Applied Soft Computing 16 (2014) 34–49

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A novel ant colony optimization based single path hierarchical classification algorithm for predicting gene ontology

Salabat Khan[a,*], Abdul Rauf Baig[b], Waseem Shahzad[a]

[a] National University of Computer and Emerging Sciences, Computer Science Department, Islamabad, Pakistan
[b] Al Imam Mohammad Ibn Saud Islamic University (IMSIU),College of Computer and Information Sciences, Riyadh, Saudi Arabia

ARTICLE INFO

Article history:
Received 4 March 2012
Received in revised form 13 April 2013
Accepted 25 November 2013
Available online 11 December 2013

Keywords:
Hierarchical classification
Ant colony optimization
Bio-informatics data sets with gene ontology
Protein function prediction
Correlation based IF–THEN rule list

ABSTRACT

There exist numerous state of the art classification algorithms that are designed to handle the data with nominal or binary class labels. Unfortunately, less attention is given to the genre of classification problems where the classes are organized as a structured hierarchy; such as protein function prediction (target area in this work), test scores, gene ontology, web page categorization, text categorization etc. The structured hierarchy is usually represented as a tree or a directed acyclic graph (DAG) where there exist IS-A relationship among the class labels. Class labels at upper level of the hierarchy are more abstract and easy to predict whereas class labels at deeper level are most specific and challenging for correct prediction. It is helpful to consider this class hierarchy for designing a hypothesis that can handle the tradeoff between prediction accuracy and prediction specificity. In this paper, a novel ant colony optimization (ACO) based single path hierarchical classification algorithm is proposed that incorporates the given class hierarchy during its learning phase. The algorithm produces IF–THEN ordered rule list and thus offer comprehensible classification model. Detailed discussion on the architecture and design of the proposed technique is provided which is followed by the empirical evaluation on six ion-channels data sets (related to protein function prediction) and two publicly available data sets. The performance of the algorithm is encouraging as compared to the existing methods based on the statistically significant Student's *t*-test (keeping in view, prediction accuracy and specificity) and thus confirm the promising ability of the proposed technique for hierarchical classification task.

1. Introduction

Advanced sensing, capturing and computing technologies enable us to collect large amount of complex (possibly, raw) data in many fields of lives but how do we know which portion of data is important and gives us an insight to help our decision making process? Moreover, when this problem is faced in the medical domain and the decision become a matter of life and death of a patient, finding the correct set of information become more prominent and crucial. Data mining techniques can be used to extract implicit, previously unknown and potentially useful [1] patterns and knowledge of interests from these vast data stores for varied purposes. One such important data mining technique, known as classification, is successfully used in a myriad of applications e.g. decision making, fraud detection, medical diagnosis, credit scoring, customer relationship management, character recognition, speech recognition, protein function prediction etc.

Numerous classification techniques are proposed throughout the decades, such as Decision Trees, Neural Networks (NN), *k*-Nearest Neighbor's (*k*-NN), Logistic Regression and Support Vector Machine (SVM), etc. [2–4]. Some of these techniques (e.g. NN and SVM) produce incomprehensible classification models that are usually opaque to common users, while others e.g., IF–THEN rule list produced with decision trees, are easily comprehensible to the experts working in different domains. These techniques are reported to perform well in various domains and considered computationally efficient (e.g. SVM), robust to noisy data (e.g. decision trees) and easy to learn (e.g. *k*-NN). However, most of these classification techniques are designed to handle the data with binary or nominal class labels (where class labels are independent). These classical strategies lack the ability to handle the problems where the class labels are related and are organized based on a class hierarchical structure (CHS). The later one is a complex instance of classification problems, known as hierarchical classification, as compared to the one level flat classification problems [5]. Hierarchical classification has applications in various domains.

This research work focuses on hierarchical protein function prediction defined under the scheme of Gene Ontology (GO), specifically "molecular function" domain. The GO structure [6] represents

* Corresponding author. Tel.: +92 3345202110.
E-mail addresses: salabat.khan@nu.edu.pk (S. Khan), raufbaig@ccis.imamu.edu.sa (A.R. Baig), waseem.shahzad@nu.edu.pk (W. Shahzad).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.11.012

the relationship among the protein functions using directed acyclic graph (DAG) CHS. It is well known that a protein can perform multiple functions (considered as class labels) and these functions are usually related (modeled as tree or DAG based IS-A relationship), makes protein function prediction a suitable and ideal problem for proposed algorithm. There is a large amount of uncharacterized protein data which is available for analysis that has led to an increased interest in computational methods to support the investigation of the role of proteins in an organism [7,8]. Analyzing the functions of proteins in different organisms is crucial to improve biological knowledge, diagnosis and treatment of diseases. It is not possible to conduct the biological experiments for the functional essay analysis of every uncharacterized protein due to involvement of high cost and human based analysis [9]. It therefore raise the need of the development of computational methods (especially related to data mining domain, like the one proposed in this paper) to be used for this purpose.

In case of hierarchical classification, the class labels to be predicted, are naturally organized as a class hierarchy/taxonomy, typically represented as a tree or DAG, see e.g. Fig. 1a and b. The class labels in the hierarchy are represented as nodes and the relationships between the class labels are shown with undirected edges. For tree structure, a node can have only one parent whereas no such restriction is imposed over DAG CHS. Predicting a single class label in the hierarchy, implies that all the ancestor class labels are also predicted. In other words, a single class label is a path from root node to the predicted child node (explained later) that is consistent to the IS-A relationship.

Considering the class hierarchy, nodes at the upper levels represent more general class labels whereas the nodes at the lower levels represent the more specific class labels. General class labels are easy to predict as numerous examples related to them are available (to the hypothesis learner). On the other hand, classes at the deeper levels of the hierarchy (i.e. specific classes) are difficult to predict as less information is available to discriminate among them. There is always a *tradeoff between generality and specificity* in hierarchical classification. In Fig. 2, a dataset is given with corresponding CHS for different animals. Given a test example (from this dataset), if we predict the class label 'Animal', the prediction is 100% accurate but we get no valuable information about the specific class of animal. Predicting specific class of animal is more important but the chances of erroneous prediction is high.

In order to identify the types of problems that our proposed algorithm can handle, more information is provided in follows. Based on class label(s) associated with an example, the hierarchical classification has further two categories [9]:

(1) *Hierarchical Single Label (path) Classification*: In this type of classification, an example is associated with only *one class label at any level* of the class hierarchy.
(2) *Hierarchical Multi-label (path) Classification*: In this type of classification, an example can be associated *with more than one class label at any level* of the hierarchy (multi-paths in the hierarchy).

The hierarchical classification problems can further be divided in two categories based on the level (depth) of the predicted class label [9,10]: (1) Mandatory Leaf Node Prediction (MLNP), and (2) Optional Leaf Node Prediction (OLNP) or Non-Mandatory Leaf Node Prediction (NMLNP). Based on MLNP, it is mandatory for a classifier to predict at least one of the leaf class labels (from CHS) for classifying a test example. The OLNP problems are somewhat flexible and classifier can predict class label(s) for a test example at any level of the class hierarchy. The proposed algorithm can only handle hierarchical single path classification problems, considering only the OLNP case.

The remainder of this paper is organized as follows. In the next section, we review related research for hierarchical classification task. In Section 3, we briefly present the basics and the background of ACO meta-heuristic. In Section 4, the architecture of the proposed solution will be discussed. Subsequently, in Section 5, we present simulation results to show the promising ability of our technique. Finally, Section 6 will conclude this work.

2. Related work

One simple approach to deal with the hierarchical classification problems is to completely ignore the given CHS by using a flat classification algorithm (e.g. decision tree or SVM, etc.), predicting only leaf class nodes. This approach provides an indirect solution to the hierarchical classification problem as if a class at leaf node is predicted, all the ancestor classes (considering the IS-A relationship) are also implicitly assigned to the instance being classified. However, this approach suffers with higher prediction error. Moreover, it cannot handle the OLNP based classification problems [10].

In order to incorporate the given CHS, researchers have proposed extensions in the baseline classification methods. These extended methods can basically be categorized based on multiple criterions [10]. First criterion concerns the ability of the method to handle the type of CHS (e.g. Tree or DAG). Second criterion is related to the depth of CHS that is being targeted during the classification procedure e.g. MLNP, OLNP, etc. Third criterion is about how the given CHS is explored? Keeping in view these three criterions, some strategies are reviewed in the following subsections.

2.1. *Local classifier approaches*

Local classifier approaches can handle MLNP and OLNP based hierarchical single path as well as multipath classification problems with both tree and DAG class hierarchical structures. Koller and Sahami [11] have proposed a local classifier approach based on top-down strategy. In general, each class node is associated with a baseline classifier, producing a classification model that predicts the presence/absence of the corresponding class labels for the given test instance. The question is: how many baseline classifiers, one need to train? There exist three standard ways: (1) training a local classifier per node (LCN), (2) training a local classifier per parent node (LCPN), and (3) training a local classifier per level (LCL) [10]. We have chosen LCN approach (using C4.5 Decision Tree as base classifier) as one of the competitors to the proposed algorithm [7]. Disadvantages of local classifier approaches are summarized in follows that motivated us to design a big-bang hierarchical classification approach (discussed in next subsection).

Local classifier approaches are computationally demanding, since a classifier need to be trained *n* times where *n* is the number of total class labels present in the CHS which can be thousands in numbers. These methods are usually affected with blocking problem (for NMLNP) where misclassifications at higher levels are propagated toward lower levels [10]. Top-down procedure, naturally fits a tree-structured class hierarchy, where if parent classifier rejects the test example, a child classifier does not receive the test example. When dealing with DAG based CHS, the top-down procedure requires a modification to handle multiple parent's property [10]. The class labels at deeper levels, potentially have fewer positive examples in contrast to a greater number of negative examples. Many baseline classifiers have problems with imbalanced class distribution and thus classification error at deeper level is expected to be high [7].

2.2. *Big-bang or global approach*

For big-bang classification approach, a single global model is built, considering all the hierarchically related classes at once, from

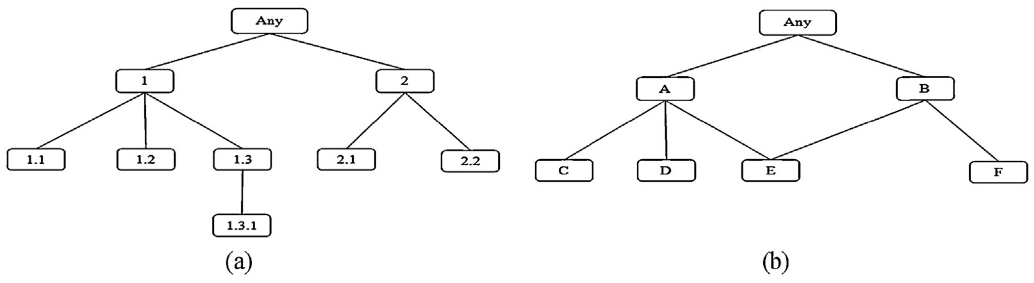


Fig. 1. (a) Tree hierarchy, (b) DAG hierarchy (two parents for 'E').

a global perspective. This strategy generates a global classification model that is compact in size and is typically considerably smaller as compared to any of the local classifier approaches, discussed in previous subsection. Big-bang approach is also known as "global" learning [10]. It is, however, difficult to model a global hierarchical classification technique. In order to determine potential competitor(s), some big-bang hierarchical classification algorithms are discussed in follows.

