Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 6320273, 15 pages
https://doi.org/10.1155/2017/6320273

Research Article

Orderly Roulette Selection Based Ant Colony Algorithm for Hierarchical Multilabel Protein Function Prediction

Zhengping Liang, Rui Guo, Jiangtao Sun, Zhong Ming, and Zexuan Zhu

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Correspondence should be addressed to Zexuan Zhu; zhuzx@szu.edu.cn

Received 29 December 2016; Revised 19 May 2017; Accepted 25 May 2017; Published 28 June 2017

Academic Editor: Francisco Chicano

Ant colony optimization (ACO) algorithms have been successfully applied to identify classification rules in data mining. This paper proposes a new ant colony optimization algorithm, named *hm*AntMiner$_{\text{order}}$, for the hierarchical multilabel classification problem in protein function prediction. The proposed algorithm is characterized by an orderly roulette selection strategy that distinguishes the merits of the data attributes through attributes importance ranking in classification model construction. A new pheromone update strategy is introduced to prevent the algorithm from getting trapped in local optima and thus leading to more efficient identification of classification rules. The comparison studies to other closely related algorithms on 16 publicly available datasets reveal the efficiency of the proposed algorithm.

1. Introduction

In the last few decades, various techniques have been successfully proposed to solve classification problems in the fields of machine learning and data mining [1–3]. However, most of the existing classification techniques are designed to handle data with binary or nominal class labels (where class labels are independent). They cannot handle problems with multiple class labels organized in hierarchical structure (CHS) [4]. Such problems are commonly known as hierarchical classification with regard to the one-level flat classification problems.

Due to the complex structure of hierarchical multilabel classification problems, they are more difficult to solve than the flat single label classification problems. The samples may be assigned to several classes that form a hierarchical structure, for example, a tree or directed acyclic graph [5], at the same time. Some difficulties are inherent for those problems. Firstly, because of the hierarchical structure of classes, less nodes are available at the bottom of the tree than that at the top. As such, it is more difficult to classify the nodes if the tree is deep. Secondly, samples classified in the lower levels of the hierarchy must satisfy the parent-child relationships; that is, they should also fall within the parent classes. Finally, a sample can also be classified to multiple classes that have no parent-child relationship.

Recently, many bioinspired heuristic algorithms have been designed to solve optimization problems and successfully applied in data classification problems [6–8]. Among them, ant colony optimization (ACO) algorithms have shown promising performance in mining classification rules in the form of "*IF* $\langle term_1 \rangle$ *AND* $\langle term_2 \rangle$ *AND* $\cdots \langle term_n \rangle$ *THEN* $\langle class \rangle$." The rules identified by ACO algorithms not only perform well in terms of the predictive accuracy, but also can be easily expressed in natural language and thus lead to good comprehensibility [9]. Nevertheless, the exponential increasing of data volume and types in the fields of machine learning and data mining has posed great challenges for ACO algorithms to deal with hierarchical multilabel classification problems especially in terms of computation efficiency and robustness.

In this paper, a novel ACO-based algorithm named *hm*AntMiner$_{\text{order}}$ is presented to identify classification rules for the hierarchical multilabel classification problem in protein function prediction. *hm*AntMiner$_{\text{order}}$ is equipped with an orderly roulette selection strategy and a new pheromone update strategy to enhance the capability of handling large-scale problems and the robustness. Particularly, in the orderly

Input: training samples
Output: adiscovered list of rules
(1) *examples* ← *all training examples*;
(2) *Rule_set* ← ϕ;
(3) **while** |*training samples*| > maximum uncovered **do**
(4) *Initialize Pheromones*(), *Heuristic Information*(*examples*), $rule_{global\text{-}best}$;
(5) $t \leftarrow 0$;
(6) **while** t < maximum iterations **and** no stagnation **do**
(7) **for** $n \leftarrow 1$ **to** ants_size **do**
(8) $rule_n \leftarrow$ *Create Rule*(*examples*);
(9) *Prune*($rule_n$);
(10) *Evaluate* $rule_n$;
(11) $rule_{iteration\text{-}best} \leftarrow rule_n$;
(12) **end for**
(13) *Update Pheromones*($rule_{iteration\text{-}best}$);
(14) *Evaluate* $rule_{iteration\text{-}best}$;
(15) $rule_{global\text{-}best} \leftarrow rule_{iteration\text{-}best}$;
(16) $t \leftarrow t + 1$;
(17) **end while**
(18) *Traing set* ← *Taining set* − *Covered*($rule_{gb}$, *examples*);
(19) *Rule_set* ← *Rule_set* + $rule_{global\text{-}best}$;
(20) **end while**
(21) **return** *Rule_set*;

Algorithm 1: The pseudocode of AntMiner algorithm.

roulette selection, the data attributes are sorted such that the algorithm can distinguish the pros and cons of each attribute and construct good classification model more efficiently. The new pheromone update strategy is designed to guide the ants to find better global optimal solutions, which strengthens the degree of pheromone update to avoid falling into local optimum. To evaluate the performance of the proposed algorithm, eighteen publicly available datasets are employed. Two closely related decision-tree-based algorithms (CLUS-HSC and CLUS-SC) [10] and two ACO-based algorithms (*hm*AntMiner [11] and *hm*AntMiner-C [12]) are involved in the comparison study. $hm\text{AntMiner}_{\text{order}}$ shows superiority in terms of prediction accuracy and comprehensibility of classification model.

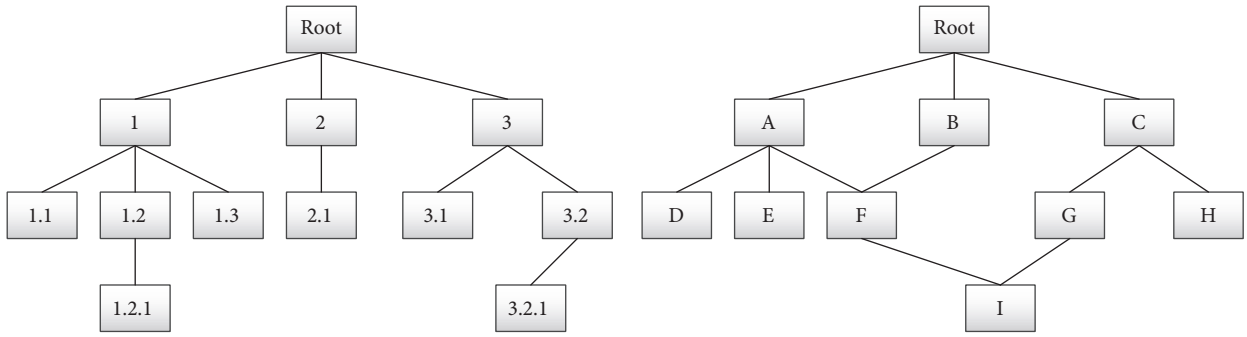
The remainder of this paper is organized as follows. Section 2 reviews ACO algorithms for classification rules discovery and the existing algorithms in hierarchical multilabel classification for protein function prediction. The proposed algorithm is also described in Section 2 where the details of the orderly roulette selection strategy and the new pheromone update strategy are provided. Section 3 presents the experimental results of the comparison studies on publicly available datasets. Finally, Section 4 concludes this study and some future directions are discussed.

