However, note that the datasets available in [23] present one numerical attribute: the molecular weight of the protein. This attribute was removed from the dataset. We performed experiments which showed that by removing this attribute we were not degrading the prediction accuracy obtained, apart from the dataset GPCRProsite.

TABLE III
AVERAGE ACCURACY AND STANDARD DEVIATION FOR GPCRPRINTS DATASET USING THE FOLLOWING PARAMETERS: θ_{share} = 0.8, HIERARCHICAL CONFIDENCE THRESHOLD OF 0.9 AND COMMITTEE SIZE = 3

Method	Level 1	Level 2	Level 3	Level 4
HCGA	90.91 ± 0.78	76.87 ± 1.71	55.97 ± 1.25	89.39 ± 0.73
OlexGA	91.74 ± 0.35 ●	77.52 ± 0.81 ●	49.52 ± 0.64 ▲	77.60 ± 1.56 ▲
NB	90.93 ± 0.53 ●	76.18 ± 0.32 ●	52.52 ± 0.44 ▲	90.63 ± 0.33 ▼
BNet	90.47 ± 0.39 ●	75.21 ± 0.60 ●	48.01 ± 0.22 ▲	69.08 ± 0.61 ▲
J48	91.63 ± 0.42 ●	80.23 ± 0.19 ▼	54.86 ± 0.58 ●	80.64 ± 0.98 ▲
HyPi	89.84 ± 0.87 ●	14.50 ± 1.04 ▲	6.13 ± 1.14 ▲	7.92 ± 2.97 ▲

TABLE IV
AVERAGE ACCURACY AND STANDARD DEVIATION FOR GPCRPFAM DATABASE USING THE FOLLOWING PARAMETERS: θ_{share} = 0.9, HIERARCHICAL CONFIDENCE THRESHOLD OF 0.9 AND COMMITTEE SIZE = 4

Method	Level 1	Level 2	Level 3	Level 4
HCGA	92.24 ± 0.67	47.27 ± 1.28	12.47 ± 0.24	0.00 ± 0.00
OlexGA	92.86 ± 0.20 ●	24.33 ± 0.26 ▲	7.18 ± 0.17 ▲	1.19 ± 0.15 ▼
NB	92.36 ± 0.25 ●	46.80 ± 0.86 ●	12.82 ± 0.29 ●	0.00 ± 0.00 ●
BNet	91.83 ± 0.49 ●	46.82 ± 0.81 ●	12.70 ± 0.25 ●	0.00 ± 0.00 ●
J48	92.83 ± 0.22 ●	48.01 ± 0.92 ●	12.94 ± 0.21 ▼	0.00 ± 0.00 ●
HyPi	92.34 ± 0.29 ●	22.90 ± 0.26 ▲	6.78 ± 0.12 ▲	1.19 ± 0.15 ▼

TABLE V
AVERAGE ACCURACY AND STANDARD DEVIATION FOR GPCRINTERPRO DATABASE USING THE FOLLOWING PARAMETERS: θ_{share} = 0.8, HIERARCHICAL CONFIDENCE THRESHOLD OF 0.9 AND COMMITTEE SIZE = 2

Method	Level 1	Level 2	Level 3	Level 4
HCGA	88.33 ± 0.47	91.71 ± 0.50	48.69 ± 2.15	89.67 ± 0.72
OlexGA	90.20 ± 0.24 ▼	74.02 ± 0.46 ▲	49.73 ± 1.21 ●	82.15 ± 0.67 ▲
NB	90.11 ± 0.27 ▼	79.05 ± 0.38 ▲	52.88 ± 0.94 ▼	89.32 ± 0.60 ●
BNet	89.40 ± 0.24 ▼	73.59 ± 0.73 ▲	47.11 ± 0.72 ●	88.15 ± 0.66 ▲
J48	90.14 ± 0.29 ▼	78.91 ± 0.44 ▲	53.66 ± 1.22 ▼	92.58 ± 0.60 ▼
HyPi	78.33 ± 3.33 ▲	13.30 ± 3.66 ▲	4.34 ± 1.05 ▲	1.49 ± 0.20 ▲

TABLE VI
AVERAGE ACCURACY AND STANDARD DEVIATION FOR GPCRPROSITE DATABASE USING THE FOLLOWING PARAMETERS: θ_{share} = 0.8, HIERARCHICAL CONFIDENCE THRESHOLD OF 0.5 AND COMMITTEE SIZE = 4