In the research work of Chen et al. [1], flat decision tree classifier is extended to handle the data arranged with hierarchical class labels. The algorithm use an extended entropy measure for the selection of best attributes. However, this algorithm cannot handle the DAG CHS and therefore is not suitable for protein function prediction problem. Another worth mentioning decision tree based method to handle the hierarchical multi-label classification task, is proposed in the work of Vens et al. [12], known as CLUS-HMC. The idea of CLUS-HMC is based on the theory of predictive clustering tree framework [13]. Each node in the tree is conceived as a cluster. The variance of the set of training examples (in class labels space) for a candidate cluster is used to decide about the selection of an attribute-value pair. CLUS-HMC provides solution to both the tree and DAG class hierarchical structures. CLUS-HMC is more suitable for hierarchical multi-path classification problems, so do not qualify, to become a candidate competitor of our proposed method.

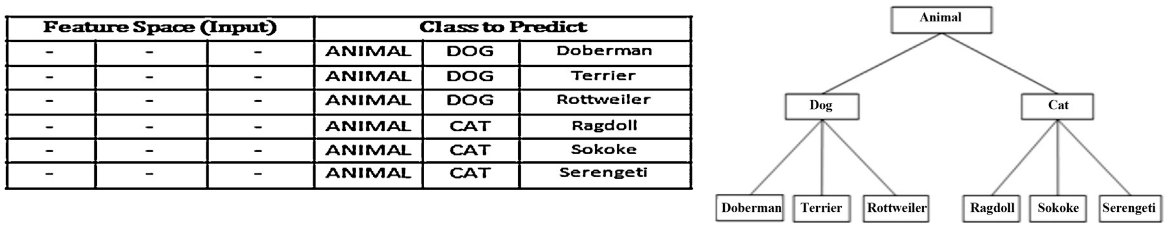
The hAM (stands for hierarchical AntMiner) [7] is the first ACO based algorithm that can handle hierarchical single path classification problems. It discovers hierarchical ordered IF–THEN rules. The search space is based on two sub-graphs. First construction graph, called Antecedent Construction Graph (ACG) is used to create antecedent part of the rule. Second construction graph, called Consequent Construction Graph (CCG) is used to form the consequent of the rule. The topology of ACG is same as used in c-AntMiner [15]. CCG's topology is exactly the same as to that of the given CHS. Interestingly, two separate ant colonies are employed to discover the hierarchical classification rules. One ant colony is employed into the ACG and the second ant colony is employed into the CCG. The first ant of the first colony is augmented with the first ant of the second colony and so on so forth the remaining ants in the first colony are paired with the corresponding ants in the second colony. The augmentation defines this way is used to form a complete IF–THEN rule. However, no communication is carried out between the ant colonies and therefore the construction of the rule antecedent and consequent parts is independent of each other. hAM is yet another competitor to the proposed algorithm.

3. Ant colony optimization

Swarm Intelligence [16–18], which deals with the collective behavior of small and simple entities, has been used in many application domains. ACO, proposed in the early 90s [19–22], is one of the most famous meta-heuristic under the umbrella of Swarm Intelligence. Since its inception, ACO has been used to solve many complex problems including those related to data mining [23–25] as well as other combinatorial optimization problems. ACO is inspired by the food foraging behavior of biological ants [26]. Ants pass on the information about the trail they are following by spreading a chemical substance called pheromone, in their search environment. Ants communicate with each other by means of pheromone. Other ants that arrive in the vicinity are more likely to take the path with higher concentration of pheromone than the paths with lower concentrations. In other words, the desirability of possible paths is proportional to their pheromone concentrations. The pheromone evaporates with time and if new pheromone is not added, the older one dwindles away. This phenomenon has been modeled in the ACO meta-heuristic. The learning of proposed algorithm is based on ACO with following major requirements:

- The ability to represent a complete solution as a combination of different components.
- There should be a method to determine the fitness or quality of the solution.
- A heuristic measure for the solution's components (this is desirable but not necessary).

Suppose, we have a connected graph $G = (V, E)$ where $|V|$ denotes the total number of nodes/ vertices and $|E|$ total number of connecting edges in graph. The simple ant colony optimization meta-heuristic can be used to find the shortest path between a given source node 'V_s' and a given destination node 'V_d' in the graph 'G'. The path length is either given by the number of nodes on the path or summation of cost values on edges constituting the path. Each edge of the graph connecting the nodes 'V_i' and 'V_j' has a variable (artificial pheromone), which is modified by the ants when they visit the nodes [27].



Feature Space (Input)			Class to Predict		
-	-	-	ANIMAL	DOG	Doberman
-	-	-	ANIMAL	DOG	Terrier
-	-	-	ANIMAL	DOG	Rottweiler
-	-	-	ANIMAL	CAT	Ragdoll
-	-	-	ANIMAL	CAT	Sokoke
-	-	-	ANIMAL	CAT	Serengeti

Fig. 2. Hierarchical dataset given with partial animal CHS.

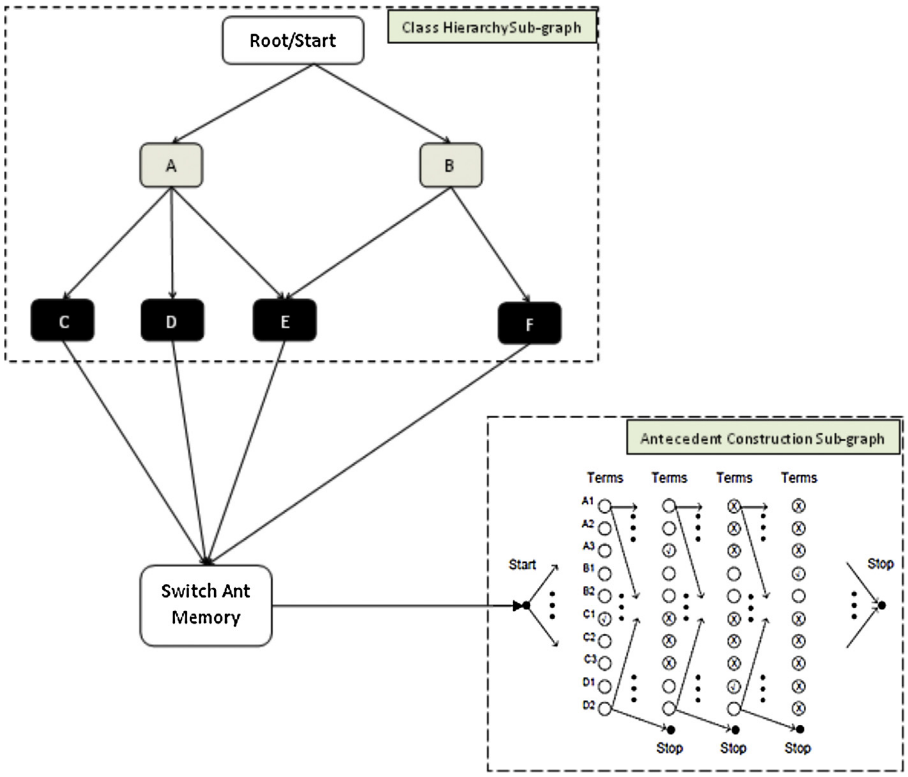


Fig. 3. An example search space of hAM-C.

From a node, when an ant decides which node to move next, it uses two parameters to calculate the probability of moving to a particular node; first, distance to that node and second, amount of pheromone on the connecting edge. Let $d_{i,j}$ be the distance between the nodes '*i*' and '*j*', the probability that the ant chooses '*j*' as the next node after it has arrived at city '*i*' where '*j*' is in the set '*S*' of cities that have not been visited [27].

$$p_{i,j} = \frac{[\tau_{i,j}]^{\alpha}[\eta_{i,j}]^{\beta}}{\sum_{k \in S}[\tau_{i,k}]^{\alpha}[\eta_{i,k}]^{\beta}} \tag{1}$$

where $\tau_{i,j}$ is the pheromone value on edge *e(i,j)* and $\eta_{i,j}$ is a heuristic value calculated as $1/d_{i,j}$. The parameters α and β are influencing factors of pheromone value and heuristic value respectively. The pheromone at edges is usually initialized with small random values at start. The complete route/tour of an ant from a source node to a destination node is called a solution to the problem at hand. The evaluation of a solution is done using a fitness function. Some best ants (having good solutions) or all ants modify the pheromone values on the edges added to their tour. One possible modification of the pheromone may be done as:

$$\tau_{i,j} = \tau_{i,j} + \frac{Q}{L} \tag{2}$$

where '*Q*' is some constant and '*L*' is the length of the tour, small the value of '*L*' high the pheromone value added to the previous pheromone value on an edge. With time, concentration of pheromone decreases due to diffusion affects; a natural phenomenon known as evaporation. This also ensures that old pheromone should not have a too strong influence on the future. So, with evaporation, chances to stuck at local minima is minimized in ACO. This evaporation can be performed as:

$$\tau_{i,j} = \tau_{i,j}.\rho \quad \{\text{Where } \rho \text{ will be between 0 and 1}\} \tag{3}$$

4. Method

In this section, we discuss different stages of the proposed ACO based hierarchical classification algorithm. We begin with the definition of the problem tackled in this work followed by a brief general description of the proposed algorithm. Afterwards, each and every stage of the approach is further discussed in a fair amount of details. The stages are: search space design, rule construction based on pheromone and a correlation based heuristic function, rule evaluation, rule pruning, pheromone update and evaporation, stopping criteria and formation of a default rule.

4.1. Problem definition

Given an input data set with a CHS, the goal is to build a list of IF–THEN rules, such that two objectives i.e. prediction accuracy and prediction specificity is maximized.

4.2. General description

The core of the algorithm is the incremental construction of a classification rule of the type

$$\text{IF} < \text{term}_1 \text{ AND term}_2 \text{ AND} \ldots > \quad \text{THEN} \\ < \text{class\_1}, \text{ class\_2}, \ \ldots \text{class\_}k > \tag{4}$$

Each term is an attribute-value pair related by an operator. In our current experiments, we use "=" as the only possible relational operator. An example term is "Color = red". The attribute's name is "color" and "red" is one of its possible values. Since we use only "=" operator, any continuous (real-valued) attribute present in the data, need to be discretized in a preprocessing step. The consequent of the rule contain class labels that are hierarchically related based on the given CHS. The stages constituting the proposed approach are discussed in the following subsections.