2. Materials and Methods

2.1. ACO for Classification

2.1.1. ACO for Flat Single Label Classification Problems. ACO has been widely used in the flat single label classification problems. The AntMiner algorithm proposed by Parpinelli et al. [13] represents one of the most well-known ACO-based classifiers. In AntMiner, a heuristic search method is introduced to identify the rule information in the dataset and a sequential covering strategy to discover a rule. Based on the discovered rules, the classified training samples are removed and the training set is reduced. The remaining samples are used for further rule discovery and the process iterates until no enough training samples are available. Particularly, the process of AntMiner is outlined in Algorithm 1.

First, the rule set is initialized to be empty, and each ant starts to build a rule by adding one term at a time. The pheromone and heuristic values of the new term decide whether it should be added to the rule set. To calculate the heuristic values of one term, the entropy and normalized information gain [14] are used. The rule is applied to the dataset by a majority vote mechanism and the irrelevant rule terms are pruned to raise the accuracy [15]. Then the next rule is constructed by other artificial ants. After all the ants have built their rules, the best rule in that iteration is identified and the pheromone is updated based on that basis [16]. If the best rule is better than the global best rule, the global best rule will be set to be equal to the local best rule. Otherwise, the iteration best rule will be discarded. The global best rule up till now is used to remove the examples correctly covered and the next global best rule is found using the remaining training examples. This process of constructing a global best rule is repeated until the maximum number of iterations is reached, or the current constructed best rule is the same as the best one constructed by a specified number of previous iterations. The outer global rule set growing iteration stops if the number of



(a) Tree hierarchy (b) DAG hierarchy

Figure 1: Representation of class labels in hierarchical classification.

remaining examples is less than a threshold. The output of this algorithm is an ordered set of rules, which is used to classify the test dataset.

Many variants of AntMiner have been proposed to improve the performance of classification. For example, AntMiner2 [17] and AntMiner3 [18] use a simple heuristic function, which adopts a density estimation in rules discovery and is calculated only once for each term, to replace the relatively complex heuristic in AntMiner. Moreover, to encourage exploration, AntMiner3 presents a new pheromone update method, in which the pheromone is updated and evaporated only for those predefined conditions occurring in the rules. In an enhanced version of AntMiner (AntMiner+), the class label is selected before the ants build their rules and a class-specific heuristic function is imposed to enable the ants to know the class of an extracting rule [19]. The relative importance of the pheromone and heuristic values is adjusted by two important ACO parameters α and β. A new AntMiner variant, namely, AntMiner-CC, considers the relationship between the term selected previously and the next candidate term by utilizing a new heuristic function [20] based on the correlation of dataset attributes, given the preselected class label and its potential to maximize the correct coverage.

$\text{AntMiner}_{\text{mbc}}$ [21] proposed by Liang et al. adopts a new heuristic information function considering both the correlation and coverage for the purpose to avoid deceptive high accuracy. cAnt-Miner [22] and its improved version [23] introduce continuous attributes handling strategy and new rule sequential covering strategy, respectively, to enhance the performance of rule identification. Smaldon and Freitas [24] improved AntMiner to produce an unordered set of classification rules. ACORI [25] uses an optimization method to find the near optimal order of rules in the decision list. μcAnt-Miner [26] embeds several extensions into the original AntMiner algorithms. Multiple pheromone level types are considered when the rule's consequent class is selected prior to the antecedent of the rule construction.

AntMiner can also be improved by mixing with other heuristic optimization algorithms or classifiers. For example, by combining the strengths of AntMiner and particle swarm optimization [27], a resultant hybrid algorithm provides a very promising performance thanks to their specific capabilities in handling continuous attribute and nominal attribute-value construction. Ant-Tree-Miner [28] induces decision tress rather than a set of rules, which is consequently quite different from AntMiner and its variants. The advantage of the decision trees is that the model it represents is easy to understand in a graphical form and the ACO algorithm outputs a set of classification rules. Boryczka and Kozak proposed an ACDT algorithm [29] which can make agents-ants interact during the construction decision trees via pheromone values to generate solutions efficiently. In a real world application, Feng et al. combined SVM method with the clustering based on self-organized ant colony network to take the advantages of both while avoiding their weaknesses and then used this algorithm to classify network activities as normal or abnormal [30].

2.1.2. ACO for Hierarchical Multilabel Classification. In hierarchical classification, the class labels are naturally organized as a class hierarchy/taxonomy, which typically are represented as a tree or directed acyclic graph (DAG), as shown in Figures 1(a) and 1(b). In the hierarchy, the nodes represent the class labels and edges represent the relationship between the class labels. Different class hierarchy structures impose different restrictions on the graph; for example, in DAG a node can have more than one parent. To predict a class label in the hierarchy, the classifier should also predict all the ancestor class labels.

It is clear that the edges between the parent and the children node represent IS-A relationship in the hierarchy. The nodes at the top levels of that hierarchy are easier to predict because they represent more general class labels, whereas the nodes at the bottom levels are more difficult to predict, because more information is needed to distinguish them. For these reasons, the classifier should look for a tradeoff between generality and specificity in the hierarchical classification. An example is given in Figure 2 for the class classification of human. If we predict an item as "Human," it is 100% correct. However, predicting the lower level specific

Attribute 1	Attribute 2	Attribute 3	Class labels		
V_{11}	V_{21}	V_{31}	Human	Asian	Mongol
V_{12}	V_{22}	V_{32}	Human	Asian	Malaysian
V_{13}	V_{23}	V_{33}	Human	African	Ethiopian
V_{14}	V_{24}	V_{34}	Human	African	Bantu
V_{15}	V_{25}	V_{35}	Human	Caucasoid	Greek
V_{16}	V_{26}	V_{36}	Human	Caucasoid	English

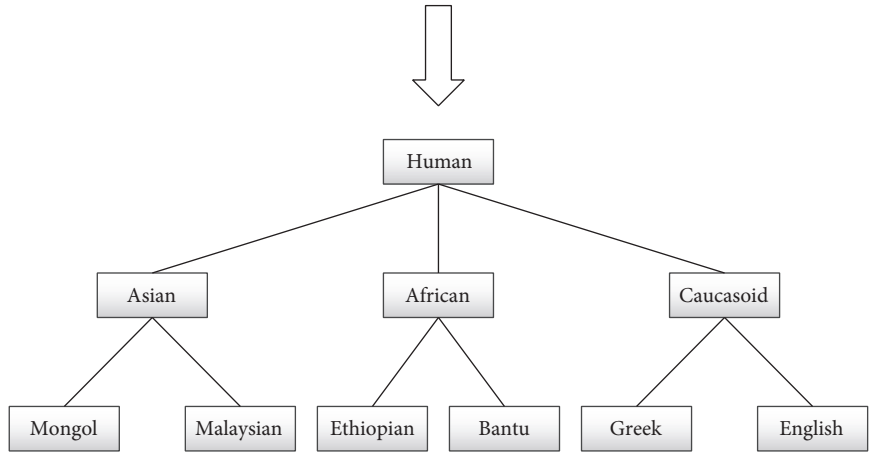


Figure 2: Hierarchy structure of human classes.

class is more important in this setting and it is more liable to make the wrong prediction.

