Correlation as a Heuristic for Accurate and Comprehensible Ant Colony Optimization Based Classifiers

Abdul Rauf Baig, *Member, IEEE,* Waseem Shahzad, and Salabat Khan

***Abstract*—The primary objective of this research is to propose and investigate a novel ant colony optimization-based classification rule discovery algorithm and its variants. The main feature of this algorithm is a new heuristic function based on the correlation between attributes of a dataset. Several aspects and parameters of the proposed algorithm are investigated by experimentation on a number of benchmark datasets. We study the performance of our proposed approach and compare it with several state-of-the art commonly used classification algorithms. Experimental results indicate that the proposed approach builds more accurate models than the compared algorithms. The high accuracy supplemented by the comprehensibility of the discovered rule sets is the main advantage of this method.**

***Index Terms*—Ant colony optimization, classification algorithms, data mining.**

I. Introduction

SWARM intelligence algorithms [1]–[3] are based on the study and modeling of the collective behavior of small and simple entities, and are considered a genre of population-based algorithms [4]. Ant colony optimization (ACO) [5]–[9] is a metaheuristic covered under the umbrella of swarm intelligence. It is based on the food foraging behavior of ants. In recent years, ACO has increasingly been used as an effective search method for solving complex and difficult combinatorial optimization problems, including those related to data mining [5], [9], [10].

Classification is one of the main data mining tasks [11]–[13]. Many classification algorithms already exist, such as decision trees, neural networks, k-nearest neighbor classifiers, and support vector machines. Some of them (e.g., neural networks and support vector machines) act as black box systems and their manner of reasoning is opaque to users, while others are easily comprehensible (e.g., decision tress). In many applications, both comprehensibility and accuracy are required.

Manuscript received January 29, 2012; revised July 31, 2012; accepted October 15, 2012. Date of publication December 5, 2012; date of current version September 27, 2013.

A. R. Baig is with Imam Muhammad bin Saud Islamic University, Riyadh, Saudi Arabia, on leave from the National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan (e-mail: rauf.baig@nu.edu.pk).

W. Shahzad and S. Khan are with the National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan (e-mail: waseem.shahzad@nu.edu.pk; salabat.khan@nu.edu.pk).

Digital Object Identifier 10.1109/TEVC.2012.2231868

Classification rules have the advantage of comprehensibility. The classification rule discovery problem is: given a set of training data comprising one or more attributes and a class attribute, we discover rules governing the separation of that data into different classes. ACO-based classification rule discovery algorithms have shown promising results [14]–[18]. In this paper, our focus is on exploring a new heuristic function and utilizing it in the framework of ACO. We propose three variants of a novel algorithm for the discovery of classification rules. Our approach has the desirable properties of high accuracy and classifier comprehensibility.

The remainder of this paper is organized as follows. In Section II, we explain the basic concepts of ACO and give an overview of existing ACO algorithms for classification rules discovery. Section III describes our proposed approach. In Section IV, we discuss the experimental results on some publicly available data sets and analyze its various aspects. Section V concludes this paper and gives some future directions of research.

II. ACO and Classification Rule Discovery

In this section, we give a summary of ACO and review the previous efforts for its application to the classification rule discovery process.

A. ACO

ACO is a family of metaheuristics, initially proposed by Dorigo [5], based on the food foraging behavior of biological ants. It is considered as a branch of swarm intelligence (and, in general, a genre of population-based heuristic algorithms).

Ants pass on the information about the trail they are following to other ants by spreading a chemical substance, called pheromone, in the environment. The concentration of pheromone on a given path acts as a guide for later ants. Ants that arrive in the vicinity are more likely to take the path with higher concentration of pheromone than the paths with lower concentrations. The pheromone evaporates with time and becomes insignificant if not enforced by the passage of more ants. This indirect form of communication between ants helps them to establish a short path between their nest and a food source. Ants taking the shorter of the available paths return sooner and thus concentration of pheromone on

1089-778X © 2012 IEEE

that path is reinforced more quickly than on competing longer paths.