4.3. Search space and tour construction

The first stage in an ACO algorithm starts with designing a problem search space in which the ants conduct the search in order to find the candidate solutions. The search space for hAM-C (hierarchical AntMiner based on Correlation of Attributes) is defined with the help of input dataset and a CHS. As shown in Fig. 3, the overall search space is divided into two parts; "Class Hierarchy sub-graph" and "Antecedent Construction sub-graph". In the same way, an ant is equipped with two types of memories. First type of memory named "class memory" is used to save the classes selected during the tour of the ant in the class hierarchy sub-graph. The second type of memory named "antecedent memory" saves antecedent part of the rule during the ant tour in the antecedent construction sub-graph. In this way, the rule consequent and antecedent parts are constructed cooperatively, and collectively represents a complete tour of an ant. In order to use proposed heuristic function (Section 4.7), it is mandatory to construct the consequent of the rule before constructing its antecedent part. In other words, the selection of conditions to be included in the rule antecedent part is determined based on the committed class labels. The search space of hAM-C is modified and two sub-graphs are connected in order to serve this purpose (Fig. 3).

The class hierarchy sub-graph is equivalent to the CHS given with the problem data set. The topology of the antecedent construction sub-graph is the same as used in [28]. The antecedent construction sub-graph can be considered as a DAG, where the vertices are used to represent the terms (attribute–value pair) extracted from the input dataset. For constructing IF–THEN rules, the task of ants is to visit a vertex based on some heuristic (discussed later) and pheromone value found on the edges in the search space. The statistic about the total number of terms present in the dataset can be calculated as:

$$Total\_terms = \sum_{n=1}^{a} b_n \tag{5}$$

where 'a' is the total number of attributes (excluding the class attribute) and 'b_n' is the number of possible values in the domain of attribute 'A_n'. After a term has been selected (i.e. a vertex is visited), the attribute related to that term is flagged as used. During the tour of an ant, once an attribute is flagged used, no other term related to that attribute can be selected, further. This restriction is necessary; as we cannot allow conditions of the type "Weather = Cloudy **AND** Weather = Sunny" (both at the same time). In this way, an ant can pick any of the vertex (in the antecedent construction sub-graph) and there is no fixed ordering in which the vertices are required to be visited. This is to discourage the biasness of vertices ordering introduced in AntMiner+ [22] which is not a good strategy as discussed in [37].

When an ant starts it tour, the class memory is activated and the antecedent memory is deactivated. Ant starts it tour from the start vertex (in class hierarchy sub-graph) which is originally the root class label of the class hierarchy, present in the given CHS. As, the proposed algorithm handles single path hierarchical classification algorithm, only one node is allowed to be selected at every single level of the class hierarchy sub-graph. The selection of the nodes in the class hierarchy sub-graph is based on the heuristic values associated with the edges (discussed later). Once the ant reaches the node with the label "Switch Ant Memory", the antecedent memory is activated and class memory is switched off.

Each and every ant can select different sets of class labels as done in hAM [7]. However, this is against the main essence of cooperative learning of ACO. For example, each ant learning a separate goal, cannot meaningfully update the common search environment (by means of pheromone). In order to ensure that entire swarm learns the same set of class labels (i.e. common goal) during an iteration of the hAM-C algorithm (discussed later), only first ant traverses through the class hierarchy sub-graph and the class labels it selects is simply copied to the class memory part of all the remaining ants. Thus, all the ants searches for the terms to be added to the antecedent of the rule that best describes the selected class labels. All the ants are then placed at node "Switch Ant Memory" and the antecedent memories of all the ants are activated while the class memories of all the ants are deactivated. Now, all the ants traverse the antecedent construction sub-graph and fill up their antecedent memories.

4.4. Complete algorithm

The heart of the algorithm is the outer most WHILE loop (Fig. 4) where several candidate rules are constructed. Class labels are committed prior to the construction of the antecedent part of the rule. It is pertinent to note that all the ants search for the terms (to be added into their antecedent memories) that best describes and discriminate the same common class labels present in their class memories. During an iteration of the first inner REPEAT-UNTIL loop of the algorithm, several ants are sent to construct antecedent part of the candidate rules (second inner REPEAT-UNTIL loop). The quality of the candidate rules is determined after pruning and the iteration best rule is selected. This rule is used to update the pheromones of the trails in the antecedent construction graph; where the pheromone on the edges present in the rule are increased based on the quality of the rule. Later, evaporation is performed on all the trails of the antecedent construction graph (discussed later). Each new ant in an iteration of the second inner REPEAT-UNTIL loop is guided by the experience of the ants in previous iteration (of the first inner REPEAT-UNTIL loop), available in the form of pheromone values. The best rule is updated if the iteration best rule is better than the best rule found so far. After several iterations of the first inner REPEAT-UNTIL loop, the search process converges. The convergence is declared by analyzing that the iteration best rules are equivalent over iterations equal to a user defined parameter *no_rules_converge* or in other words, best rule is not updated during last few iterations equal to *no_rules_converge*. In this way, early convergence is possible before executing total number of iterations given as an input by user. The learning process also stops if the size of the training set has reduced enough and contain training instances that are no more than a user defined parameter *Max_Uncovered_Cases*.

4.5. Pheromone initialization

The initial pheromone between two terms $term_i$ and $term_j$ belonging to two different attributes is computed as:

$$\tau_{ij}(iter = 1) = \frac{1}{\sum_{n=1}^{a} x_n b_n} \tag{6}$$

where x_n is set to 0 if $term_i$ belongs to attribute A_n, otherwise, it is set to 1. This method of pheromone initialization is adopted from AntMiner-C [28].

4.6. Term/class selection

An ant incrementally add terms in the antecedent part of the rule that it is constructing during its tour in the antecedent construction

```
DiscoveredRuleList = {}; /* rule list is initialized as empty list */
TrainingSet = {all training samples};
Construct the class hierarchy sub-graph;
WHILE( |TrainingSet| > Max_uncovered_samples)  /*main WHILE loop (core of hAM-C)*/
    Calculate heuristic values for all the edges of class hierarchy sub-graph;
    Send an ant to construct consequent of the rule from class hierarchy sub-graph;
    Copy the selected class labels to the class memories of all the ants;
    De-active class memories and activate antecedent memories of all the ants;
    Construct the antecedent sub-graph for reduced training set;
    Initialize pheromone values on all the edges of antecedent construction sub-graph;
    Calculate heuristic values for all the edges of antecedent construction sub-graph for the selected
    class labels;
    iter = 1;           /* counter for iterations */
    best_rule = {};
    rcc = 1;  /* counter for rule convergence test */
    REPEAT /*first inner REPEAT loop*/
        Iteration_best_rule = {};
        t = 1;      /* counter for ants */
        REPEAT /*second inner REPEAT loop*/
            Send Ant t to construct antecedent of the rule $R_t$ for the selected class labels;
            Prune rule $R_t$ and asses its quality;
            IF ($R_t$ is better than Iteration_best_rule)
                THEN Iteration_best_rule = $R_t$;
                IF (Iteration_best_rule is better than best_rule)
                    THEN best_rule = Iteration_best_rule;
                END IF
            END IF
            t = t + 1;
        UNTIL ($t \geq$ swarm_size)
        update pheromone of all the antecedent sub-graph trails based on Iteration_best_rule;
        IF (Iteration_best_rule == best_rule) /* update convergence test */
            THEN rcc = rcc + 1;
            ELSE rcc = 1;
        END IF
        iter = iter + 1;
    UNTIL (iter $\geq$ total_iterations) OR (rcc $\geq$ no_rules_converge)
    Prune the best best_rule and asses its quality;
    Add the pruned best_rule to DiscoveredRuleList;
    Remove the training samples covered by the pruned best_rule;
END WHILE
Add a default rule in the DiscoveredRuleList:
Prune the DiscoveredRuleList; (Optional)
```

Fig. 4. hAM-C Algorithm.

sub-graph. The selection of the next term is subject to the restriction that a term from its attribute A_n should not already be present in the current partial rule. In other words, once a term has been included in the rule then no other term from that attribute can be considered onwards. Subject to this restriction, the probability of selection of a term for addition in the current partial rule is given by the equation:

$$P_{ij}(t, iter) = \frac{\tau_{ij}{}^{\alpha}(iter)\eta_{ij}{}^{\beta}(s)}{\sum_{j=1}^{Total\_terms} x_j\{\tau_{ij}{}^{\alpha}(iter)\eta_{ij}{}^{\beta}(s)\}} \quad (7)$$

where $\tau_{ij}(iter)$ is the amount of pheromone associated with the edge between $term_i$ and $term_j$ for the current iteration equal to '*iter*', $\eta_{ij}(s)$ is the value of the heuristic function for the current iteration s of the main WHILE loop (constant/similar for a batch of ants), x_j is a binary variable that is set to 1 if $term_j$ is selectable, otherwise it is 0. Selectable terms are those which belong to attributes which have not become prohibited due to the selection of a term belonging to them. The denominator is used to normalize the numerator value for all the possible choices.

The parameters α and β are used to control the relative importance of the pheromones and heuristic values in the probability determination of next movement of the ant. For the selection of nodes or class labels in the class hierarchy sub-graph, the Eq. (7) can be modified as follows:

$$P_{ij}(t = 1) = \frac{\eta_{ij}}{\sum_{j=1}^{|E_i|} x_j\{\eta_{ij}{}^{\beta}\}} \quad (8)$$

where η_{ij} is the value of the heuristic function on the edge between $node_i$ and $node_j$ where the node is a class label in the class hierarchy sub-graph and $node_j$ must have IS-A relationship with $node_i$ based on the given CHS. x_j is a binary variable that is set to 1 if *class* $node_j$ is present in the current training set, otherwise it is 0.

4.7. Antecedent heuristic function

The heuristic value of an edge gives an indication of its usefulness and thus provides a basis to guide the search process. The heuristic function (Eq. (9)) for antecedent construction sub-graph is based on the correlation of the most recently chosen term (in the tour of the current ant) with other candidate terms in order to guide the selection of next term. In order to make most reasonable (and possibly most accurate) next move, ants can see backward toward the moves it already made. This ensures that the next move would be appropriate to the current context rather than the context at the start of the search process. This novelty is introduced to accommodate the limitation of hAM [7] where no historical information is used. For instance, the decision (generated by a brain) about how fast to run (or when to change direction), made by a Cheetah while preying a Deer is based on the context at that time and current position of the Cheetah rather than the position from where the Cheetah actually started his deadliest action. Same is the case for hill climbing and many of the other real world problems. So, with this motivation, we use history of the ant partial tour to guide it further in the search space. Only single recent term selected by an ant is considered for making a decision about the selection of next term within the antecedent construction sub-graph. This makes the heuristic calculation computationally less demanding as compared to analyzing all the terms present in the current partial tour of an ant.

$$\eta_{ij} = \sum_{k=1}^{L} \frac{|term_i, term_j, class_k|}{|term_i, class_k|} \cdot \frac{|term_j, class_k|}{|term_j|} \tag{9}$$

Here L is the total number of class labels present in the class memory of an ant and $k \in [1\ L]$. The most recently chosen term is $term_i$ and the term being considered for selection is $term_j$. The number of uncovered training samples having $term_i$ and $term_j$ and which belong to the class label k in the committed labels L of the rule is given by $|term_i, term_j, class_k|$. This number is divided by the number of uncovered training samples which have $term_i$ and which belong to $class_k$ to find the correlation between the terms $term_i$ and $term_j$. This correlation value is finally multiplied with the probability of $term_j$ having class k. The probability of $term_j$ having class k is the weight of $term_j$ in order to correctly classify the instances having class k. The value of the heuristic function is zero for edges between terms of the same attribute.