Hierarchical multilabel classification problems can be handled by constructing a baseline classifier for each class label, also known as local approach, or considering all the hierarchically related classes on a whole, that is, global approach. For example, Koller and Sahami [31] proposed a local classifier approach algorithm that works by training a decision tree for each class label individually. For a given item, a baseline decision tree is used to predict the presence/absence of the corresponding class labels. Chen et al. [32] extended the decision tree classifier to predict the hierarchical class labels, where the best attributes are selected using an extended entropy measure. Vens et al. [10] investigate two local approaches based on decision tree, namely, CLUS-HSC and CLUS-SC, and a global approach, CLUS-HMC, to classify the labels in the hierarchy simultaneously. Particularly, CLUS-HMC is based on the theory of predictive clustering tree framework [33], and each node in the tree is conceived as a cluster. Generally, local approaches tend to be more computationally demanding as a classifier must be trained many times. Moreover, the misclassifications at higher levels are propagated and affect the classification of lower level labels [34]. Global approaches can overcome the aforementioned problems by considering all the hierarchically related classes at once. However, global approaches are more difficult to model than local approaches.

ACO-based approaches have also been increasingly used to deal with hierarchical multilabel classification problem in a global manner. Chan and Freitas [35] proposed a new ACO algorithm, named MuLAM (Multilabel AntMiner), to discover a multilabel classification rule which can predict one or more class labels at a time. Otero et al. [5] proposed *h*AntMiner (Hierarchical Classification AntMiner) for hierarchical classification problem, which is an extension of the flat classification AntMiner algorithm. In *h*AntMiner, a hierarchical rule evaluation measure, heuristic information, and an extended rule representation are used for classification. *h*AntMiner is further extended to *hm*AntMiner [11] to handle hierarchical multilabel classification problem of protein function prediction. A new heuristic function based on the Euclidean distance is introduced in *hm*AntMiner to discover an ordered list of hierarchical multilabel classification rules. The experimental results presented in [11] demonstrate the superiority of *hm*AntMiner to other local/global methods including CLUS-HSC, CLUS-SC, and CLUS-HMC. Khan and Baig [12] proposed an *hm*AntMiner variant, namely, *hm*AntMiner-C by introducing search space simplification mechanisms, more accurate correlation based heuristic function, and new representation of pheromone matrix and evaporation process. In this work, we also improve *hm*AntMiner by introducing an orderly roulette selection strategy and a new pheromone update strategy. The resultant algorithm, namely, *hm*AntMiner$_{\text{order}}$ is described in the following section.

*2.2. The Proposed hm*AntMiner$_{\text{order}}$ *Algorithm.* Following the general structure of ACO algorithm, some modifications

```
Input: protein training examples
Output: classification model decision_list
(1)  decision_list ← ϕ;
(2)  examples ← protein training examples;
(3)  while |examples| > max_uncovered_examples and not converged do
(4)     Calculate New Heuristic Information(examples);
(5)     Initialize Pheromone();
(6)     rule_gb ← ϕ;
(7)       i ← 1;
(8)     while not (i ≥ max_number_iterations OR Rule_Convergence) do
(9)        rule_ib ← ϕ;
(10)       for n ← 1 to colony_size do
(11)            // use the new roulette selection strategy
(12)            rule_n ← Orderly_roulette_selection_strategy_CreateRule(examples);
(13)            Prune(rule_n);
(14)            if Quality(rule_n) > Quality(rule_ib) then
(15)               rule_ib ← rule_n;
(16)            end if
(17)       end for
(18)          // use the new pheromone update strategy
(19)          Intensive_Update_ Pheromone(rule_gb);
(20)          if Quality(rule_ib) > Quality(rule_gb) then
(21)             rule_gb ← rule_ib;
(22)          end if
(23)    end while
(24)     decision_list ← decision_list + rule_gb;
(25)     examples ← examples − Covered(rule_gb, examples);
(26)  end while
(27) return decision_list;
```

Algorithm 2: Pseudocode of *hm*AntMiner$_{\text{order}}$ algorithm.

are made in *hm*AntMiner$_{\text{order}}$ to construct each list of rules. Firstly, we design a new roulette selection strategy to distinguish the merits of the data attributes through attributes importance ranking. So each ant can find a better rule. Secondly, we use a new pheromone update strategy to strengthen the degree of pheromone update and give the ant a better guide. That update strategy uses the global best rule instead of the local optimal rule. Finally, the new algorithm utilizes a large number of uncharacterized proteins in the analysis and does better to determine their functions in the biological process. The result of *hm*AntMiner$_{\text{order}}$ algorithm is an ordered list of hierarchical multilabel classification rules to predict protein functions. The pseudocode of *hm*AntMiner$_{\text{order}}$ algorithm is described in Algorithm 2. The rule building process continues until all ants have built their own rules. Then the Klösgen measure [36] is used to evaluate the constructed rules so the precision can be corrected for the class distribution. *hm*AntMiner$_{\text{order}}$ reduces the computational cost by only considering the relevant term of the iteration best rule. The global best rule is updated after multiple assessments to help the ants find better rules in the next iteration.

The details of *hm*AntMiner$_{\text{order}}$ are provided as follows. As shown in Algorithm 2, in line (1), it starts with an empty decision list. Then in lines (3)–(26), in the outer *while* loop, the algorithm iteratively adds one rule at a time to the decision list until the termination criteria are satisfied. In lines (8)–(23), an inner *while* loop is executed, and in each iteration a rule is constructed by an ACO procedure. All ants choose data attributes to be added to their current partial rule by the orderly roulette selection strategy. In line (13), the duplicate data attributes are pruned in the rule. The quality of the rules in the current iteration is evaluated and a best rule in this iteration is selected as the iteration best rule, as described in lines (14)–(16). The pheromone trails are updated in line (19) using the global best rule based on the intensive pheromone update strategy to guide ants to search for better rules. In lines (20)–(22), if the iteration best rule is better than the global best rule, the iteration best rule will be selected as the global best rule. Then the global best rule constructed covered is added to the decision list of rules and the covered training examples (training examples that satisfy the antecedent of the global best rule) are removed from the training set in lines (24)-(25). The procedure of creating a rule is repeated until the accuracy on validation begins to reduce and the rest of protein training samples are examples less than the predefined max uncovered examples. This prevents the classification model from the noise in the training data and the separate validation set can be monitored during the training phase.

IF
aa_rat_pair_a_h ≥ 0.058
AND aa_rat_pair_t_c ≥ 0.1055
AND aa_rat_pair_c_w < 0.0695
AND aa_rat_pair_a_e < 0.2960
AND aa_rat_pair_t_h ≥ 0.0275
THEN
GO0000226: 0.10, GO0000943: 0.50,
GO0001302: 0.10, GO0003647: 1.00,
GO0003676: 0.50, GO0003723: 0.50,
...
GO0045185: 0.10, GO0046907: 0.20,
GO0051234: 0.20, GO0051235: 0.10,
GO0051649: 0.20, GO0051651: 0.10

Figure 3: Example of the consequent of a rule discovered by *hm*AntMiner$_{\text{order}}$.

2.2.1. Hierarchical Multilabel Rule Consequent. The outputs of many previous algorithms are usually a single path in the consequent construction graph, that is, a trail from the root class label down to the leaf class label in the class hierarchy. But for protein prediction it will not work in that form. To apply those outputs to protein prediction problems, the examples covered by the rule (examples that satisfy the rule antecedent) can be used as the information to determine the rule consequent. The consequent of a rule in *hm*AntMiner$_{\text{order}}$ algorithm is computed by a deterministic procedure as follows:

$$\text{class}_{r,i} = \frac{\left|\text{Set}_r \ \& \ \text{label}_i\right|}{\left|\text{Set}_r\right|}, \tag{1}$$

where Set_r represents the set of examples covered by rule r, which generates a vector of length m (m is the number of class labels) as a result of that rule. label_i is the ith component of the class vector. $|\text{Set}_r \ \& \ \text{label}_i|$ is the number of examples belonging to the ith class of the class hierarchy that is covered by rule r. The class i, a vector of length i, represents the proportion of examples which are covered by rule r in a particular ith class.