Method	Level 1	Level 2	Level 3	Level 4
HCGA	84.04 ± 0.22	46.71 ± 0.93	19.27 ± 0.59	44.42 ± 0.91
OlexGA	84.25 ± 0.20 ●	27.57 ± 0.36 ▲	13.26 ± 0.33 ▲	12.70 ± 0.64 ▲
NB	83.88 ± 0.25 ●	46.55 ± 0.64 ●	18.55 ± 0.43 ●	44.40 ± 1.01 ●
BNet	84.13 ± 0.23 ●	46.26 ± 0.67 ●	18.43 ± 0.65 ●	41.74 ± 1.08 ▲
J48	85.08 ± 0.17 ▼	47.19 ± 0.76 ●	18.88 ± 0.52 ●	41.85 ± 1.16 ▲
HyPi	83.61 ± 0.12 ▲	21.24 ± 0.27 ▲	10.91 ± 0.30 ▲	12.21 ± 0.55 ▲

B. Results

After preparing the datasets, preliminary experiments were performed in order to choose a set of parameters for HCGA. Table II shows the chosen values for the basic parameters, which are not optimal. Note that the value of the mutation rate is quite high (0.4), but in an initial empirical evaluation it showed to be the best to prevent a premature convergence of the population. We also played with the population initialization methods, which can be done in two ways in the first levels and three ways from the second on (see Section III-C). In the first level, 80% of the population is initialized using training examples. This number drops to 50% from the second hierarchical level on. Also, given the sparsity of the datasets, a maximum number of attributes is allowed to be set in cases where the χ^2 initialization is used. This number is set to 30 for all datasets.

There are three other parameters to be set: the θ_{share} value for fitness sharing, the threshold value for going down the hierarchical tree during classification and the size of the committee. These three parameters are very dataset dependent, and their values are reported together with the experiments. All experiments are based on the third type of committee defined in Section III-F, as it presented the best results. An implementation of HCGA with a dynamically adaptive committee type is left for future work.

Tables III to VI show the results of micro-f1 obtained in the experiments. The first column presents the algorithms executed, followed by the results in each of the four levels of hierarchy (average ± standard deviation). All figures are based on a 5-fold cross-validation with 5 repetitions of HCGA (average over 25 runs). The first line presents the results obtained by HCGA, followed by the execution of five other algorithms, namely OlexGA, Naive Bayes (NB), Bayesian Networks (BNet), J48 and HyperPipes (HyPi) using the top-down approach, with one classifier generated for each hierarchy node. These algorithms were the same ones use din [18]. All algorithms were run using Weka [24] with its default parameters. A comparison with the PSO/ACO algorithm proposed in [18] using the same datasets is left for future work.

Apart from the first line (HCGA), each result is followed by a symbol, which indicates whether the results are statistically significant according to a 2-tailed paired t-test, given a 95% confidence level. ▲ implies that HCGA is statistically significant better than the method referred by that line, ● represents a non significant variation and ▼ a significant negative variation of HCGA over the method being considered.

Analyzing the results in Table III, we observe that for level 1 all classifiers are statistically the same, while in the second level J48 is the best over all, obtaining an accuracy of 80% against the 76-77% of other classifiers. As we go down the hierarchical tree, the classification becomes more difficult, as we have more classes and fewer examples. In GPCRPrints, for instance, the first level has 8 classes, while the third level has 76. In the third level, the best results are obtained by HCGA and J48. All other methods have statistically worse values of micro-f1. For the fourth and last levels, HCGA is better than all classifiers apart from Naive Bayes, which obtains a significant but very small gain. Hence, in general, if we had to choose one classifier to predict function in all levels, J48 would present advantages over the second level. However, in the fourth level, which usually is the one that brings more interesting information to the user, J48 presents a micro-f1 of 80.64, while HCGA reaches 89.39. In this way, HCGA would be a good choice.

It is important to point out that the results presented here can still be improved. Considering the way HCGA works, adaptive parameter tuning or a more extensive optimization technic made level by level could significantly improve the results. However, the main idea of this paper was to present a proof of concept result. It is also interesting to point out that HCGA has the advantage of generating models which can be easily interpretable, as they concern only the presence or absence of an attribute (in this case, a protein motif) in a class.