The heuristic values on the edges between the Start node and the first layer terms of the antecedent construction sub-graph are calculated on the basis of the following Laplace-corrected confidence [25] of a term:

$$\eta_{Start,j} = \frac{\sum_{k=1}^{L}(|term_j, class_k| + 1)/(|term_j| + m)}{L} \tag{10}$$

where m is the number of classes present in the dataset. This heuristic function has the advantage of penalizing the terms that would lead to very specific rules and thus helps to avoid over-fitting. For example, if a term occurs in just one training sample and it class labels are the same as chosen class labels, then its confidence is 1 without the Laplace correction (i.e. without adding 1 and m in the Eq. (10)). With Laplace correction, its confidence is 0.66 if there are two classes in the data. Before usage, the values obtained by Eq. (10) are normalized by the summation of all such values between the Start node and the first layer of terms.

4.8. Consequent heuristic function

The heuristic function to be used in the class hierarchy sub-graph is based on the frequency of class labels and is calculated as:

$$\eta_{ij} = |Class_j| \tag{11}$$

where η_{ij} is the value of the heuristic function on the edge between $node_i$ and $node_j$ and $|class_j|$ is the number of training instances that contain $class_j$ in their class domain. The heuristic function has a bias toward class labels that have a greater number of examples, which therefore will initially favor the discovery of rules with the class labels with higher frequencies [7]. However, due to the use of a sequential covering procedure, such instances will be removed from the training set based on the coverage of the predicted rules; thus, class labels with lower frequencies will eventually get their chance of selection.

4.9. Rule quality

Once the rule is constructed by an ant, its quality is determined which is used to update the pheromone values in the search space. An appropriate quality measure is essential in order to influence the search process in right direction. The quality of the hierarchical classification rule is based on a modified hierarchical measure proposed in [29] and used in [7]. The evaluation measure takes into account both the hierarchical precision and recall and it also keep in consideration the IS-A relationship among the class labels.

As discussed earlier, the consequent of the rule is the trail of an ant in the class hierarchy sub-graph from root class node to one of the leaf class nodes. For tree based CHS, all the ancestors of a predicted class label will be present in the consequent of the rule. However, for a CHS represented by a DAG (where multiple paths between a given pair of class nodes can exist), all the ancestors of a predicted class label are explicitly added as:

$$L_{ex} = L \cup \{\forall l_i \in L \quad Ancestors \quad (l_i)\} - root \tag{12}$$

where L represents the original predicted class labels during the tour of an ant in the class hierarchy sub-graph. L_{ex} is the expanded class labels such that all the ancestors of a predicted class label $l_i \in L$ denoted as $Ancestors(l_i)$ are also included in the final set of predicted class labels. The hierarchical macro-averaged measures of precision and recall are calculated as:

$$hP = \frac{\sum_{i \in Cov_r}(|T_i \cap L_{ex}|)/(|L_{ex}|)}{|Cov_r|} \quad \text{and} \quad hR = \frac{\sum_{i \in Cov_r}(|T_i \cap L_{ex}|)/(|T_i|)}{|Cov_r|} \tag{13}$$

where Cov_r is the set of all training instances covered (satisfying the rule antecedent part) by the rule r. T_i is the set of true class labels of the *ith* instance. The hierarchical precision represents the average number of true class labels correctly predicted by rule r, divided by the total number of class labels present in L_{ex}. The hierarchical recall is the average number of predicted class labels predicted correctly by rule r, divided by the total number of true class labels that should be predicted for Cov_r.

Both of these measures, hP (hierarchical precision) and hR (hierarchical recall), are combined (using Eq. (14)) to calculate the hF (hierarchical F-measure) value. The hF value is used as the quality criteria for the hierarchical rule 'r'.

$$Q_r = hF = \frac{2\ .\ hP\ .\ hR}{hP + hR} \tag{14}$$

```
best_rule = rule_r;
$q_{best}$ = Q (rule_r);

REPEAT /*first outer REPEAT loop*/
        rule_i = best_rule;
        rule_i→antecedent = remove_last_term ( rule_i→antecedent );
        $q_i$ = Q (rule_i);
        $consequent_c$ = rule_i→consequent;
        REPEAT /*first inner REPEAT loop*/
                $consequent_c$ = remove_last_class ( $consequent_c$ );
                rule_c→antecedent = rule_i→antecedent;
                rule_c→consequent = $consequent_c$;
                IF ( Q(rule_c) > $q_i$) THEN
                        rule_i = rule_c;
                        $q_i$ = Q(rule_c);
                END IF
        UNTIL ( $|consequent_c|$ == 1)
        IF ($q_i \geq q_{best}$) THEN
                best_rule = rule_i;
                $q_{best}$ = $q_i$;
        END IF
UNTIL ( $q_i < q_{best}$ OR |best_rule→antecedent| == 1)
```

Fig. 5. Pseudo code for rule pruning procedure.

4.10. Rule pruning

Is it possible to further improve the quality of a discovered rule? One possibility is to apply the pruning process on the rule soon after it gets discovered (and before adding it to the discovered rule list). The pruning process is intended to improve the quality of a rule after removing irrelevant or copious terms and/or class labels from the antecedent and consequent parts of the rule, respectively. It also makes the rules compact in size which adds to the comprehensibility power of the rules-based classification algorithms. We have adopted the pruning process (Fig. 5) as used in the research work of Fernando et al. [7].

4.11. Pheromone update and evaporation

The decision of an ant about making it moves in the search space is controlled with two important parameters as shown in the Eqs. (7) and (8). Pheromone values on the edges are an important learning dynamic for the ACO algorithm. The quality of ant trails is used to make efficient use of the pheromone values. The pheromone values on the edges are updated in proportion to the quality of the trails and thus define the learning directions for the subsequent transitions of the entire swarm.

For hAM-C, the pheromone values in antecedent construction graph are stored as a square matrix where one dimension of the matrix is equal to the number of total terms present in the given dataset. The pheromone matrix is interpreted as an asymmetric matrix which helps in exploring the search space in great details. An example pheromone matrix is shown in Fig. 6. The pheromone matrix is asymmetric and captures the fact that pheromone values on edges originating from a term to other terms are kept the same in all the layers of the antecedent construction graph. In other words, the same matrix is valid for any pair of the two consecutive layers. The matrix is asymmetric because for a constructed rule with $term_i$ occurring immediately before $term_j$, the pheromone on edge between $term_i$ and $term_j$ ($\rightarrow t_i t_j$) is updated but the pheromone on edge between $term_j$ and $term_i$ ($\rightarrow t_j t_i$) is not updated.

During the first iteration of the first inner repeat loop of the hAM-C algorithm, the pheromone values on the edges of the antecedent construction graph are the ones initialized with the help of pheromone initialization procedure. The pheromone values in

	A1	A2	A3	B1	B2	C1	C2	C3	D1	D2
S	τ_{01}	τ_{02}	τ_{03}	τ_{04}	τ_{05}	τ_{06}	τ_{07}	τ_{08}	τ_{09}	τ_{010}

(a)

	A1	A2	A3	B1	B2	C1	C2	C3	D1	D2
A1	0	0	0	τ_{14}	τ_{15}	τ_{16}	τ_{17}	τ_{18}	τ_{19}	τ_{110}
A2	0	0	0	τ_{24}	τ_{25}	τ_{26}	τ_{27}	τ_{28}	τ_{29}	τ_{210}
A3	0	0	0	τ_{34}	τ_{35}	τ_{36}	τ_{37}	τ_{38}	τ_{39}	τ_{310}
B1	τ_{41}	τ_{42}	τ_{43}	0	0	τ_{46}	τ_{47}	τ_{48}	τ_{49}	τ_{410}
B2	τ_{51}	τ_{52}	τ_{53}	0	0	τ_{36}	τ_{37}	τ_{38}	τ_{39}	τ_{310}
C1	τ_{61}	τ_{62}	τ_{63}	τ_{64}	τ_{65}	0	0	0	τ_{69}	τ_{610}
C2	τ_{71}	τ_{72}	τ_{73}	τ_{74}	τ_{75}	0	0	0	τ_{79}	τ_{710}
C3	τ_{81}	τ_{82}	τ_{83}	τ_{84}	τ_{85}	0	0	0	τ_{89}	τ_{810}
D1	τ_{91}	τ_{92}	τ_{93}	τ_{94}	τ_{95}	τ_{96}	τ_{97}	τ_{98}	0	0
D2	τ_{101}	τ_{102}	τ_{103}	τ_{104}	τ_{105}	τ_{106}	τ_{107}	τ_{108}	0	0

(b)

Fig. 6. The pheromone values for the antecedent construction graph of example problem given in the Fig. 3. The elements of (a) are the pheromone values on edges from Start node to nodes of the first layer of terms in the antecedent construction graph. The elements of (b) are pheromone values on the edges between the terms present in any of the two consecutive layers of the antecedent construction graph. For example, the first row elements are the pheromone values for edges from the term A1 to all the other terms in the next layer. These values are the same for edges between 1st and 2nd layer, 2nd and 3rd layer, and so on. If an ant chooses C1, A3, D1 and B1 terms in its trail, the elements τ_{06}, τ_{63}, τ_{39} and τ_{94} are updated according to Eq. (15). For normalization, each pheromone element in a row is divided with the summation of all pheromone values in the same row.

the antecedent construction graph remain unchanged during the execution of the second inner repeat loop. In other words, all the ants have the same environment in terms of pheromone values. After completion of the second inner loop, the iteration best trail is used to update the pheromone matrix. In this way, the experience of the swarm in the iteration '*t*' of the first inner repeat loop is made available to the swarm in next iteration i.e. '*t* + *1*' of the first inner repeat loop. It is assumed that each new iteration of the first inner repeat loop, results in a better experience of the swarm toward the problem search space. The amount of pheromone on the edges between consecutive terms occurring in the iteration best trail is updated as:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + (1-(1/1+Q))\tau_{ij}(t) \qquad (15)$$

where $\tau_{ij}(t)$ is the pheromone value encountered in iteration '*t*' (of the first inner repeat loop) between $term_i$ and $term_j$. The pheromone evaporation rate is represented by ρ and Q is the quality of the current iteration best trail.

Eq. (15) updates pheromones by first evaporating a percentage of the previously seen pheromone and then adding a percentage of the pheromone dependent on the quality of the iteration best rule. If the rule is well representative to the data available in the current training set, the quantity of pheromone added is greater than the pheromone evaporated and the terms constituting the rule become more attractive for the ants in the subsequent iterations. The evaporation process (as performed in Eq. (15)) improves exploration, otherwise in the presence of a static heuristic function, the ants tend to converge quickly to the terms selected by the entire swarm during the first few iterations of the first inner repeat loop.