Based on the previous definition, each element of a vector is a continuous value ranging from 0.0 to 1.0, rather than single value 1 or 0, that is, true or false value of a particular class label. The value is a probability of the examples covered by a rule satisfying the antecedent to belong to the corresponding ith class of the hierarchy. Figure 3 shows an example of a result of a rule discovered by *hm*AntMiner$_{\text{order}}$ algorithm. The predictor attributes in the IF statement are amino acid ratios from the protein sequence and the THEN part are Gene Ontology terms representing the class labels. Following each GO term is the probability of the sequence belonging to that GO term class label.

2.2.2. Hierarchical Multilabel Rule Construction

(1) Heuristic Information Function. In *hm*AntMiner$_{\text{order}}$, the heuristic function incorporates a distance-based information using the class hierarchy. The variance of the one set of examples covered by the term is incorporated in the heuristic information. A numeric vector of length m represents the class label of each example. If the ith component of the class label vector is 0 it means the example does not belong to that class, and the same logic applies to the 1's case. We use a weighted Euclidean distance to represent the distance between the class label vectors as follows:

$$\text{Ed}\left(v_1, v_2\right) = \sqrt{\sum_{i=1}^{m} w\left(l_i\right) \cdot \left(v_{1,i} - v_{2,i}\right)^2}, \tag{2}$$

where $w(l_i)$ is the weight of the ith class label and $v_{1,i}$ and $v_{2,i}$ represent the ith class value of two examples, respectively. We use the average square distance between each of example class labels and the set's mean class vector to represent the variance of a set of examples as follows:

$$\text{var}\left(\text{Set}_T\right) = \frac{\sum_{k=1}^{|\text{Set}_T|} \text{Ed}\left(v_k, \overline{v}\right)^2}{\left|\text{Set}_T\right|}, \tag{3}$$

where Set_T is the set of examples covered by a term T and $\overline{v}$ is the set's mean class label vector. Finally, the heuristic information of a term T is given by

$$\eta_T = \frac{\text{var}_{\max} - \text{var}\left(\text{Set}_T\right)}{\text{var}_{\max}}, \tag{4}$$

where $\text{var}_{\max}$ is the sum of the best and the worst variance values of all terms. The definition ensures assigning values greater than zero to the worst terms, which otherwise would prevent them from being selected by an ant.

Moreover, *hm*AntMiner$_{\text{order}}$ also uses a class-specific weighting scheme, where the weight is defined as follows:

$$w\left(l\right) = w_0 \cdot \frac{\sum_{i=1}^{|p_l|} w\left(p_i\right)}{\left|p_l\right|}, \tag{5}$$

where w_0 is set to 0.75, p_l is the parent class label set of the class label l, and $w(p_i)$ is the weight associated with the ith

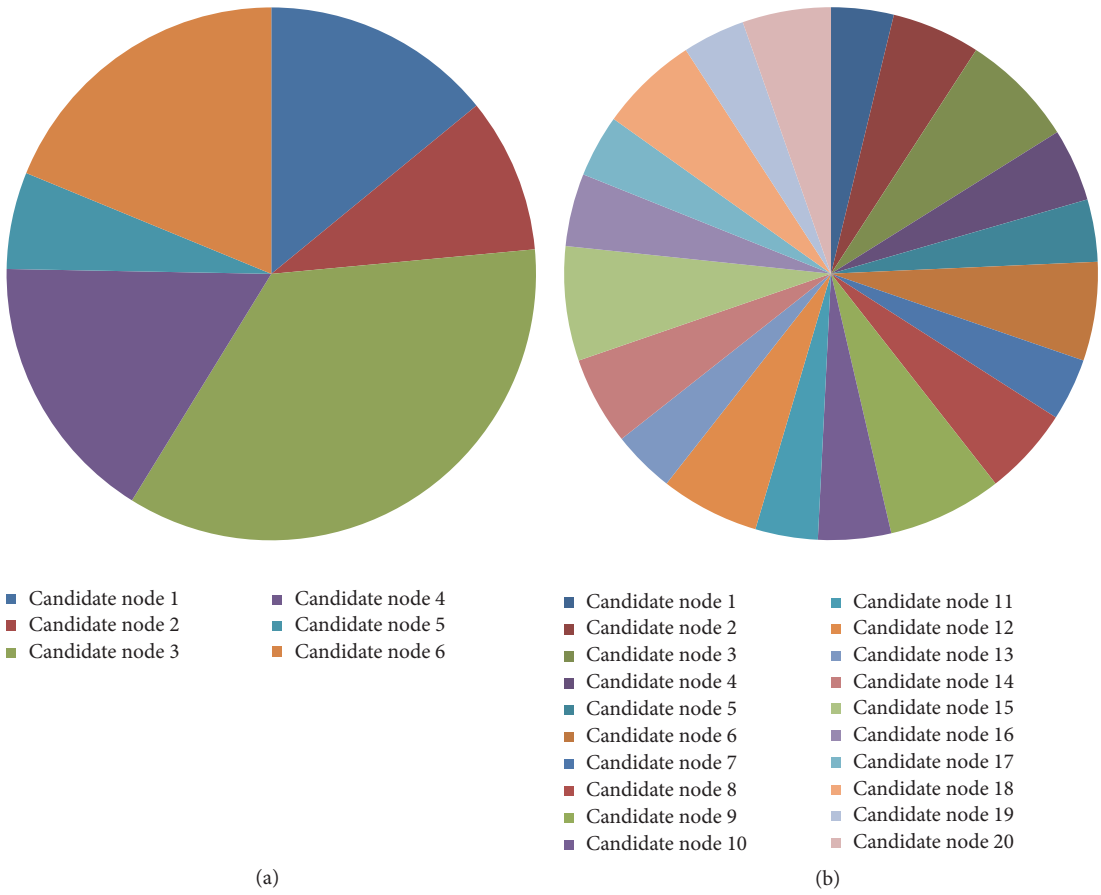


Figure 4: Roulette selection strategy.

parent class label of the class label l. According to (5), classes in the higher part of the class hierarchy have bigger weights than the class label in the lower part in the hierarchy.

(2) A New Roulette Selection Strategy. In the search process of ACO, artificial ants constantly choose nodes through the guidance of pheromone and heuristic information and eventually search for a best solution. Each node corresponding to the heuristic information value is calculated based on (4). The pheromone values associated with an edge between two nodes accumulate constantly in the iteration process of ant colony optimization. The probability of selection of node is given by the following formula:

$$p_{ij}(t) = \frac{\tau_{ij}(t)\,\eta_j(t)}{\sum_{k=1}^{\text{total nest values}} \tau_{ik}(t)\,\eta_k(t)}, \tag{6}$$

where $\tau_{ij}(t)$ is the concentration of pheromone between node$_i$ and node$_j$ for the tth ant, $\eta_j(t)$ is the value of the heuristic information in node$_j$, $\tau_{ik}(t)$ is the amount of pheromone concentration between node$_i$ and node$_j$, where k is a value increasing from 1 to the total number of next attribute values, and $\eta_k(t)$ is its current value of the heuristic function. All the selected nodes belong to those attributes that have not become prohibited.