This food foraging behavior has been modeled in the ACO metaheuristic. Given a problem whose solution can be represented as a combination of components, an artificial ant is sent to construct a solution. Using the terminology of graphs, we assume vertices to be the possible components of the solution and edges to be pairs of these components. An edge can also be visualized as a hypothetical connection or path between a pair of components. Each edge has an associated pheromone. An ant constructs a solution by selecting components one by one. This selection is done probabilistically based on the pheromone concentrations on edges of competing components. A constructed solution is then evaluated by a measure of fitness. The components of the solution are assigned pheromone concentrations proportional to the quality of the solution. Subsequent ants thus have a higher probability of selecting those components again which have contributed toward a better solution. The components with higher pheromone concentrations are thus identified as contributing to a good solution and repeatedly appear in the solutions. A heuristic value of each component is also added in the probabilistic selection, if such a heuristic is available. Usually, after sufficient iterations, the ants converge on a good, if not the optimal, solution.

For the application of ACO to a problem, we should be able to represent its solution as a combination of different individual components. Furthermore, there should be a method to determine the fitness or quality of the solution. A heuristic measure for the solution's components is also desirable although not necessary. Since the early 90s, different ACO algorithms have been proposed which differ in the manner of probability calculation, pheromone update, etc. Three of the more well-known and successful algorithms are ant system (AS) [6], MAX–MIN ant system (MMAS) [7], and ant colony system (ACS) [8]. They mainly differ in two ways: construction of new solutions and update of pheromone levels.

B. Solution Construction

In all of these three algorithms, an ant constructs a solution by probabilistically selecting its components one by one. When it has selected a component i and has so far constructed the partial solution, the probability of selecting the next component j is given by

$$P_{ij}(t) = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j=1}^{\text{Total}-\text{components}} x_j\{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}\}} \tag{1}$$

where x_j is 1 if the component j has not already been used in the partial solution constructed by the ant and 0 otherwise. The parameters α and β determine the relative importance of the pheromone τ_{ij} versus the heuristic information η_{ij}.

Equation (1) is used in all three algorithms. However, ACS is different from the other two algorithms in that it uses a random variable q uniformly distributed over [0, 1] and a parameter q_0 for determining whether (1) will or will not be used for calculating the next transition of an ant. If $q = q_0$, then (1) is not used and the component with maximum value of $\tau_{ij} \cdot \eta_{ij}^{\beta}$ is selected from the available components.

C. Pheromone Update

A user-defined number of ants (called a colony) construct their solutions. When all the ants have constructed their solutions, these solutions are evaluated and pheromone is updated according to the quality of these solutions.

In AS, the pheromone is updated for each of the solutions. The pheromone τ_{ij} is associated with the edge joining the components i and j and is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{t=1}^{m}\Delta\tau_{ij}^{t} \tag{2}$$

where ρ is the evaporation rate, and m is the number of ants. The quantity of pheromone laid on edge (i, j) by ant t is represented by $\Delta\tau_{ij}^{t}$ and it is a function of the quality of the solution constructed by the ant.

In MMAS only the pheromone on the best solution's trail are updated. Furthermore, the value of the pheromone is bound. The pheromone update is implemented as follows:

$$\tau_{ij} \leftarrow \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}}\right]_{\tau_{\min}}^{\tau_{\max}} \tag{3}$$

where $\tau_{\max}$ and $\tau_{\min}$ are, respectively, the user-defined upper and lower bounds on the pheromone. The value inside the bracket is considered as $\tau_{\max}$ if it is greater than $\tau_{\max}$ and $\tau_{\min}$ if it is less than $\tau_{\min}$. The best solution to utilize for the update can be the best one found by the current colony of ants or it can be the best found from the start of the algorithm. This is a user decision.

In ACS there is a local pheromone update after each transition of an ant, in addition to the usual pheromone update. The local update is

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0 \tag{4}$$

where $\varphi \in (0, 1]$ is the pheromone decay coefficient, and τ_0 is the value of the pheromone used for initialization at the beginning of the algorithm. This decreasing of pheromone values on the utilized components discourages subsequent ants of the current colony to traverse the same path. This makes it more likely that different ants will produce different solutions and thus encourages diversity of solutions produced. After a colony of ants has constructed its solutions, the pheromone is updated for the best ant. This best ant can be either the best among the current colony or the best-so-far. The update formula is

$$\tau_{ij} \leftarrow \begin{cases}(1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}, & \text{if}(i,j)\,\text{belongs to the best solution}\\ \tau_{ij}, & \text{otherwise.}\end{cases} \tag{5}$$

For all of these three algorithms, the process of solution construction, evaluation, and pheromone updates by a colony of ants constitutes one pass of the algorithm. The algorithm iterates several times. In each of these iterations, the experience of previous iterations is available to guide the ants in the form of pheromone values. The best solution found during these iterations is retained as the discovered solution.