The next step is to normalize the pheromone values. Each pheromone value is normalized by dividing it with the summation of pheromone values on edges of all its competing terms. In the pheromone matrix (Fig. 6), normalization of the elements is

Table 1
Summary of the differences between hAM-C and hAM.

hAM-C	hAM
Utilizes a single ant colony	Utilizes two ant colonies
A search space with two connected sub-graphs	Two separate search graphs
Consequent is committed beforehand in order to construct the antecedent of the rule	Construction of antecedent and consequent part is independent and irrelevant
Hierarchical correlation based heuristic function for which the history of ant tour is important	Hierarchical information gain where terms interactions are ignored
Utilizes an asymmetric pheromone matrix	Utilizes a symmetric pheromone matrix
In the main iteration of the algorithm, all the ants perform learning for a single set of class labels	During an iteration, every single ant might be learning for a different set of class labels
No direct support for continuous attributes	It can handle continuous attributes

done by dividing them with the summation of values of the row to which they belong. Note that for the rows where no change in the pheromone values has occurred (after applying Eq. (15)), the normalization process yields the same values as before. This process changes the amount of pheromone associated with those terms that are not present in the iteration best rule but are the competitors of the selected terms.

4.12. Default rule

A rule, predicted during the execution of the algorithm, is restricted to cover at least a number of training instances equivalent to a user defined parameter i.e. *Min_Cases_Per_Rule*. This restriction helps in avoiding too specific rules that would cause over-fitting problem. Based on this restriction, there is a possibility that no rule is predicted during an iteration of the outer most WHILE loop. This usually happens when the training set is reduced enough and contain very few instances and there in no term available to be added to the rule which covers instances equal to *Min_Cases_Per_Rule*. In such a situation, learning is stopped and a rule with empty antecedent part and selected class labels is added to the discovered rule list (rule with empty antecedent, covers all the training instances). It is also important to note that the coverage of a rule in hAM-C, has different meaning as compared to flat classification based AntMiner's. For flat classification based AntMiner's, a rule is consider to cover a training sample, if the sample values is in accordance with the conditions of the rule antecedent part and the class value of the sample also matches the rule consequent part. It is however not straight forward to find a hierarchical rule with the consequent exactly matching with the class values of a training sample. Therefore, for the coverage of a hierarchical rule, consequent part is not checked against the class values of training sample.

Another such situation (for stopping the execution of algorithm) is faced when the size of the training set has reduced enough and contain training instances that are no more than a user defined parameter *Max_Uncovered_Cases*. In this case, a default rule is added to the discovered rule list. The default rule has an empty antecedent part and therefore covers all the training instances in the current training set. If only one class label is left (i.e. root class label) in the current training set, it is assigned as the consequent of the default rule. Otherwise, the class label (except root) that is occurring in majority of the current training instances, is selected as the consequent of the default rule. If there is a tie between two or more class labels, class label that is at lower level of the class hierarchy is preferred. The class label at the lower level is preferred to improve the prediction specificity. However, in a situation when the class labels that are in tie are at the same level of the class hierarchy, a single class label in tie is selected, randomly.

After the algorithm finished its learning, the discovered rules are used (in order) to classify the test examples, In particular, a test example is selected and given to the model, generated by training phase. The rule antecedent part is checked against the given test example, starting from the first predicted rule and continuing the search on the predicted rule list (in sequence), until a matching rule is found. At the end, consequent of the selected rule is assigned to the test example.

4.13. Overview of differences between hAM-C and hAM

The proposed algorithm has few parts in common to that of hAM; as all of the current versions of ant-based classification model are originally inspired from the work of Parpinelli et al., known as AntMiner [14], thus share a number of common traits. In Table 1, we summarize some of the main differences between hAM and hAM-C (proposed algorithm). Some of these key differences are empirically analyzed in Section 5.3.

5. Results and discussion

In this section, we present the simulation results of our proposed method (hAM-C) in comparison with another hierarchical single path classification ACO based algorithm (hAM), proposed in the work of Otero et al. [7]. The proposed algorithm is implemented in the Microsoft Visual Studio (2008) development environment using C-Sharp language. On the other hand, for hAM [7], JAVA based implementation is kindly made available by Otero. All the

Table 2
Summary of the datasets used in the experiments.

Class hierarchy type	Dataset	Attributes	# Continuous attributes	# Instances	Total classes
DAG	Ds1_aa	22	22	147	19
	Ds1_interpro	92	2	147	19
	Ds2_aa	22	22	371	17
	D2_interpro	155	2	359	17
Tree	Glass	9	9	214	12
	Bridge	12	4	108	08
	AA_tree	22	22	147	19
	Interpro_tree	92	2	147	19

Table 3
Class hierarchical structure of all the datasets.

Ds1_aa & Ds1_interpro datasets class hierarchy Node {list of direct parent(s)}	Ds2_aa & Ds2_interpro datasets class hierarchy Node {list of direct parent(s)}
GO:0005215 @Root	GO:0005216 @Root
GO:0015267 {GO:0005215}	GO:0005253 {GO:0005216}
GO:0015075 {GO:0005215}	GO:0005261 {GO:0005216}
GO:0005342 {GO:0005215}	GO:0015276 {GO:0005216}
GO:0005275 {GO:0005215}	GO:0005244 {GO:0005216}
GO:0015268 {GO:0015267}	GO:0005254 {GO:0005253}
GO:0008324 {GO:0015075}	GO:0005272 {GO:0005261}
GO:0008509 {GO:0015075}	GO:0005227 {GO:0005261}
GO:0005216 {GO:0015268,GO:0015075}	GO:0005267 {GO:0005261}
GO:0046943 {GO:0005342}	GO:0005262 {GO:0005261}
GO:0015171 {GO:0005275,GO:0046943}	GO:0004889 {GO:0005261,GO:0005231}
GO:0005261 {GO:0008324,GO:0005216}	GO:0005230 {GO:0015276}
GO:0005253 {GO:0008509,GO:0005216}	GO:0022843 {GO:0005244}
GO:0005244 {GO:0005216}	GO:0005249 {GO:0022843,GO:0005267}
GO:0015276 {GO:0005216}	GO:0005245 {GO:0022843,GO:0005262}
GO:0005262 {GO:0005261}	GO:0005231 {GO:0005230}
GO:0005267 {GO:0005261}	GO:0005242 {GO:0005249}
GO:0005245 {GO:0005262,GO:0005244}	
GO:0005249 {GO:0005267,GO:0005244}	

Glass dataset class hierarchy Node {list of direct parent(s)}		Bridge dataset class hierarchy Node {list of direct parent(s)}	
Glass	@Root	Type	@Root
window	{Glass}	WOOD	{Type}
non-window	{Glass}	SUSPEN	{Type}
float	{window}	ARCH	{Type}
non-float	{window}	TRUSS	{Type}
5	{non-window}	CANTILEV	{TRUSS}
6	{non-window}	CONT-T	{TRUSS}
7	{non-window}	SIMPLE-T	{TRUSS}
1	{float}		
2	{non-float}		
3	{float }		
4	{ non-float}		

experiments are conducted on an Intel Core i3 Processor with a CPU clock rate of 2.4 GHz and 2 GB of main memory. This section is further divided in several subsections as follows. First, we discuss the Database/data set generation method and evaluation methodology used in our experimental results. Second, the effectiveness of the proposed technique is evaluated based on tenfold cross validation. In order to analyze the statistical significance between the performances of two competing algorithms, two tailed Student's *t*-test is conducted and performance differences are reported. Afterwards, the overall model complexities of the algorithms is compared using two tailed Wilcoxon signed rank test. Experiments are conducted for a decision tree classifier that works based on LCN strategy and overall comparison of all the algorithms is reported. At the end, several design choices for hAM-C algorithm are empirically investigated in order to observe the significance of the differences with hAM algorithm.

5.1. Dataset generation and evaluation methodology

In order to evaluate the proposed method, six ion-channels protein functions data sets (related to protein function prediction) and two publicly available data sets are collected. The datasets related to ion-channels protein functions are kindly provided by Fernando E.B. Otero [7]. The remaining two publically available datasets (glass and bridge) are collected from the UCI Machine Learning archive [31]. The datasets glass and bridge are usually used to evaluate the performance of a flat classification algorithm. However, the CHS is also available with these datasets in order to format the datasets to be used as benchmark for hierarchical classification algorithms [1]. The detail of datasets is provided in Table 2. The class labels of all of these datasets are organized based on DAG hierarchy except the bridge and glass datasets where tree representation is used. The description of CHS, associated with these datasets, is summarized in Table 3. The discussion on why the analysis of ion-channel proteins is important and how these datasets [7] are created is provided as follows. The proteins related to ion-channels are present in all biological (living) cells and considered a pore forming protein across the cell membrane [30]. The ion channels are the integral membrane proteins and allows specific inorganic ions (e.g. K^{+}, Na^{+}, Ca^{2+}, Cl^{-}) to cross/pass through the cell membrane and play an important role in many cell functions, such as in functions related to the muscle contraction, nervous system and T-cell activation [7]. It is therefore essential to analyze the ion-channels proteins functions in order to improve biological knowledge, diagnosis and treatment of diseases and understanding the working of organisms at micro level.

As, it is not possible to analyze the entire GO hierarchy at once, first, a subset of the hierarchy is selected. Specifically, the direct

```
IF
        IPR001993 = 'no'   AND   IPR002394 = 'no'          AND
        IPR002293 = 'no'   AND   IPR001991 = 'no'          AND
        IPR001905 = 'no'   AND   IPR002946 = 'no'          AND
        IPR001807 = 'no'   AND   weight = '-inf-66437.9'   AND
        IPR000900 = 'no'   AND   IPR001925 = 'no'          AND
        IPR011701 = 'no'   AND   IPR001873 = 'no'
THEN
        'GO:0005215, GO:0015075, GO:0005216,
        GO:0015268, GO:0015267, GO:0005261,
        GO:0008324, GO:0005262, GO:0005245, GO:0005244'
```

Fig. 7. An expanded pruned rule for a fold result of DS1_interpro dataset (hAM-C).

Table 4
Parameters used in experiments.

Parameter	Value
Number of ants	20
Max. uncovered samples	10
Evaporation rate	0.15
No. of rules converged	10
Minimum cases per rule	5
Maximum iterations	500
Alpha	1
Beta	1

ancestors and all the descendents GO terms of ion-channel activity (GO:0005216) are selected to form the CHS to be predicted [7]. Next, the protein interaction data from the IntAct database is retrieved considering database cross-references for all the selected GO terms. The GO terms with less than 10 protein examples are ignored. In order to collect predictor attributes, the amino acid sequence and InterPro pattern references are collected for each protein example from the UniProt database (release 12.0), using the database cross-reference to UniProt found in IntAct protein example records. In this way, 147 protein examples involving 19 (2 ignored) GO terms are collected. The dataset 'Ds1_aa' & 'Ds2_aa' is composed of each of the 20 amino acid composition information as predictor attributes, containing the percentage of the sequence as continuous values [7]. The dataset 'Ds1_interpro' & 'Ds2_interpro' used the InterPro pattern information as predictor attributes, containing Boolean attributes, representing the presence or absence of a particular InterPro pattern. The datasets 'Ds2_aa' and 'Ds2_interpro' are formed without the restriction of selecting proteins with protein interaction data. Two additional tree based datasets are formed, namely 'AA_tree' and 'Interpro_tree'. The attributes of the 'AA_tree' dataset is based on amino acid sequence. Datset 'AA_tree' is formed in the same way as to that of 'Ds1_aa' dataset with the difference that only the first single direct parent of a particular class label is considered in order to ignore DAG based relationships. The 'Interpro_tree' dataset is based on InterPro pattern references and formed in the same way as to that of 'Ds1_interpro' dataset with the difference that only the first single direct parent of a particular class label is considered and multiple parents are ignored. After creating these datasets, two more predictor attributes (namely, sequence length and molecular weight) derived from the proteins primary sequences are added to all of these datasets. For more details, readers are kindly referred to [7].