Based on (6), each ant uses the roulette selection strategy to select effective nodes. Roulette selection is also known as selection operator; that is, the probability of an individual being selected is proportional to its fitness function value, as shown in the following formula:

$$P_i = \frac{F_i}{\sum_{i=1}^{n} F_i}, \tag{7}$$

where n is the number of candidate nodes; the fitness value of each candidate node is F_i. The larger the value of P_i, the greater the probability of ith node being selected.

For example, one of the candidate nodes, in Figure 4(a), represents the area of one piece of the pie chart. The area of the block is proportional to the fitness value of the candidate nodes. As the number of protein data attributes increases like Figure 4(b), the roulette selection strategy tends to be trapped in random selection. This paper proposes an orderly roulette selection strategy; all nodes are in an orderly sequence according to the probability of each node. In this way, artificial ants can differentiate the merits of the node. Orderly roulette selection strategy removes the poor

candidate nodes; that is, only the better candidate nodes are selected, such that all artificial ants can select excellent candidate nodes more efficiently and generate better rules.

(3) Hierarchical Multilabel Rule Evaluation. Using a distance-based measure, the variance gain can be applied to compute a rule quality measure. The basic idea to evaluate a rule r using the variance gain measure is to virtually divide the training set S into two partitions: the set of examples covered by the rule r (S_r) and the set of examples not covered by the rule r ($S_{\neg r}$). Then the variance gain of rule r relative to S is computed as follows:

$$\text{var\_gain}(r, S) = \text{var}(S) - \frac{|S_r|}{|S|}\text{var}(S_r) - \frac{|S_{\neg r}|}{|S|}\text{var}(S_{\neg r}). \tag{8}$$

The variance can naturally cope with hierarchical multilabel data, taking into account the relationships and similarities between class labels. And it favors rules that partition the training set into more homogeneous sets of examples. At last, rules that cover a more homogeneous set of examples, as well as leaving a more homogeneous set of examples uncovered, are preferred.

(4) A New Pheromone Update Strategy. The pheromone values are associated with an edge between two vertices in the graph. Because the number of protein data attributes is large, artificial ants are difficult to converge to optimal solution. In the pheromone matrix, the decrease of the pheromone concentration is accomplished by pheromone evaporation. Over time, the amount of pheromone on all the edges reduces by an evaporation factor ρ, while the global best rule based on its quality reinforces its pheromone concentration. The quality of a rule is shown as follows:

$$Q = \left(\frac{\text{TP} + \text{FP}}{P + N}\right) \cdot \left(\frac{\text{TP}}{\text{TP} + \text{FP}} - \frac{P}{P + N}\right), \tag{9}$$

where TP and FP, respectively, refer to the numbers of correct and incorrect examples covered by the rule that have the same class label. P is the total number of examples whose class labels are the selected class. N is the total number of examples belonging to other classes. Equation (9) is used to evaluate all the rules.

The pheromone update formula is given as follows:

$$\tau_i(t+1) = \tau_i(t) + Q_{\text{best}}, \tag{10}$$

where Q_{best} is quality of the global best rule, $\tau_i(t)$ is the concentration of pheromone released by the ith ant in the tth iteration. In our new pheromone update function, the update amplitude of pheromone concentration of the global best rule increases more than in the original version. The pheromone on the better rules accumulates faster and more, at the same time, which strengthens the convergence of the algorithm.

3. Experimental Results

In this section, the experimental setting is first introduced and then the performance of the proposed *hm*AntMiner$_{\text{order}}$ algorithm is evaluated using 16 publicly available datasets [10], which include two different class hierarchy structures: the tree structure, that is, the FunCat dataset, and the DAG structure, the Gene Ontology (GO) dataset. The DAG structure represents a more complex hierarchical organization, where a particular node of the hierarchy can have more than one parent. In contrast, in tree structures, each node has only one parent. The average numbers of class labels of FunCat and GO datasets are 489 and 3932, respectively. The average numbers of labels per example in FunCat and GO datasets are 8.5 and 34.2, respectively. The detailed information of the two datasets is provided in Table 1.

In the experiments, 2/3 of each dataset is used for training and the remaining 1/3 is used for testing. The proposed algorithm *hm*AntMiner$_{\text{order}}$ is compared with two closely related decision-tree-based algorithms (CLUS-HSC and CLUS-SC) [10] and two ACO-based algorithms (*hm*AntMiner [11] and *hm*AntMiner-C [12]). CLUS-SC is a local approach that induces a decision tree for each class label individually to deal with hierarchical multilabel classification problems. CLUS-HSC is also a local approach to construct decision trees in a top-down fashion to predict the functions of protein data. *hm*AntMiner is a global approach than can discover an ordered list of hierarchical multilabel classification rules based on ant colony optimization. *hm*AntMiner-C is an improved version of *hm*AntMiner. We use the same training and test partitions for all algorithms in the experiments to guaranty a fair comparison.

3.1. Performance Metric and Parameters Setting. To evaluate the proposed algorithms, the main consideration is the classification accuracy, which is the percentage of correctly classified test samples. The comprehensibility of the classifiers [37, 38] is accessed by the number of discovered rules and the number of terms per rule, which are used as indirect performance metrics.

Generally, more iterations and ants can help get a better result. However, simply increasing those two parameters may cause a great raise in execution time but a small gain in accuracy. To overcome this problem, we use F-Race [38] racing procedure to identify optimal parameter settings. For the two parameters mentioned above, three different values for each are tested. The number of ants is selected from $\{10, 100, 1000\}$ while the number of iterations is chosen in $\{10, 100, 1000\}$. These nine combinations of parameters are commonly used in ACO-based algorithms [39]. Our experiments show that when the maximal number of iterations is set to 10, the algorithm obtains best tradeoff between convergence and time consumption. Besides, the number of ants is set to 10, which ensures that more ants are employed to find a better solution. It is validated in lots of experiments that the remaining parameters set to 10, respectively, can obtain higher accuracy while maintaining a reasonable execution time [11]. All the parameter settings for our proposed algorithm are shown in Table 2, while the

Table 1: Summary of the protein datasets used in the experiments.

Dataset	Training set	Testing set	Attributes	Classes
FunCat				
cellcycle	2476	1281	77	500
derisi	2450	1275	63	500
eisen	1587	837	79	462
expr	2488	1291	551	500
gasch1	2480	1284	173	500
pheno	1009	582	69	456
seq	2580	1339	478	500
spo	2437	1266	80	500
GO				
cellcycle	2473	1278	77	4126
derisi	2447	1272	63	4120
eisen	1583	835	79	3574
expr	2485	1288	551	4132
gasch1	2477	1281	173	4126
pheno	1005	581	69	3128
seq	2568	1332	478	4134
spo	2434	1263	80	4120

Table 2: Parameter settings for *hm*AntMiner$_{order}$ algorithm.

Parameter	Value
Max uncovered examples	10
Max number of iterations	10
Rule convergence	10
Min examples per rule	10
Ant colony size	10

parameter settings of CLUS-SC and CLUS-HSC are set as recommended in their papers [10].