ACO is naturally suited to discrete optimization problems. Since its inception, it has been applied to solve many problems [2], [4], such as quadratic assignment, job scheduling, subset problems, network routing, vehicle routing, load dispatch in power systems, and bioinformatics. It has also been applied for classification rule discovery, which is the subject of the present work.

D. ACO for the Discovery of Classification Rules

In recent years, several attempts have been made to apply ACO for the discovery of classification rules. The first ACO-based algorithm for this purpose, called AntMiner, was proposed by Parpinelli *et al.* [14]. In AntMiner, given some data, we conduct a heuristic search for finding rules governing the data. The core of AntMiner algorithm is the incremental construction of a classification rule of the type: "IF <term_1 AND term_2 AND ...> THEN <class>" by an ant. Each term is an attribute-value pair related by an operator. The authors only use "=" as the relational operator. An example term is "Color = red." The attribute's name is color and red is one of its possible values. Since only "=" is used, any continuous (real-valued) attributes present in the data have to be discretized in a preprocessing step.

AntMiner is a sequential covering algorithm. It discovers a rule and the training samples correctly covered by this rule (i.e., samples that satisfy the rule antecedent and have the class predicted by the rule consequent) are removed from the training set. Subsequently, the algorithm discovers another rule using the reduced training set and after its discovery the training set is further reduced by removing the training samples covered by the newly discovered rule. This process continues until the training set is empty or almost empty.

For the discovery of each rule, several candidate rules are constructed by artificial ants. An ant constructs a rule by starting with an empty rule and incrementally adding one condition at a time. The choice of adding a term in the current partial rule is made probabilistically and is based on its pheromone and heuristic values. The heuristic used is based on the entropy of terms and their normalized information gain. After the antecedent part of a rule has been constructed, the consequent of the rule is assigned by a majority vote of the training samples covered by the rule. The constructed rule is then pruned of irrelevant terms in an effort to improve its accuracy. The quality of the pruned rule is determined and pheromone values are updated on its basis. Then another rule is constructed. After all ants have constructed their rules, the best one among them is added to the list of discovered rules. The training samples correctly classified by that rule are removed from the training set. Then another rule is discovered in the same manner. The list of discovered rules grows and the algorithm terminates when an exit criterion is met. The output of the algorithm is an ordered set of rules. This rule set can then be used to classify unseen data.

Liu *et al.* proposed density estimation as a heuristic function in AntMiner2 [15] and AntMiner3 [16], [17], and show that this simpler heuristic does the job as well as the complex one used by AntMiner. Another novelty of AntMiner3 is the use of a different pheromone update method. The pheromone is updated and evaporated for only those conditions that occur in the rule and the pheromone values of unused conditions are not evaporated. In this way exploration is encouraged.

The same simpler heuristic is also used in AntMiner+ proposed by Martens *et al.* [18]. However, it differs from the previous AntMiners in many different aspects. A different search environment for the ants is defined, class label of rules is chosen prior to their construction, the ability to form interval rules is incorporated, a different measure for determining rule quality is used, and the selection of two important ACO parameters alpha and beta are made a part of the search process. Only the iteration-best ant is allowed to update the pheromone and the range of the pheromone trail is limited within an interval according to MMAS.

Other works on AntMiner include [19], in which an algorithm for discovering unordered rule sets has been presented. In [20], the PSO algorithm is used for continuous valued attributes and ACO for nominal valued ones and these two algorithms are jointly used to construct rules. The issue of continuous attributes has also been dealt with in [21] and [22]. An AntMiner version for multilabel classification problem can be found in [23]. Relevant background information regarding AntMiners can be found in [24] and [25].