The evaluation of hAM-C experiments is performed using 10-fold cross validation and hF (hierarchical F-Measure) values analysis. In particular, a data set is randomly partitioned into 10 non-overlapping and mutually exclusive subsets. For the experiment of fold *i*, subset *i* is selected as testing set and the remaining nine subsets are used to train the classifier. Using 10-fold cross validation experiments, the performance of the hAM-C can be confirmed against any kind of the selection biased of the samples for training and testing phases. In order to have a fair comparison, same 10-fold cross validation subsets are used for all the data sets. Moreover, the model size (in terms of number of rules and number of terms per rules) of the two competing algorithms are also compared to analyze the model complexities. An example hierarchical IF–THEN rule, predicted by hAM-C for DS1_interpro dataset, is shown in Fig. 7. At the end, the number of ants originally used to predict a single rule in the discovered rule list and total probability calls evaluated during the execution of the proposed algorithm are reported.

There are six user defined parameters associated with hAM-C: swarm size, maximum uncovered samples, evaporation rate, convergence counter threshold, alpha and beta. The values of these parameters are given in Table 4. These values have been chosen because they seems to provide reasonable performance as reported in the earlier versions in literature [7]. Optimization of these parameters is avoided, as our goal is to evaluate the parameter settings generalization ability across a wide range of data sets.

5.2. Experimental results over datasets

In this section, the results of hAM-C are compared with hAM [7] on the basis of hF values and model size. The proposed algorithm works with categorical attributes and continuous attributes need to be discretized in a preprocessing step. An unsupervised discretization filter of Weka-3.4 machine learning tool [25] is used for discretizing continuous attributes. This filter first computes the intervals of continuous attributes from the dataset and then uses these intervals during discretization procedure. On the other hand, hAM is capable of handling both the categorical as well as continuous values. We therefore conducted the experiments using hAM for discretized (hAM-Disk) and non-discretized (hAM-Cont for continuous valued attributes) datasets, separately.

Table 5 presents the results concerning the hR (hierarchical Recall), hP (hierarchical Precision) and hF (hierarchical F-Measure) values where the higher the hF values the better the classification performance is. Moreover, all the algorithms are assigned with the average ranking based on a nonparametric Friedman test with post hoc Holm test [32,33] based on hF values for all the datasets. The Friedman test is chosen because it does not make any assumptions about the normal distribution of the underlying data (a requirement for equivalent parametric tests) and it is a recommended and suitable test to compare a set of classifiers over multiple data sets, according to the guidelines presented in [32,33]. The lower the average ranking the better is the performance of the algorithm. A single value in Tables 5, 7 and 9 corresponds to the average value obtained using tenfold cross validation followed by the standard error in the form (average ± standard error). Table 6 presents the comparison of significance differences between the tenfold hF values (over a dataset) of the two competing algorithms based on two tailed Student's *t*-test [34] at the 0.01 significance level. The null hypothesis is that the difference between the average hF values of

Table 5
Comparison of hAM-C with hAM based on hF values.

Dataset	hAM-continuous data (average rank = 2.0)			hAM-discretized (average rank = 2.31)			Proposed hAM-C (average rank = 1.68)		
	hP	hR	hF	hP	hR	hF	hP	hR	hF
Ds1_aa	0.55 ± 0.04	0.70 ± 0.04	0.61 ± 0.04	0.49 ± 0.06	0.53 ± 0.12	0.50 ± 0.07	0.71 ± 0.02	0.63 ± 0.05	**0.66 ± 0.03**
Ds1_interpro	0.60 ± 0.03	0.82 ± 0.03	0.69 ± 0.02	0.63 ± 0.06	0.83 ± 0.04	0.71 ± 0.04	0.82 ± 0.03	0.78 ± 0.02	**0.80 ± 0.02**
Ds2_aa	0.55 ± 0.04	0.73 ± 0.04	**0.62 ± 0.04**	0.58 ± 0.06	0.55 ± 0.05	0.56 ± 0.05	0.56 ± 0.03	0.51 ± 0.02	0.54 ± 0.02
D2_interpro	0.83 ± 0.02	0.80 ± 0.01	0.81 ± 0.01	0.83 ± 0.06	0.81 ± 0.03	**0.82 ± 0.03**	0.78 ± 0.08	0.80 ± 0.06	0.79 ± 0.06
Glass	0.83 ± 0.02	0.82 ± 0.02	0.82 ± 0.02	0.58 ± 0.05	0.47 ± 0.06	0.51 ± 0.06	0.84 ± 0.01	0.86 ± 0.01	**0.85 ± 0.01**
Bridge	0.65 ± 0.04	0.66 ± 0.04	0.66 ± 0.04	0.69 ± 0.04	0.70 ± 0.04	0.69 ± 0.04	0.69 ± 0.02	0.70 ± 0.03	**0.69 ± 0.02**
AA_tree	0.47 ± 0.03	0.50 ± 0.04	0.48 ± 0.03	0.43 ± 0.02	0.45 ± 0.03	0.43 ± 0.02	0.65 ± 0.04	0.49 ± 0.03	**0.56 ± 0.03**
Interpro_tree	0.59 ± 0.03	0.63 ± 0.02	0.61 ± 0.02	0.58 ± 0.01	0.66 ± 0.02	0.62 ± 0.02	0.75 ± 0.03	0.58 ± 0.02	**0.65 ± 0.02**

The values where the algorithms perform the best are highlighted as bold.

Table 6
Comparison based on two tailed Student's *t*-test at 0.01 significance level (hF).

Hypothesis	Dataset	*t-Value*	*p-Value*	Null hypothesis (H_0)
hAM-C vs. hAM-Cont	Ds1_aa	0.888266629	3.86×10^{-01}	–
	Ds1_interpro	3.136265724	$\mathbf{5.71 \times 10^{-03}}$	Reject▲
	Ds2_aa	2.003902882	6.04×10^{-02}	–
	D2_interpro	1.274774214	2.19×10^{-01}	–
	Glass	1.37508549	1.86×10^{-01}	–
	Bridge	0.723433495	4.79×10^{-01}	–
	AA_tree	1.539743226	1.41×10^{-01}	–
	Interpro_tree	1.387863284	1.82×10^{-01}	–
hAM-C vs. hAM-Disk	Ds1_aa	3.820856906	$\mathbf{1.25 \times 10^{-03}}$	Reject▲
	Ds1_interpro	2.88040733	$\mathbf{9.96 \times 10^{-03}}$	Reject▲
	Ds2_aa	0.872333251	3.95×10^{-01}	–
	D2_interpro	1.41793595	1.73×10^{-01}	v
	Glass	6.050119463	$\mathbf{1.02 \times 10^{-05}}$	Reject▲
	Bridge	0.093329811	9.27×10^{-01}	–
	AA_tree	3.149423884	$\mathbf{5.55 \times 10^{-03}}$	Reject▲
	Interpro_tree	1.093361243	2.89×10^{-01}	–
hAM-Cont vs. hAM-Disk	Ds1_aa	2.460562637	2.42×10^{-02}	–
	Ds1_interpro	0.926591608	3.66×10^{-01}	–
	Ds2_aa	1.534480411	1.42×10^{-01}	–
	D2_interpro	0.303178962	7.65×10^{-01}	–
	Glass	5.314157269	$\mathbf{4.73 \times 10^{-05}}$	Reject▲
	Bridge	0.685950055	5.01×10^{-01}	–
	AA_tree	1.349933104	1.94×10^{-01}	–
	Interpro_tree	0.465846437	6.47×10^{-01}	–

The values where the algorithms perform the best are highlighted as bold.

Table 7
Comparison of algorithms based on Model Size.

Dataset	hAM-Cont		hAM-Disk		Proposed hAM-C	
	Rules	Terms/rules	Rules	Terms/rules	Rules	Terms/rules
Ds1_aa	24.00 ± 0.26	2.65 ± 0.06	24.60 ± 0.70	2.12 ± 0.15	**17.90 ± 0.10**	**2.11 ± 0.04**
Ds1_interpro	19.30 ± 0.47	5.14 ± 0.20	17.30 ± 1.16	**3.68 ± 0.59**	**10.00 ± 0.52**	4.71 ± 0.39
Ds2_aa	**23.90 ± 0.23**	**2.61 ± 0.05**	62.40 ± 0.97	2.95 ± 0.08	62.20 ± 0.25	3.03 ± 0.03
D2_interpro	48.60 ± 0.37	4.66 ± 0.22	**45.40 ± 1.84**	**3.98 ± 0.47**	51.80 ± 1.32	5.33 ± 0.87
Glass	**25.90 ± 0.48**	3.12 ± 0.07	29.10 ± 0.28	**2.06 ± 0.03**	34.00 ± 0.26	2.21 ± 0.03
Bridge	**14.70 ± 0.21**	1.98 ± 0.05	15.60 ± 0.31	**1.93 ± 0.04**	18.30 ± 0.15	2.09 ± 0.04
AA_tree	23.80 ± 0.25	2.69 ± 0.05	25.00 ± 0.00	2.16 ± 0.03	**18.00 ± 0.00**	**2.12 ± 0.03**
Interpro_tree	20.00 ± 0.39	5.86 ± 0.14	18.30 ± 0.26	**4.27 ± 0.13**	**10.10 ± 0.10**	5.26 ± 0.33

The values where the algorithms perform the best are highlighted as bold.

the two algorithms is zero ($H_0 = \mu 1 - \mu 2 = 0$) which is rejected if the observed *p-value* ≤ *significance level*. The smaller the *p*-value the higher the difference between the means of the algorithms. In case, if the null hypothesis is rejected, a ▲ symbol is placed as an indication that the first algorithm has performed significantly better than the second algorithm (e.g. in the hypothesis 'algo1 vs. algo2', algo1 is the first algorithm). Alternatively, ▼ symbol is placed, if the second algorithm has significantly outperformed the first algorithm. In Table 7, the results are summarized considering the size of classification model in terms of total number of rules, present in the discovered rule list and the total number of terms in a rule. The smaller the values of these two measures, the simpler and compact will be the classification model. The comparison for the model sizes of the two techniques (in terms of average number of rules) over all the datasets is conducted using nonparametric two tailed Wilcoxon signed rank test [35] at the significance level of 0.01. The interpretation of the results presented in Table 8 can be performed in the same way as discussed for Student's *t*-test.