3.2. Precision-Recall Curves to Evaluate Classification Model. In information retrieval [40] and hierarchical multilabel classification [10], PR (precision-recall) curves are frequently used for its suitability to deal with highly skewed datasets (much more negative examples than positive ones). PR curve plots a precision value against recall value. The precision value is the number of correct predictions divided by the total number of predictions. The recall value is the number of correct predictions divided by the total number of positive examples, that is, examples belonging to the predicted class label. Those two values only take the positive values into account, so the number of negative predictions does not influence the evaluation. As mentioned above, the lower level classes are more difficult to have a true positive result. PR curves ignore the true negative examples so this explains how well a rule predicts the presence of a particular class label.

*3.3. Comparisons of hm*AntMiner$_{order}$ *with Various Classification Algorithms.* In this subsection, the performance of our algorithm (*hm*AntMiner$_{order}$) is compared with two classical classification algorithms (CLUS-HSC and CLUS-SC) and two ACO-based classification algorithms (*hm*AntMiner and *hm*AntMiner-C). Our algorithm is implemented in Java. The software myra-3.7 [41] is adopted, while a Java Library for Multilabel Learning [42] is used to run CLUS-HSC and CLUS-SC. The results in Table 3 show the average accuracy achieved by the cross-validation procedure followed by the standard error of all algorithms in the corresponding datasets. The experimental results concerning the size of the construction classification model are summarized in Table 4, where the smallest model size on each dataset is marked with boldface. The results of CLUS-HSC and CLUS-SC are measured with the average numbers of leaf nodes in the generated decision tree. The size of classification model of the remaining algorithms is obtained by recording the average number of rules.

Vargha-Delaney A-test [43] is used to measure the statistical significance of the experimental result. It is a nonparametric effect magnitude test to differentiate between two samples of observations. Its return value is a probability value between 0 and 1, indicating the probability that a randomly selected observation from $\mathbf{X}$ is bigger or smaller than a randomly selected sample from $\mathbf{Y}$, which also represents the degree to which the two samples are overlapped. A p value in interval [0, 0.29] or [0.71, 1.0] indicates a significant difference between $\mathbf{X}$ and $\mathbf{Y}$. In other cases, no significant difference is observed. In Tables 5 and 6, the symbols "+" and "−" are used to denote that *hm*AntMiner$_{order}$ is significantly better or worse than the corresponding compared algorithm,

Table 3: The AU($\overline{\text{PRC}}$) value obtained on the test set by each algorithm across all datasets used in our experiments.

Dataset	*hm*AntMiner$_{\text{order}}$	*hm*AntMiner-C	*hm*AntMiner	CLUS-HSC	CLUS-SC
FunCat					
cellcycle					
AU(PRC)	**0.160 ± 0.001**	0.154 ± 0.001	0.154 ± 0.001	0.111	0.106
Rank	**1**	2	2	4	5
derisi					
AU(PRC)	**0.169 ± 0.001**	0.167 ± 0.001	0.161 ± 0.002	0.094	0.089
Rank	**1**	2	3	4	5
eisen					
AU(PRC)	**0.188 ± 0.002**	0.175 ± 0.002	0.180 ± 0.003	0.127	0.132
Rank	**1**	3	2	5	4
expr					
AU(PRC)	**0.182 ± 0.003**	0.167 ± 0.002	0.175 ± 0.002	0.127	0.123
Rank	**1**	3	2	4	5
gasch1					
AU(PRC)	**0.187 ± 0.002**	0.173 ± 0.002	0.175 ± 0.003	0.106	0.104
Rank	**1**	3	2	4	5
pheno					
AU(PRC)	**0.163 ± 0.001**	0.163 ± 0.001	0.162 ± 0.001	0.152	0.149
Rank	**1**	2	3	4	5
seq					
AU(PRC)	0.175 ± 0.002	0.166 ± 0.002	**0.181 ± 0.002**	0.091	0.095
Rank	2	3	**1**	5	4
spo					
AU(PRC)	**0.179 ± 0.002**	0.167 ± 0.001	0.174 ± 0.002	0.103	0.098
Rank	**1**	3	2	4	5
GO					
cellcycle					
AU(PRC)	0.352 ± 0.001	**0.430 ± 0.001**	0.332 ± 0.002	0.371	0.252
Rank	3	**1**	4	2	5
derisi					
AU(PRC)	0.348 ± 0.002	**0.437 ± 0.001**	0.334 ± 0.003	0.349	0.218
Rank	3	**1**	4	2	5
eisen					
AU(PRC)	0.383 ± 0.001	**0.450 ± 0.002**	0.376 ± 0.002	0.365	0.270
Rank	2	**1**	3	4	5
expr					
AU(PRC)	0.383 ± 0.002	**0.440 ± 0.001**	0.351 ± 0.003	0.351	0.249
Rank	2	**1**	3	3	5
gasch1					
AU(PRC)	0.367 ± 0.002	**0.442 ± 0.001**	0.356 ± 0.002	0.351	0.239
Rank	2	**1**	3	4	5
pheno					
AU(PRC)	0.340 ± 0.001	**0.427 ± 0.001**	0.337 ± 0.001	0.416	0.316
Rank	3	**1**	4	2	5
seq					
AU(PRC)	0.368 ± 0.002	**0.450 ± 0.002**	0.366 ± 0.003	0.282	0.197
Rank	2	**1**	3	4	5
spo					
AU(PRC)	0.341 ± 0.002	**0.441 ± 0.001**	0.341 ± 0.003	0.371	0.213
Rank	3	**1**	3	2	5
A. rank	**1.81**	**1.81**	2.75	3.56	4.88

Table 4: The classification model size obtained on the test set by each algorithm across all datasets used in our experiments.

Dataset	*hm*AntMiner$_{order}$	*hm*AntMiner-C	*hm*AntMiner	CLUS-HSC	CLUS-SC
FunCat					
cellcycle					
Size	**14.000 ± 1.528**	30.867 ± 1.606	28.667 ± 1.623	4037	9671
Rank	**1**	3	2	4	5
derisi					
Size	15.300 ± 0.907	**07.000 ± 0.561**	19.333 ± 1.661	3520	7807
Rank	2	**1**	3	4	5
eisen					
Size	**11.500 ± 0.522**	24.267 ± 1.926	19.000 ± 0.981	2995	6311
Rank	**1**	3	2	4	5
expr					
Size	**19.000 ± 0.760**	27.933 ± 1.987	30.600 ± 1.466	4711	10262
Rank	**1**	2	3	4	5
gasch1					
Size	**16.800 ± 0.952**	21.400 ± 2.086	24.867 ± 1.701	4761	10447
Rank	**1**	2	3	4	5
pheno					
Size	**3.200 ± 0.200**	06.400 ± 0.335	7.400 ± 0.767	777	1238
Rank	**1**	2	3	4	5
seq					
Size	**15.600 ± 0.670**	17.467 ± 1.473	20.067 ± 1.152	4923	10443
Rank	**1**	2	3	4	5
spo					
Size	13.800 ± 0.629	**7.7333 ± 0.740**	15.800 ± 1.172	3623	8527
Rank	2	**1**	3	4	5
GO					
cellcycle					
Size	**19.300 ± 1.023**	26.800 ± 1.642	35.400 ± 1.594	19085	36260
Rank	**1**	2	3	4	5
derisi					
Size	14.500 ± 1.249	**09.533 ± 1.064**	22.533 ± 1.939	16693	31175
Rank	2	**1**	3	4	5
eisen					
Size	**16.500 ± 0.969**	25.333 ± 2.072	18.200 ± 0.823	14384	24844
Rank	**1**	3	2	4	5
expr					
Size	**16.100 ± 0.809**	21.000 ± 1.670	28.600 ± 1.778	20812	38313
Rank	**1**	2	3	4	5
gasch1					
Size	**18.500 ± 0.982**	21.733 ± 1.442	27.933 ± 0.918	20070	37838
Rank	**1**	2	3	4	5
pheno					
Size	**3.100 ± 0.640**	05.467 ± 0.307	7.133 ± 0.792	5691	6213
Rank	**1**	2	3	4	5
seq					
Size	17.500 ± 0.895	**15.467 ± 1.490**	18.067 ± 1.016	21703	38969
Rank	2	**1**	3	4	5
spo					
Size	24.100 ± 1.729	**07.333 ± 0.760**	26.333 ± 2.520	15552	35400
Rank	2	**1**	3	4	5
A. rank	**1.31**	1.88	2.81	4	5