While utilizing ACO-based algorithms, the heuristic for guiding the selection of the next component, along with pheromone values, is very important. The heuristic value gives an indication of the usefulness of taking the next step and thus provides a basis to guide the search. In all the above-mentioned AntMiners, the heuristic value of a candidate term is calculated without considering any other previously chosen term. Our proposed heuristic is based on the strength of the association between the chosen term, the candidate term, and the chosen class label. This idea, which reduces the search space considerably, was first proposed in [26] and [27]. In this paper, we refine the heuristic and describe three ACO-based algorithms for its utilization. These algorithms have the advantage of comprehensibility of discovered rules combined with high accuracy.

III. Correlation and Coverage-Based AntMiner

All the AntMiner algorithms [14]–[18] for ACO-based discovery of classification rules have the following.

1) A solution (or search) space representation in which artificial ants can walk and search for a solution.
2) A policy for selecting components of the solution by the ants taking into account the pheromone and heuristic values (that is, how an ant chalks out a path for itself in the representative solution space).
3) A stopping criterion (or criteria) for ending an ant's walk.
4) A fitness criterion for evaluating the found solution.
5) A policy for updating the pheromone values on the edges between components.
6) A criterion (or criteria) for stopping the ant runs for a particular rule.
7) A criterion (or criteria) for stopping the algorithm.

In the following subsections, we present our search space, an algorithm called AntMiner-CC and its variants, and discuss the above-mentioned aspects one by one.

A. Search Space

Before conducting a search for finding rules governing some given data we need to define a space in which the search can take place. Our search space is basically the same as that of the original AntMiner algorithm [14]. This search space is defined with the help of the training dataset. The dimensions (or coordinates) of the search space are the attributes of the dataset. The possible values of an attribute constitute the range of values for the corresponding dimension in the search space. For example, a dimension called color may have three possible values {red, green, blue}. The task of the ant is to visit a dimension and choose one of its possible values to form an antecedent condition of the rule (e.g., color = red). The possible antecedent conditions (or attribute-value pairs) are called terms, and the total number of terms for a dataset is equal to

$$Total\_terms = \sum_{n=1}^{a} b_n \tag{6}$$

where a is the total number of attributes (excluding the class attribute) and an attribute A_n can take on b_n number of possible values.

The graph for the search space is constructed as follows. Each term corresponds to one vertex of the graph. Thus, there are as many vertices as the number of terms in addition to a start and a stop vertex. The start vertex is forward connected to all the other vertices (except the stop vertex). Furthermore, all the vertices (except the start vertex) are forward connected to the stop vertex. An ant starts its search from the start vertex and ends it at the stop vertex. When an ant has visited a dimension (i.e., a term from an attribute has been chosen), it cannot be visited again by that ant. This prohibition means that rule antecedent conditions of the type "Color = Red OR Color = Green" are not allowed. In the search space there is no ordering in which the dimensions can be visited and an ant may pick its next term from any unvisited dimension. For clarity, the search space can be drawn with all the vertices placed in a single column or layer and this layer duplicated as many times as the number of attributes in the dataset (excluding the class attribute). When arranged in this manner, the vertices of a layer are forward connected only to the vertices of the next layer.

An example search space is shown in Fig. 1(a) and (b). There are four attributes A_1, A_2, A_3, and A_4 having 3, 2, 3, and 2 possible values, respectively. Each of these values (or terms) is represented as a vertex in the graph of Fig. 1(a). An ant starts from the start vertex and constructs a rule by adding conditions (attribute-value pairs or terms) for the antecedent part. After a term has been selected, all the other terms from the same attribute become prohibited for the ant. The rule construction process is stopped when the ant reaches one of the stop vertices. A different representation of the same search space is shown in Fig. 1(b). In this representation, there are as many layers of these vertices as the number of attributes.