Regarding the dataset GPCRPfam, whose results are reported in Table IV. In the first level, again all algorithms present similar performance. This is expected, as we are dealing with flat classifiers, and predicting the first level is not very different from solving a flat classification model. For the second level, however, HCGA and other three classifiers still present good and statistically compared performance. In the third level, HCGA is not statistically better than J48, but its results are equivalent to those obtained by the Bayesian approaches. The last level can be predicted with very small accuracies by OlexGA and HyPi, and the other classifiers cannot generate any classification model. It is interesting to point out that HCGA is better than OlexGA in levels 2 and 3, but then it misses out something in the last level. Investigating why that happened, we notice that this is a results of error propagation. Most of the classifiers cannot distinguish classes 001.002 and 001.005, as their attribute values are pretty much the same. Concluding, in GPCRPfam HCGA is very competitive with the other methods. However, notice that this type of motif is less indicated to predict proteins at lower levels of the hierarchy, when compared to the results obtained when using Prints-based datasets.

When looking at Table V, HCGA does not perform as good as in the previously two databases. In this case, classification in the first level is statistically inferior to those of four out

of five baselines. The loss was small, and in the second level HCGA recovering, having results statistically better than all other classifiers. However, this situation is reversed again in level 3 and 4. HCGA still has equivalent results to those of Bayesian Networks and OlexGA, but worse than J48 and Naive Bayes. In the fourth level, J48 is better than all other methods, followed by HCGA.

The results of GPCRProsite, reported in Table VI, show HCGA appears to be a better method at providing results for the last hierarchical level. Although in the first level it presents a small loss for J48, in the following levels it is competitive with the state of the art algorithms, and then in the last level presents results statistically better than those produced by all other methods.

A summary of the results presented for all datasets can be found in Table VII. For each dataset in each level we performed 5 comparisons. Note that, overall, the HCGA has statistically better results in 32 cases, statistically equal results in 35 cases, and 13 losses. In general, the HCGA improves the results as the levels of the hierarchy become deeper, which is desired. In all datasets, apart from GPCRPfam, the results obtained are quite stable.

Figure 3 shows the results of evolution for specific species in the first to fourth levels. The graphs report the result of the best individual fitness in the training (BestTrain), validation (BestValidation) and test sets (BestTest). It also reports the values of precision (PrTest) and recall (RcTest) in the test set separately, so that we can have a better idea of how the rules are evolving. For the sake of completeness, we also show the average fitness (avgValidation) of the entire population in the validation set. Note that the graphs reported the best individual, which is then combined to a few others to form the committee. Note that there is no overfitting, as the fitness in the training, validation and test sets are very similar. It is interesting to analyze how precision and recall behave.

Apart from the first level, where the recall is close to 100% while the values of precision are around 87%, for all other levels the values of precision are always greater than those do recall. In some cases, for instance, in class 001, the first rules generated are already quite good, and HCGA only improves their recall overtime. In contrast, in the three graphs in the second level, corresponding to the 3^{rd} and 4^{th} levels of the hierarchy, the precision is already 100% during the first 5 to 10 generations. However, the recall improves significantly.

TABLE VII
SUMMARY OF HCGARESULTS

GPCR	Prints			PFam			Interpro			Prosite			Total		
	▲	●	▼	▲	●	▼	▲	●	▼	▲	●	▼	▲	●	▼
L1	0	5	0	0	5	0	1	0	4	1	3	1	2	13	5
L2	1	3	1	2	3	0	5	0	0	2	3	0	10	9	1
L3	4	1	0	2	2	1	1	2	2	2	3	0	9	8	3
L4	4	0	1	0	3	2	3	1	1	4	1	0	11	5	4
Total	9	9	2	4	13	3	10	3	7	9	10	1	32	35	13

Regarding the execution time, experiments with HCGA varied from 4 to 7 minutes in a Intel(R) Core(TM) i7 CPU with 2.67GHz and 6 GB RAM. Reducing the execution time by using threads and eliminating unnecessary experimentation prints, calculations and disk accesses is left for future work.

V. CONCLUSIONS AND FUTURE WORK

This paper presented HCGA, a genetic algorithm designed for hierarchical classification. Compared to other algorithms in the literature, HCGA has the advantage of dealing with both local and global information simultaneously, while other methods can deal with only one of them. The method achieves that by working in a top-down fashion both when producing the models as well as when classifying new examples. This is in contrast with other top-down methods, which explore this strategy only during the test phase.