Table 5: Summary of the comparisons of the *hm*AntMiner$_{\text{order}}$ algorithm (control) with the remaining algorithms according to the Vargha-Delaney *A*-test in terms of predictive accuracy.

Dataset	*hm*AntMiner-C	*hm*AntMiner	CLUS-HSC	CLUS-SC
FunCat				
cellcycle	+	+	+	+
derisi	+	+	+	+
eisen	+	+	+	+
expr	+	+	+	+
gasch1	+	+	+	+
pheno	≈	≈	+	+
seq	+	≈	+	+
spo	+	≈	+	+
GO				
cellcycle	−	+	≈	+
derisi	−	+	≈	+
eisen	−	+	+	+
expr	−	+	+	+
gasch1	−	+	+	+
pheno	−	+	−	+
seq	−	≈	+	+
spo	−	≈	−	+
Better/Similar/Worse	7/1/8	11/5/0	12/2/2	16/0/0

Table 6: Summary of the comparisons of the *hm*AntMiner$_{\text{order}}$ algorithm (control) with the remaining algorithms according to the Vargha-Delaney *A*-test in terms of classification model size.

Dataset	*hm*AntMiner-C	*hm*AntMiner	CLUS-HSC	CLUS-SC
FunCat				
cellcycle	+	+	+	+
derisi	−	+	+	+
eisen	+	+	+	+
expr	+	+	+	+
gasch1	+	+	+	+
pheno	+	+	+	+
seq	+	+	+	+
spo	−	≈	+	+
GO				
cellcycle	+	+	+	+
derisi	−	+	+	+
eisen	+	≈	+	+
expr	+	+	+	+
gasch1	+	+	+	+
pheno	+	+	+	+
seq	−	≈	+	+
spo	−	≈	+	+
Better/Similar/Worse	11/0/5	12/4/0	16/0/0	16/0/0

respectively. Symbol "≈" suggests the results of the two compared algorithms are similar.

From the results shown in Table 3, *hm*AntMiner$_{order}$ obtains the best overall predictive accuracy in FunCat dataset and the second best accuracy in GO dataset. *hm*AntMiner-C wins in GO dataset, whereas it is not comparable to *hm*AntMiner$_{order}$ in FunCat dataset. To represent the search space topology, *hm*AntMiner-C uses the layering of attribute-value pair, as a grid and DAG topology, which leads to much simpler search space than its competitors. It is not surprising that *hm*AntMiner-C outperforms other algorithms in GA datasets that are organized in DAG structure. On each dataset, the "Rank" value indicates the performance ranking of the corresponding algorithm among all algorithms. The last row "A. rank" denotes the average rank on all datasets. The A. rank of *hm*AntMiner$_{order}$ is 1.81, which is equal to that of *hm*AntMiner-C. *hm*AntMiner, CLUS-HSC, and CLUS-SC achieve A. rank values of 2.75, 3.56, and 4.88, respectively. It is observed from the last row in Table 3 that *hm*AntMiner$_{order}$ and *hm*AntMiner-C perform the best in terms of predictive accuracy.

Besides predictive accuracy, we also compare the average classification model sizes of different algorithms. For rules discovery classification algorithms, the number of rules reflects the size of the rule list, because each rule is correlated with a class label. In decision tree algorithms, the leaf nodes are noted by class labels, which reflect the classification model size. In Table 4, the size of *hm*AntMiner$_{order}$ is the smallest in all the datasets. Also in the last row of Table 4, the *hm*AntMiner$_{order}$ achieves the lowest average ranks, which means a better average performance than other algorithms.

Statistical test of the performance difference between *hm*AntMiner-C, *hm*AntMiner, CLUS-HSC, CLUS-SC, and *hm*AntMiner$_{order}$ is shown in Tables 5 and 6. The results present the summary of the comparisons of the *hm*AntMiner$_{order}$ algorithm (our algorithm with the best average rank) with the remaining algorithms used in our experiments according to the Vargha-Delaney *A*-test in terms of predictive accuracy and classification model size. For each algorithm, the test results obtained by Vargha-Delaney *A*-test are reported on both FunCat and Gene Ontology datasets. The last row "Better/Similar/Worse" indicates the number of datasets on which the proposed algorithm is significantly better than, similar to, and significantly worse than the other algorithms, respectively. In Table 5, *hm*AntMiner$_{order}$ shows comparable performance to *hm*AntMiner-C and statistically better performance than the other compared algorithms in terms of prediction accuracy. Regarding the classification model size, as shown in Table 6, *hm*AntMiner$_{order}$ is observed to obtain significantly smaller model size than the other algorithms in most of the test datasets. Overall, the *hm*AntMiner$_{order}$ obtains the best compromise of prediction accuracy and classification model size considering both tree and DAG hierarchical structures.

4. Conclusion and Future Work

In this paper, we propose *hm*AntMiner$_{order}$, a novel ACO-based classification algorithm with a high predictive accuracy and low model size. Some new features are introduced to the proposed algorithm. Firstly, a new roulette selection strategy is designed to distinguish the merits of the data attributes through attributes importance ranking. In this way, each ant can search for a better rule efficiently. Secondly, a new pheromone update strategy is presented to strengthen the degree of pheromone update and complete a better guide to the ants. *hm*AntMiner$_{order}$ can cope with the large increase in the number of uncharacterized proteins available for analysis and the importance of determining their functions in order to improve the current biological knowledge. These new features are implemented in our algorithm and 16 publicly available datasets are used to evaluate the classification performance of *hm*AntMiner$_{order}$. When compared with the other four closely related classification algorithms, including *hm*AntMiner-C, *hm*AntMiner, CLUS-HSC, and CLUS-SC, *hm*AntMiner$_{order}$ performs superiorly or competitively in terms of predictive accuracy and obtains preferable comprehensibility. In the future work, other components like local search [44–47] and differential operators [48] can be introduced to *hm*AntMiner$_{order}$ to improve the efficiency of the algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants nos. 61471246 and 61672358), Innovation Foundation for Higher Education of Guangdong, China (Grant no. 2016KTSCX121), Guangdong Foundation of Outstanding Young Teachers in Higher Education Institutions (Grant no. Yq2013141), Guangdong Special Support Program of Top-Notch Young Professionals (Grant no. 2014TQ01X273), and Shenzhen Scientific Research and Development Funding Program (Grant no. ZYC201105170243A).

References

[1] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, Elsevier, 2011.