Table 5 illustrates that the proposed algorithm performs the best with the avg. rank of 1.68 followed by hAM-Cont (avg. rank = 2.0). hAM-Disk is the worst performing strategy with the avg. rank of 2.31. hAM-C is also the most accurate algorithm (in terms of hF values) in six out of eight datasets. Whereas on the other hand, hAM-Cont has performed better in Ds2_aa dataset. For discretized data, hAM has achieved better hF values for Ds2_interpro dataset. However, when performed all pair wise statistical significance comparisons (given in Table 6), we have observed that the hAM-C has significantly outperformed the hAM-Cont for Ds1_interpro dataset and declares to be statically significantly better than the hAM-Disk in four out of eight datasets (namely, Ds1_aa, Ds1_interpro, Glass, and AA_tree datasets). On the other hand, hAM-Cont has significantly outperformed the hAM-Disk for Glass dataset. After observing the results presented in Table 6, we cannot find even a single case where the proposed algorithm has significantly defeated by the competitors. It may please be noted that based on the assumption that the classification task become difficult in case of less available examples per a class label, it is encouraging that the proposed algorithm performed better than the compared techniques for Ds1_aa, Ds1_interpro, AA_tree, and Interpro_tree datasets where only 147 instances are available.

Considering the model size comparisons (given in Table 8), we have observed that there is no significant difference between any of these three algorithms and they are equally simplistic. However,

Table 8
Comparison based on two tailed Wilcoxon SRT at 0.01 significance level (Model size).

Hypothesis	Score (±) ranks	*p*-Value	Null hypothesis (H_0)
hAM-Cont vs. hAM-Disk	21/15	0.7422	–
hAM-Cont vs. hAM-C	12/24	0.4609	–
hAM-C vs. hAM-Disk	27/9	0.2500	–

Table 9
Comparison of hAM-C vs. J48_LCN based on hF values.

Dataset	Our hAM-C			J48_LCN		
	hP	hR	hF	hP	hR	hF
Ds1_aa	0.71 ± 0.02	0.63 ± 0.05	0.66 ± 0.03	0.49 ± 0.02	0.77 ± 0.03	0.60 ± 0.02
Ds1_interpro	0.82 ± 0.03	0.78 ± 0.02	0.80 ± 0.02	0.58 ± 0.02	0.91 ± 0.02	0.70 ± 0.02
Ds2_aa	0.56 ± 0.03	0.51 ± 0.02	0.54 ± 0.02	0.68 ± 0.02	0.63 ± 0.02	**0.65 ± 0.02**
D2_interpro	0.78 ± 0.08	0.80 ± 0.06	0.79 ± 0.06	0.88 ± 0.01	0.88 ± 0.02	**0.88 ± 0.01**
Glass	0.84 ± 0.01	0.86 ± 0.01	**0.85 ± 0.01**	0.51 ± 0.01	0.92 ± 0.02	0.66 ± 0.01
Bridge	0.69 ± 0.02	0.70 ± 0.03	**0.69 ± 0.02**	0.28 ± 0.01	0.99 ± 0.01	0.43 ± 0.01
AA_tree	0.65 ± 0.04	0.49 ± 0.03	**0.56 ± 0.03**	0.52 ± 0.03	0.62 ± 0.04	**0.56 ± 0.03**
Interpro_tree	0.75 ± 0.03	0.58 ± 0.02	0.65 ± 0.02	0.63 ± 0.02	0.81 ± 0.04	**0.71 ± 0.02**
Average	0.725	0.669	**0.693**	0.571	0.816	0.649

The values where the algorithms perform the best are highlighted as bold.

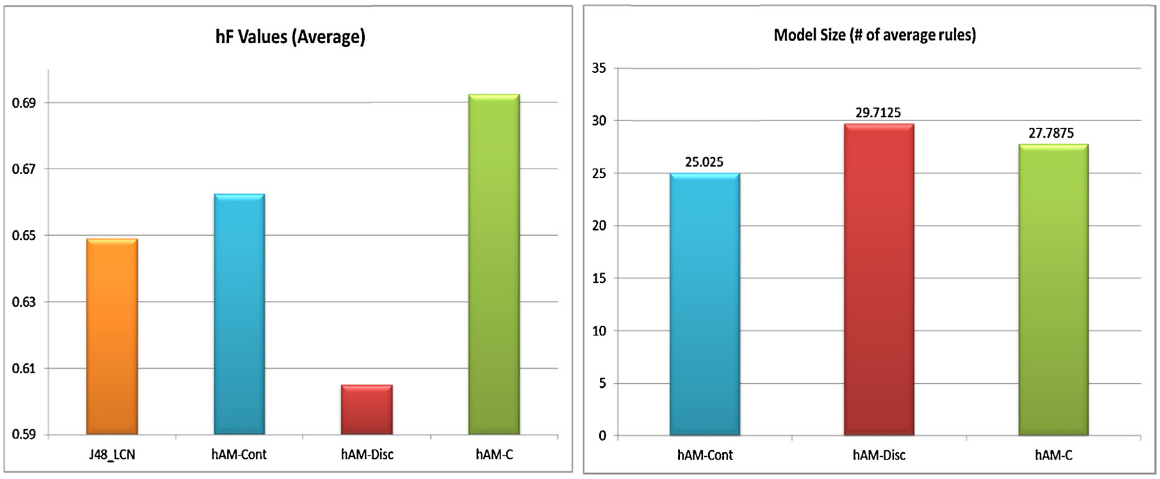


Fig. 8. Overall performance for all datasets based on hF values and model size (# of avg. rules).

for Ds2_aa dataset, hAM-Cont has resulted in the most compact classification model and differences with the competitors is noticeable. This is based on the fact that the twenty-two continuous attributes (present in Ds2_aa) are when discretized has resulted in the loss of information regarding the relationship between predictor attributes and the corresponding class labels. This observation is valid as the model size of hAM-C is equivalent to that of the hAM-Disk where hAM-Disk has the only difference with hAM-Cont that it has generated the results on discretized dataset. It also caused overfitting problem as the classification model built is tool tailored to the training set which resulted in lower generalization power based on small hF values for dataset Ds2_aa (as observed from Table 5).

We also compare the proposed algorithm with the LCN based strategy in terms of hF values by employing a baseline flat classification algorithms 'J48' (weka implementation of C4.5decision tree). As discussed earlier, a single J48 classifier is trained over a specific individual GO term, separately. Specifically, the number of classifiers trained in this case are equal to the number of classes – 1 (ignoring root) for a particular dataset. The instances in training set are used to train the classifiers; such that the positive examples consist of the examples containing the GO term for which the classifier is being trained and all the remaining examples are used to form the set of negative examples. The testing is performed using test examples in a top-down manner. Starting from the classifiers associated with class nodes at first top-level, a test example is assigned for prediction. The classifiers decide whether the given test example is positive or negative i.e. associated or not associated with their GO terms, respectively. For the classifiers, that predict the test example to be positive, test example is further pushed downward (for prediction) to their children classifiers (i.e. classifiers associated with the direct decedents class nodes). This process is iteratively repeated to the classifiers at most specific level of the class hierarchy until a test example is predicted to be negative or a leaf node is reached. All

Table 10
Time taken by each algorithm (in seconds) for tenfold experiment of a dataset.

Dataset	hAM-continuous	hAM-discretized	Proposed hAM-C
Ds1_aa	97	384	67
Ds1_interpro	711	765	324
Ds2_aa	90	993	418
D2_interpro	2321	3370	1100
Glass	48	62	14
Bridge	47	48	17
AA_tree	137	181	60
Interpro_tree	1268	344	116
Average	589.875	768.375	264.500

Table 11
Avg. ants used per rule in discovered rule list, Avg. prob. calls evaluated (resource load).

Dataset Name	Avg. ants used per rule in list	Avg. total probability calls
Ds1_aa	1004.7	3984.511
Ds1_interpro	1369.8	5378.582
Ds2_aa	2157.9	20,215.82
D2_interpro	1290.9	136,489.1
Glass	629.2	2722.939
Bridge	804	2820.356
AA_tree	1091.2	3914.220
Interpro_tree	1920.0	5470.490

the GO terms (for which positive responses are generated) are selected as predicted consequent of the give test example. The selected class labels are consistent with the given class hierarchy, as it is not possible to select a child node when any of its parent is not already present in the selected class labels.

Based on average hF values (as presented in Table 9), hAM-C gives the best average hF value of 0.693 across all the datasets as compared to J48_LCN. In particular, hAM-C is better than the J48_LCN in four out of eight datasets and is defeated in three datasets (namely, Ds2_aa, Ds2_interpro, and Interpro_tree). The comparison of Friedman test with post hoc Holm test based on the hF values over all the datasets for all these four algorithms is conducted again. The average ranks assigned to the algorithms are reported as follows (in order of better performing algorithms first): (hAM-C = 2.125 < J48_LCN = 2.25 < hAM-Cont = 2.5625 < hAM-Disk = 3.0625). In Fig. 8, the overall performance of the algorithms is plotted in terms of average hF values and average model size across all the datasets. As depicted in Fig. 8, hAM-C has obtained the best average hF value bar, however, the model size of hAM-Cont is the most compact and simplistic one. This indicates that adding the dynamic handling of continuous attribute in the classification algorithm can make it simplistic and less prone to over fitting which leads to better generalization power. However, the results of hAM-C are encouraging in terms of prediction performance and therefore its incapability of handling the continuous attributes is not that much significant limitation. The total execution time took by each algorithm for tenfold experiment is presented in Table 10. It may please be noted that the algorithms are implemented in different languages and therefore their reported execution time cannot meaningfully be compared.

In Table 11, we have summarized the number of ants that are used to predict the set of rules and total probability calls evaluated during the execution of the proposed algorithm. The higher values of both of these measures implies that the task of learning the classification model was complex and therefore computationally demanding. It may please be noted that in the worst case, total 10,000 ants (500 iterations $\times$ 20 ants) may be used to predict a single rule as per the parameter setting of hAM-C. However, the results in Table 11 indicates that the original avg. ants used to find a rule in the discovered rule list is much lesser than expected. The total avg. probability calls are usually higher in the datasets where the number of terms are much larger in number, see e.g. in the case of Ds2_interpro dataset where there are 155 number of attributes and in turn the number of terms are also larger in number.

5.3. Experimental investigation of algorithmic design choices

In order to explore the significance of the differences between hAM-C and hAM, we have analyzed several modifications of hAM-C individually and in interplay, of which the average results over tenfold are given in Fig. 9 across all the eight discretized datasets. Although, minute modifications in AntMiner's might seem straightforward, one has to consider that many of the features are entangled with one another [22,36,37]. It is the combined working of all the features of our approach, such as the search space, correlation based heuristic and pheromone function, etc., that yield the reported results. The key modifications are described next.