HCGA was applied to four datasets concerning the prediction of GPCR activity, a relevant subject to the bioinformatics area, as GPCR proteins are the target of around 50% of medical drug. The results showed HCGA can produce results statistically as good as or better than five baseline algorithms run together with the traditional top-down approach.

As future works, a more extensive study of parameters needs to be performed. Our experiments showed that varying the selection strategy of the committee, and also the threshold used to decide if an example should be moved downwards the tree during the test phase, changes significantly the results.

Moreover, we want to explore the algorithm for text classification. We already start experiments in that direction, but in this case, as datasets have at least 50,000 attributes (terms), a great deal of time has to be spent preprocessing documents, with special attention to the attribute selection phase. As this was not the major focus of this work, preparing text datasets for future experiments is our next step. Finally, new methods for hierarchical attribute selection will be explored in order to reduce the search space by focusing only on those attributes which are relevant to all hierarchical levels.

ACKNOWLEDGMENTS

This work was partially supported by CNPq, CAPES, Fapemig, and InWeb - Brazilian National Institute of Science and Technology for the Web.

REFERENCES

[1] A. A. Freitas, "A review of evolutionary algorithms for data mining," in *Soft Computing for Knowledge Discovery and Data Mining*, O. Maimon and L. Rokach, Eds., 2007, pp. 61–93.

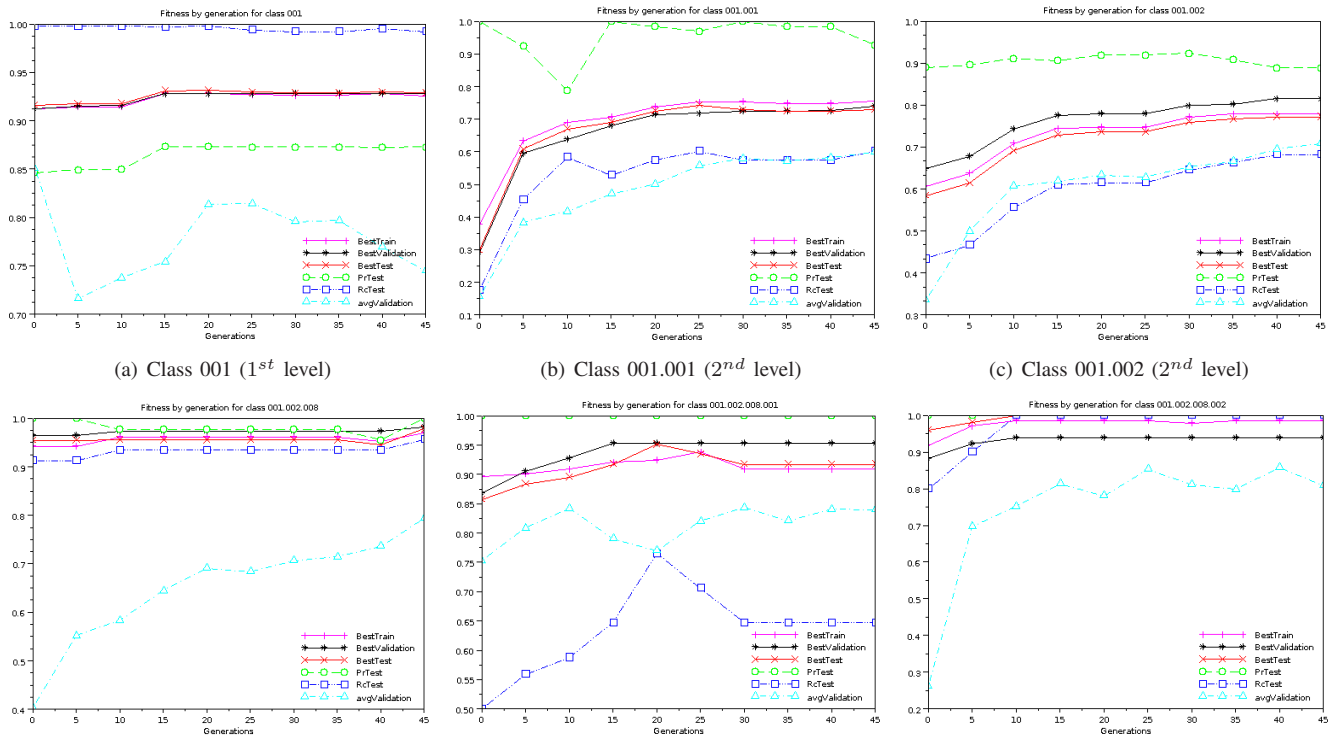
[2] C. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31–72, 2011.