[2] A. Keramati, R. Jafari-Marandi, M. Aliannejadi, I. Ahmadian, M. Mozaffari, and U. Abbasi, "Improved churn prediction in telecommunication industry using data mining techniques," *Applied Soft Computing Journal*, vol. 24, pp. 994–1012, 2014.

[3] F. Zandi, "A bi-level interactive decision support framework to identify data mining-oriented electronic health record architectures," *Applied Soft Computing Journal*, vol. 18, pp. 136–145, 2014.

[4] A. A. Freitas and A. C. de Carvalho, "A tutorial on hierarchical classification with applications in bioinformatics," in *Research and Trends in Data Mining Technologies and Applications, Idea Group*, D. Taniar, Ed., 2007.

[5] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "A Hierarchical classification ant colony algorithm for predicting gene ontology terms," *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5483, pp. 68–79, 2009.

[6] A. P. Engelbrecht, *Computational intelligence: an introduction*, John Wiley & Sons, 2007.

[7] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence*, John Wiley & Sons, 2006.

[8] B. Javidy, A. Hatamlou, and S. Mirjalili, "Ions motion algorithm for solving optimization problems," *Applied Soft Computing Journal*, vol. 32, pp. 72–79, 2015.

[9] B. Minnaert and D. Martens, "A comment on 'correlation as a heuristic for accurate and comprehensible ant colony optimization-based classifiers'," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 790-791, 2014.

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

[11] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "A hierarchical multi-label classification ant colony algorithm for protein function prediction," *Memetic Computing*, vol. 2, no. 3, pp. 165–181, 2010.

[12] S. Khan and A. R. Baig, "Ant colony optimization based hierarchical multi-label classification algorithm," *Applied Soft Computing*, vol. 55, pp. 462–479, 2017.

[13] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321–332, 2002.

[14] M. Pedemonte, S. Nesmachnow, and H. Cancela, "A survey on parallel ant colony optimization," *Applied Soft Computing Journal*, vol. 11, no. 8, pp. 5181–5197, 2011.

[15] P. N. Tan, *Introduction to data mining*, Pearson Education India, 2006.

[16] S. Hodnefjell and I. Costa Jr., "Classification rule discovery with ant colony optimization algorithm," in *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 678–687, 2012.

[17] B. Liu, H. A. Abbass, and B. McKay, "Density-based heuristic for rule discovery with ant-miner," in *The 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System*, pp. 180–184, 2002.

[18] B. Liu, H. A. Abbas, and B. McKay, "Classification rule discovery with ant colony optimization," in *IAT*, vol. 3, pp. 83–88, 2003.

[19] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651–665, 2007.

[20] A. R. Baig, W. Shahzad, and S. Khan, "Correlation as a heuristic for accurate and comprehensible ant colony optimization based classifiers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 686–704, 2013.

[21] Z. Liang, J. Sun, Q. Lin, Z. Du, J. Chen, and Z. Ming, "A novel multiple rule sets data classification algorithm based on ant colony algorithm," *Applied Soft Computing Journal*, vol. 38, pp. 1000–1011, 2016.

[22] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Ant-Miner: an ant colony classification algorithm to cope with continuous attributes," in *International Conference on Ant Colony Optimization and Swarm Intelligence*, vol. 5217, pp. 48–59, 2008.

[23] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "A new sequential covering strategy for inducing classification rules with ant colony algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 64–76, 2013.

[24] J. Smaldon and A. A. Freitas, "A new version of the ant-miner algorithm discovering unordered rule sets," in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference 2006*, pp. 43–50, July 2006.

[25] S. Asadi and J. Shahrabi, "ACORI: A novel ACO algorithm for rule induction," *Knowledge-Based Systems*, vol. 97, pp. 175–187, 2016.

[26] K. M. Salama, A. M. Abdelbar, F. E. B. Otero, and A. A. Freitas, "Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery," *Applied Soft Computing Journal*, vol. 13, no. 1, pp. 667–675, 2013.

[27] N. P. Holden and A. A. Freitas, "A hybrid PSO/ACO algorithm for classification," in *Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference, GECCO 2007*, pp. 2745–2750, July 2007.

[28] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Inducing decision trees with an ant colony optimization algorithm," *Applied Soft Computing Journal*, vol. 12, no. 11, pp. 3615–3626, 2012.

[29] U. Boryczka and J. Kozak, "Enhancing the effectiveness of Ant Colony Decision Tree algorithms by co-learning," *Applied Soft Computing Journal*, vol. 30, pp. 166–178, 2015.

[30] W. Feng, Q. Zhang, G. Hu, and J. X. Huang, "Mining network data for intrusion detection through combining SVMs with ant colony networks," *Future Generation Computer Systems*, vol. 37, pp. 127–140, 2014.

[31] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 170–178, San Francisco, CA, USA, 1997.

[32] Y.-L. Chen, H.-W. Hu, and K. Tang, "Constructing a decision tree from data with hierarchical class labels," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4838–4847, 2009.

[33] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proceedings of the 15th ternational Conference on Machine Learning*, J. Shavlik, Ed., pp. 53–63, 1998.

[34] C. N. Silla Jr. and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[35] A. Chan and A. A. Freitas, "A new ant colony algorithm for multi-label classification with applications in bioinfomatics," in *Proceedings of the 8th Annual Conference on Genetic And Evolutionary Computation*, pp. 27–34, Seattle, Wash, USA, July 2006.

[36] B. Minnaert, D. Martens, M. De Backer, and B. Baesens, "To tune or not to tune: rule evaluation for metaheuristic-based sequential covering algorithms," *Data Mining and Knowledge Discovery*, vol. 29, no. 1, pp. 237–272, 2015.

[37] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models," *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, 2011.

[38] N. R. Daud and D. W. Corne, "Human readable rule induction in medical data mining," *Lecture Notes in Electrical Engineering*, vol. 27, no. 1, pp. 787–798, 2009.

[39] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Handling continuous attributes in ant colony classification algorithms," in *Computational Intelligence and Data Mining (CIDM'09)*, pp. 225–231, IEEE Symposium, 2009.

[40] C. D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Mass, USA, 1st edition, 1999.

[41] F. E. Otero, A Collection of Ant Colony Optimization (ACO) Algorithms for the Data Mining Classification Task, 2013, Available at: https://github.com/febo/myra.

[42] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "An empirical study of multi-label learning methods for video annotation," in *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing, CBMI 2009*, pp. 19–24, grc, June 2009.

[43] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[44] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 70–76, 2007.

[45] Z. Zhu, J. Xiao, J.-Q. Li, F. Wang, and Q. Zhang, "Global path planning of wheeled robots using multi-objective memetic algorithms," *Integrated Computer-Aided Engineering*, vol. 22, no. 4, pp. 387–404, 2015.

[46] Z. Zhu, J. Xiao, S. He, Z. Ji, and Y. Sun, "A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem," *Information Sciences*, vol. 329, pp. 73–89, 2016.

[47] Z. Zhu, S. Jia, S. He, Y. Sun, Z. Ji, and L. Shen, "Three-dimensional Gabor feature extraction for hyperspectral imagery classification using a memetic framework," *Information Sciences*, vol. 298, pp. 274–287, 2015.

[48] Z. Liang, K. Hu, Q. Zhu, and Z. Zhu, "An enhanced artificial bee colony algorithm with adaptive differential operators," *Applied Soft Computing*, vol. 58, pp. 480–494, 2017.