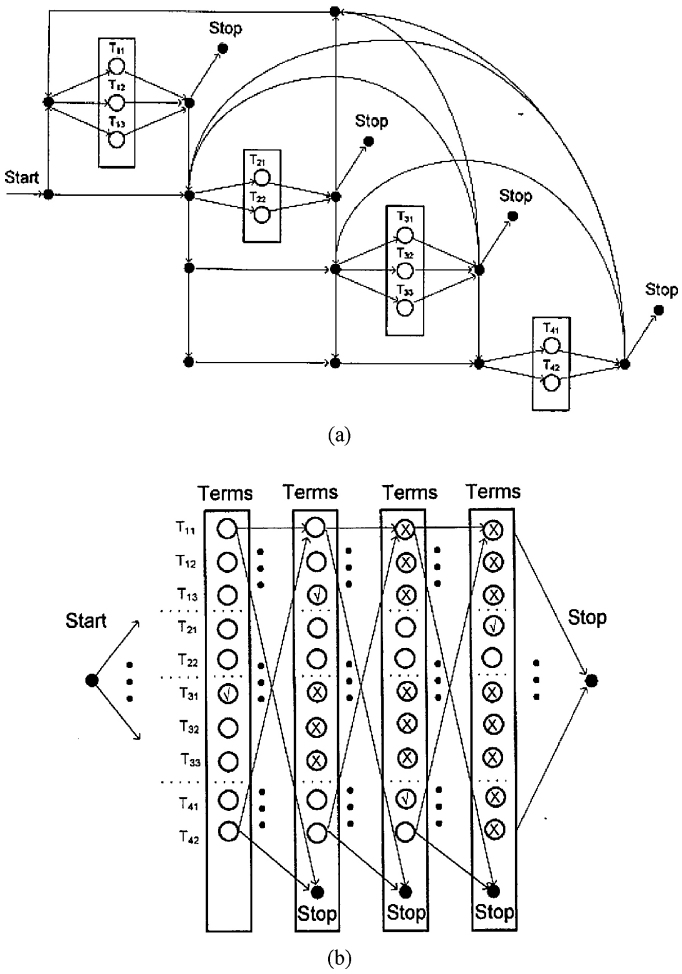

Fig. 1. (a). Example problem's search space represented as a graph. (b) Different representation of the search space shown in (a).

Suppose an ant starts from the start vertex and chooses T_{31} (the first term of the attribute A_3) as the first condition for the rule it is constructing. It cannot subsequently select any more terms from attribute A_3 for the current rule. Further suppose that the ant subsequently chooses T_{13}, T_{41} and T_{21}. The Stop vertex at the end is reached if there are no more permissible terms available. The Stop vertices shown at the bottom can also be reached if one of the two other criteria of stopping are fulfilled (discussed later).

AntMiner+ [18] uses a different search space. Each layer has vertices corresponding to the terms of only one of the attributes. The vertices of one layer are forward connected only to the vertices in the next layer. There is also a start vertex and a stop vertex. The class attribute is also represented in the search space and the attribute-value pairs for this attribute constitute its first layer of vertices. However, the vertex for the attribute-value pair which occurs in majority of data samples is excluded. Two parameters, called alpha and beta, which are used to guide the search are also included in the search space. User defined values of alpha and beta constitute the vertices of the second and third layers. The vertices of alpha are completely connected to the vertices of beta. The discretization of continuous valued attributes is not done as a preprocessing

```
1. Initialize user-defined parameters (number of ants, evaporation rate, number of rules converged, alpha, and beta);
2. TrainingSet = {all training samples};
3. DiscoveredRuleList = {};
4. WHILE (TrainingSet has samples of class labels other than the majority class label)
5.   t = 1;        /* counter for ants */
6.   rcc = 1;      /* counter for rule convergence test */
7.   Select class label from the set of class labels (excluding majority class label);
8.   Set up the search space and initialize all edges with the same amount of pheromone (i.e.
 initialize pheromone matrix);
9.   Calculate heuristic values for all edges taking the selected class label into consideration
            (initialize heuristic matrix);
10.  REPEAT
11.     Send an Ant_t which constructs a classification rule R_t for the selected class;
12.     Assess the quality of the rule and update the pheromone of all trails;
13.     IF (t > 1 AND R_t is equal to R_t-1)   /* update for convergence test */
14.        THEN rcc = rcc + 1;
15.        ELSE rcc = 1;
16.     END IF
17.     t = t + 1;
18.  UNTIL (t ≥ No_of_ants) OR (rcc ≥ No_rules_converg);
19.  Prune the best rule R_best;
20.  Add the pruned best rule R_best to DiscoveredRuleList;
21.  Remove the training samples correctly classified by the pruned best rule R_best;
22. END WHILE
23. Add a default rule in the DiscoveredRuleList;
```

Fig. 2. AntMiner-CC algorithm.

step, but is handled within the search space. For this purpose, each continuous valued attribute has two layers of vertices instead of one. This mechanism enables ants to include interval conditions in their rules. Each of the ants begins its walk by choosing one of the classes (excluding majority class), and then selects a value for alpha and another one for the beta parameter. Due to the forward connection of vertices an ant is bound to move in an ordered fashion from the first layer to the last layer. An ant, however, may or may not choose a term from the layer it is visiting. When a layer has been visited, it cannot be visited again by an ant. Only the last layer of vertices is connected to the stop vertex.