[3] K. Punera, S. Rajan, and J. Ghosh, "Automatically learning document taxonomies for hierarchical classification," in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*. International World Wide Web Conference, 2005, pp. 1010–1011.

[4] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *J. Mach. Learn. Res.*, vol. 7, pp. 31–54, 2006.

[5] A. Clare and R. D. King, "Predicting gene function in saccharomyces cerevisiae," *Bioinformatics*, vol. 19, no. $suppl_2$, pp. ii42–49, 2003.

[6] C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, pp. 185–214, 2008.



(a) Class 001 (1^{st} level) (b) Class 001.001 (2^{nd} level) (c) Class 001.002 (2^{nd} level)

(d) Class 001.002.008 (3^{rd} level) (e) Class 001.002.008.001 (4^{th} level) (f) Class 001.002.008.002 (4^{th} level)

Fig. 3. Best individual HCGAevolution in GPCRInterpro set in some classes

[7] D. Fillmore, "It′s a GPCR world," *Modern Drug Discovery*, vol. 11, no. 7, pp. 24–28, 2004.

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

[9] J. de Freitas, G. L. Pappa, A. S. da Silva, M. A. Gonçalves, E. S. de Moura, A. Veloso, A. H. F. Laender, and M. G. de Carvalho, "Active learning genetic programming for record deduplication," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[10] A. Tsakonas, G. Dounias, J. Jantzen, H. Axer, B. Bjerregaard, and D. G. von Keyserlingk, "Evolving rule-based systems in two medical domains using genetic programming," *Artificial Intelligence in Medicine*, vol. 32, no. 3, pp. 195–216, 2004.

[11] A. Pietramala, V. L. Policicchio, P. Rullo, and I. Sidhu, "A genetic algorithm for text classification rule induction," in *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, 2008, pp. 188–203.

[12] E. D. de Jong, R. A. Watson, and D. Thierens, "On the complexity of hierarchical problem solving," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05, 2005, pp. 1201–1208.

[13] E. D. de Jong, D. Thierens, and R. A. Watson, "Hierarchical genetic algorithms," in *Parallel Problem Solving from Nature- PPSN VIII*, 2004, pp. 1201–1208.

[14] M. Ceci and D. Malerba, "Classifying web documents in a hierarchy of categories: a comprehensive study," *J. Intell. Inf. Syst.*, vol. 28, no. 1, pp. 37–78, 2007.

[15] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving text classification by shrinkage in a hierarchy of classes," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 359–367.

[16] S. Dumais and H. Chen, "Hierarchical classification of web content," in *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000, pp. 256–263.

[17] M. Sasaki and K. Kita, "Rule-based text categorization using hierarchical categories," in *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, 1998, pp. 2827–2830.

[18] N. Holden and A. A. Freitas, "Improving the performance of hierarchical classification with swarm intelligence," in *Proceedings of the 6th European conference on Evolutionary computation, machine learning and data mining in bioinformatics*, ser. EvoBIO '08, 2008, pp. 48–60.

[19] F. E. Otero, A. A. Freitas, and C. G. Johnson, "A hierarchical classification ant colony algorithm for predicting gene ontology terms," in *Proceedings of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, ser. EvoBIO '09, 2009, pp. 68–79.

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

[21] T. Baeck, D. Fogel, and Z. Michalewicz, *Evolutionary Computation 2: Advanced Algorithms and Operators*. Taylor and Francis, 2000.

[22] P. G. Higgs and T. K. Attwood, *Bioinformatics and Molecular Evolution*. Blackwell, 2005.

[23] N. Holden and A. Freitas, "Hierarchical classification of g-protein-coupled receptors with a pso/aco algorithm," in *Proc. IEEE Swarm Intelligence Symposium (SIS-06)*, 2006, pp. 77–84.

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