(1) *Symmetric/asymmetric pheromone matrix*: Our pheromone matrix is asymmetric which encourages exploration of the search space and discourages the premature convergence. For hAM-C, we have conducted experiments in which pheromone values on both the edges between two chosen consecutive terms are updated. This pheromone update method yields hAM-C-Sym (a symmetric pheromone matrix based hAM-C).

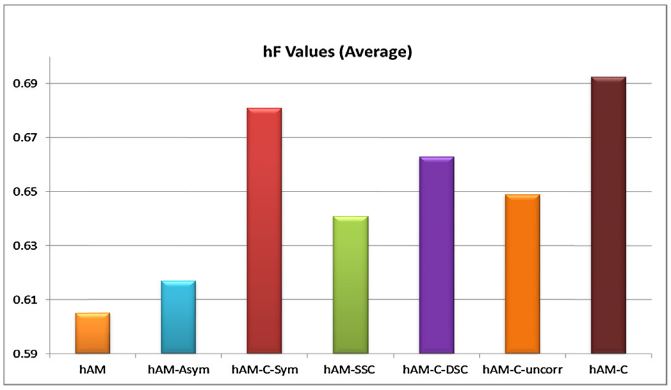


Fig. 9. Overall performance for all datasets based on hF values.

Experimental results, shown in Fig. 9, indicate a decrease in performance when symmetric pheromone update is used. The ants have a lesser chance for exploration and converge quickly to a solution and it may be an inferior one. For brevity, we only discuss that the premature convergence is analyzed by two statistics rather than presenting all the results: (1) the average of ants used to construct the rule set has significantly decreases, and (2) the count on early termination of the main loop of hAM-C algorithm is quite high due to duplicate rule generation. Similarly, hAM is equipped with asymmetric pheromone matrix (hAM-Asym) which caused it to perform slightly better as compared to its original form, due to the same reasons.

(2) *Learning same/different set of class labels*: For hAM-C, we make sure that all the ants learn the same set of class labels during an iteration of the main loop. On the other hand, for hAM, each ant might be learning a different set of class labels. For hAM, ants are searching non-cooperatively, each with different goals; so their experience if shared by means of pheromone, should not be very helpful for search process; as this is against the main essence of ACO. In this dimension, hAM-C is modified such that each ant can learn different set of class labels (hAM-C-DSC i.e. hAM-C with different set of class labels). Similarly, for hAM, all the ants are enforced to learn common goal in order to mimic cooperative learning behavior (hAM-SSC i.e. hAM with same set of class labels). However, in order to incorporate this change in hAM, three more modifications are required: (1) construction of antecedent part should depends on prior commitment of consequent part, (2) ACG and CCG must be connected in the same way as done for hAM-C, and (3) a heuristic function that guides in the view of committed consequent. These changes are incorporated in hAM-SSC and following heuristic function (modified version of Eq. (9)) is used for this purpose:

$$\eta_{ij} = \sum_{k=1}^{L} \frac{\left|term_j, class_k\right|}{\left|term_j\right|} \tag{16}$$

Although not significant, an overall average improvement is reported for hAM-SSC. hAM-C-DSC has resulted in slight degradation in performance. With these experiments, we have observed that updating the environment based on cooperative learning with the common goal is more helpful.

(3) *Correlation based heuristic function*: In hAM, the construction of rule antecedent part is independent from the construction of rule consequent part (Ants in ACG do not communicate with the ants in CCG). This is a serious limitation of hAM; as the ants have no idea of class labels for which they are constructing the rule (or the other way around: the ants have no idea of antecedent part for which the class labels are to be selected). With this limitation, it is less likely to discover IF–THEN rules of good

quality. For hAM-C, the two sub-graphs of hAM are connected so that the construction of rule antecedent and consequent parts become relevant. This is also mandatory for using the proposed correlation based heuristic function which requires beforehand consequent selection in order to construct the antecedent part of a rule. Selection of terms based on the history of ant trail is the main component of hAM-C that makes it perform much better than hAM. In order to validate this observation, hAM-C heuristic function is modified to the one given in Eq. (16) (hAM-C-uncorr. i.e. hAM-C with uncorrelated heuristic function). According to Fig. 9, significant degradation is observed in the average performance of hAM-C-uncorr. We cannot include correlation based heuristic function in hAM, as this require several modification in hAM (e.g. connected sub-graphs, prior consequent commitment, common goal to learn, etc.), after which hAM will become almost identical to hAM-C.

6. Conclusion

In this article, we have presented a novel ant colony optimization based single path hierarchical classification algorithm, named hAM-C. A detailed review of different types of hierarchical classification problems and different categories of corresponding solutions is also provided to enhance the understanding regarding the target problem and to facilitate the readers. Extending on the ideas of our previous flat classification algorithm AntMiner-C, the hAM-C is tailor to handle the hierarchical classification task with both tree (a class node can have single parent apart from root) and DAG (a class node can have multiple parents apart from root) class hierarchical structures. hAM-C employs a sequential covering approach and discover a single global classification model in the form of IF–THEN rules that are used in sequence during the testing phase. The usage of a new correlation based heuristic which takes into consideration the relationships between attribute-value pairs and class hierarchy, makes the hAM-C very well suited to the underlying problem.

The proposed algorithm is evaluated on six ion-channels datasets (complex for classification task) related to protein function prediction and two publically available datasets. The empirical comparisons are performed with a previously proposed single path hierarchical classification algorithm (based on ACO) hAM and a baseline decision tree (J48) approach extended to handle the hierarchical classification. According to the experimental results, hAM-C is significantly superior to the competitors (based on several statistical tests) in terms of prediction accuracy (based on hierarchical F-Measure) and gives comparable model complexities. We have also presented the statistics about the resources used by hAM-C in terms of ants used to find a single rule and total number of probability calls (more the calls the higher the computation complexity) evaluated during the execution of the algorithm. With these statistics, we have concluded that hAM-C makes efficient and effective usage of the assigned resources.

There are several future avenues in order to extend the proposed technique. First, it would be interesting to equip the algorithm with a dynamic continuous variable handling mechanism, that would give more insights in the data being mined. It might also be useful for reducing the size of classification model built by algorithm and avoid the over fitting which will make the algorithm more robust. A difference based heuristic measure in the class hierarchy space (as used in CLUS-HMC) can be introduced which will extend the algorithm for handling hierarchical multi-label classification problems. It is also encouraging to testify the variations of new rule pruning strategies or new rule evaluation measures. A new sequential covering approach introduced in cAntMiner-PB [36] is worth enough to be investigated for optimizing the quality of the discovered rules.

References

[1] Y.-L. Chen, H.-W. Hu, K. Tang, Constructing a decision tree from data with hierarchical class labels, Expert Systems with Applications 36 (3) (2009) 4838–4847.
[2] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[3] J.R. Quinlan, Generating production rules from decision trees, in: Proceedings of the International Joint Conference on Artificial Intelligence, San Francisco, USA, 1987.
[4] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.
[5] A. Freitas, A. de Carvalho, A tutorial on hierarchical classification with applications in bioinformatics, in: D. Taniar (Ed.), Research and Trends in Data Mining Technologies and Applications, IGI Global, 2007, pp. 175–208.
[6] M. Ashburner, et al., The gene ontology: tool for the unification of biology, Nature Genetics 25 (1) (2000) 25–29.
[7] F.E.B. Otero, A.A. Freitas, 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 (EvoBio), LNCS 5483, Springer, Tübingen, Germany, 2008, pp. 68–79.
[8] F.E.B. Otero, A.A. Freitas, C.G. Johnson, A hierarchical multi-label classification ant colony algorithm for protein function prediction, Memetic Computing 2 (3) (2010) 165–181.
[9] F.E.B. Otero, New Ant colony optimization algorithms for hierarchical classification of protein functions, 2010, PhD thesis.
[10] C.N. Cilla, B. Becker, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Mining & Knowledge Discovery 22 (1–2) (2010) 31–72.
[11] D. Koller, M. Sahami, Hierarchically classifying documents using very few words, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 170–178.
[12] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Machine Learning 73 (2) (2008) 185–214.
[13] H. Blockeel, L. De Raedt, J. Ramon, Top-down induction of clustering trees, in: Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 55–63.
[14] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transaction on Evolutionary Computation 6 (August (4)) (2002) 321–332.
[15] F. Otero, A. Freitas, C. Johnson, cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes, in: Ant Colony Optimization and Swarm Intelligence (ANTS 2008), LNCS 5217, Springer, Brussels, Belgium, 2008.
[16] A.P. Engelbrecht, Computational Intelligence, An Introduction, 2nd ed., John Wiley & Sons, Hudson County, New Jersey, United States, 2007.
[17] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley & Sons, Hudson County, New Jersey, United States, 2005.
[18] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann/Academic Press, Middlesex County, Massachusetts, United States, 2001.
[19] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[20] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 26 (February (1)) (1996) 29–41.
[21] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the travelling salesman problem, IEEE Transaction on Evolutionary Computation 1 (April (1)) (1997) 53–66.
[22] D. Martens, M. de Backer, R. Haesen, J. Vanthienen, M. Snoeck, B. Baesens, Classification with ant colony optimization, IEEE Transactions on Evolutionary Computation 11 (October (5)) (2007) 651–665.
[23] A. Abraham, C. Grosan, V. Ramos, Swarm Intelligence in Data Mining. Studies in Computational Intelligence, vol. 34, Springer, Heidelberg, New York, 2006.
[24] J. Han, M. Kamber, Data Mining: Concepts and Techniques, 2nd ed., Morgan Kaufmann Publishers, Middlesex County, Massachusetts, United States, 2006.
[25] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, Middlesex County, Massachusetts, United States, 2005.
[26] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing. Natural Computing Series, 2nd ed., Springer, 2007.
[27] S. Khan, M. Bilal, M. Sharif, M. Sajid, R. Baig, Solution of n-Queen problem using ACO, in: International Multitopic Conference, IEEE, Islamabad, 2009, pp. 1–5.
[28] A.R. Baig, W. Shahzad, A correlation based ant miner for classification rule discovery, Neural Computing and Applications Journal (2010) (in press and available online from Springer website for NCA journal).
[29] S. Kiritchenko, S. Matwin, A.F. Famili, Functional annotation of genes using hierarchical text categorization, in: BioLINK SIG: Linking Literature, Information and Knowledge for Biology, 2005.
[30] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, The Molecular Biology of the Cell, 4th ed., Garland Press, New York, NY, 2002.
[31] S. Hettich, S.D. Bay, The UCI KDD Archive, Dept. Inf. Comp. Sci., Uni. California, Irvine, CA, 1996 [Online] Available: http://kdd.ics.uci.edu
[32] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.

[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Machine Learning Research 7 (2006) 1–30.
[34] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1923.
[35] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.
[36] F.E.B. Otero, A.A. Freitas, C.G. Johnson, A new sequential covering strategy for inducing classification rules with ant colony algorithms, IEEE Transaction on Evolutionary Computation 17 (February (1)) (2012) 64–76.
[37] A.R. Baig, W. Shahzad, Salabat Khan, Correlation as a heuristic for an accurate and comprehensible ACO based classifier, IEEE Transactions on Evolutionary Computation 17 (December (5)) (2012) 686–704.