1) *Fixed Order or Flexible Order:* The fixed order of term selection, as is done in AntMiner+, introduces a user-defined bias for splitting the data by attributes occurring earlier in the search space. Hence, we do not impose a fixed order of attributes in our search space.

2) *Class Choice Prior or After Rule Construction:* Experiments show that selection of class label prior to rule construction helps to focus the search [18], [26]. In context of the present research, it also helps by allowing the use of a more efficient heuristic function (described later).

In our algorithm the class label is selected prior to the rule construction, but it is not selected by the individual ants. The class is chosen once and becomes fixed for all the ant runs made in the search of a rule. Class selection is done probabilistically, by roulette wheel selection, on the basis of the weights of classes present in the yet uncovered data samples. The weight of a class is the ratio of its uncovered data samples to the total number of uncovered data samples (excluding the data samples belonging to default class). If 40% of the yet uncovered data samples are of class A, then there is a 40% chance of its selection.

In AntMiner [14], the class label is assigned after a rule has been constructed. It is determined according to the majority class label among the cases covered by the rule. In AntMiner+ [18], the selection of class label is done prior to the rule construction but it is done by each and every ant and does not remain fixed for all the ant runs made for the discovery of a rule.

3) *Selection of Alpha and Beta and Discretization of Continuous Variables:* Alpha and beta are parameters for balancing the relative importance of pheromone and heuristic values. We can determine acceptable values of these parameters by experimenting with different datasets. The selection of alpha and beta by each and every ant, as is done in AntMiner+ [18], increases the complexity of search. Including the determination of these values in the main search process is beneficial only if there is a considerable variation in their values from dataset to dataset. Furthermore, the limited discrete valued options made available for them in the search space, as is done in AntMiner+ [18], introduce additional user-defined parameters.

Similarly, the discretization of continuous variables within the algorithm, as is done in AntMiner+ [18], also introduces complexity and may favor unnecessary long convergence times, with few benefits. There are many efficient discretization algorithms available which can be used as a preprocessing step. In our case, the discretization of continuous valued attributes is done as a preprocessing step and is not handled within the search space (and not even within the algorithm).

4) *Search for Rules of Majority Class:* Rules for majority class samples are not discovered in one of our proposed algorithms. In that algorithm, the majority class label is assigned as the class label of a default rule appended at the end of the discovered rule set. Our experiments (Section IV-A) show that this helps to improve the accuracy of the classifier. Majority class rules are also not discovered by AntMiner+ [18] due to the same reason.

B. Complete Algorithm

The first of our algorithms, AntMiner-CC, is as shown in Fig. 2. The heart of the algorithm is the REPEAT-UNTIL loop where several candidate rules are constructed. The best one among them is pruned and added to the discovered rule list. The samples correctly covered by the discovered rule are removed from the training set and then another rule is discovered (WHILE loop). The output of the algorithm is an ordered set of rules which can be used to classify unseen data.

C. Rule Construction

The main part of the rule discovery process is rule construction. One rule is discovered after one execution of the WHILE loop of Fig. 2. For the discovery of this rule, several rules are constructed and evaluated. Each iteration of the REPEAT-UNTIL loop of Fig. 2 sends an ant to construct a rule. The quality of the rule constructed by an ant is determined and pheromone values are updated and then another ant is sent to construct another rule. Each new ant is guided by the experience of the previous ants, available in the form of pheromone values. The best one among these rules is added to the list of discovered rules after some postprocessing.

Classical ACO algorithms have pheromone updates after several ants have constructed their solutions, whereas in our algorithm, pheromone update is done by each and every ant. Thus, according to classical ACO terminology, our ant colony consists of only one ant. In the rest of this paper, we refer to the ant runs made during the complete execution of REPEAT-UNTIL loop (i.e., one execution of the WHILE loop) by using the words batch of ants which is a short form for batch of 1-ant colonies.

The process of rule construction for a batch of ants starts by assigning a class label to the ants. A class label is chosen from the set of class labels present in the uncovered samples of the training set. Majority class label is excluded from this selection process. The selected class label remains constant for a batch of ants (one iteration of the WHILE loop of Fig. 2).

After the assignment of class label, ants are sent out, one by one, to construct rules. An ant starts with an empty list of conditions and constructs the antecedent part of the rule by adding one condition at a time. The choice of adding a term in the current partial rule is based on the pheromone values and heuristic values associated with the edges from the previously added term to the candidate terms. An ant terminates its rule construction when there are no more attributes left for addition in the rule. Early termination of rule construction is also possible, if any one of the two criteria of stoppage, described later, is met. The different aspects of rule construction are as follows.

1) *Probabilistic Selection of Conditions for Rule Construction:* An ant incrementally adds terms as conditions in the antecedent part of the rule that it is constructing. The selection of the next term is subject to the restriction that a term from the same attribute should not be already 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. Subject to this restriction, the probability of selection of a term for addition in the current partial rule is given by the equation

$$P_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(s)}{\sum_{k=1}^{Total\ terms} x_k\{\tau_{ik}^{\alpha}(t)\eta_{ik}^{\beta}(s)\}} \tag{7}$$

where $\tau_{ij}(t)$ is the amount of pheromone associated with the edge between $term_i$ and $term_j$ for the current ant (the pheromone value is updated after passage of each ant), $\eta_{ij}(s)$ is the value of the heuristic function for the current iteration s of the WHILE loop of Fig. 2. It is constant for a batch of ants. The denominator is used to normalize the numerator value for all the possible choices. The binary variable x_k is set to 1 if a $term_k$ is selectable by the current ant at this stage, otherwise it is 0. Selectable terms belong to those attributes that have not become prohibited due to the selection of a term belonging to them. The $term_j$ has to be one of the selectable terms. The value $\tau_{ik}(t)$ is the amount of pheromone associated with the edge between $term_i$ and $term_k$ for the current ant and $\eta_{ik}(s)$ is its current value of the heuristic function. The parameters α and β are used to control the relative importance of the pheromone and heuristic values in the probability determination of the next movement of the ant.

Equation (7) is an adaptation of the classical ACO probabilistic selection (1) and used in AS, MMAS, and ACS (where the state transition is also dependent on one other equation). The same adaptation is used in AntMiner [14] with $\alpha = \beta = 1$, and also in AntMiner+ [18].

2) *Pheromone Information:* Initialization of pheromone values on an ant's trail and its enhancement or depletion for guiding subsequent ants is one of the basic and important aspects of using ACO metaheuristic. At the beginning of an iteration of the WHILE loop, the pheromone values on edges between all terms are initialized with the same amount of pheromone. The initial pheromone is

$$\tau_{ij}(t = 0) = \frac{1}{\sum_{i=1}^{a} b_i} \tag{8}$$

where a is the total number of attributes (excluding the class attribute) and b_i is the number of possible values that can be taken on by an attribute A_i. Since all the pheromone values are the same, the first ant has no historical information to guide its search. Subsequently, the pheromone values are updated according to a policy explained in Section III-D, and the next ant has access to the experience of previous ant(s).

AntMiner [14] uses (8) for pheromone initialization. AntMiner+ [18] utilizes MMAS and sets all the pheromone equal to a maximum value called $\tau_{\max}$.

3) *Heuristic Information:* Along with pheromone values, the heuristic for guiding the selection of the next component is

very important. The heuristic value gives an indication of the usefulness of taking the next step and thus provides a basis to guide the search. For ACO-based algorithms, heuristic values are either associated with the vertices, or with the edges.

All previously proposed AntMiners use heuristic functions. AntMiner [14] utilizes a heuristic function based on the entropy of the terms and their normalized information gain

$$\frac{\log_2 m - H(W|term_j)}{\sum_{competing\_terms}(\log_2 m - H(W|term_j))} \tag{9}$$

where W is the class attribute whose entropy H given $term_j$ is defined as

$$H(W|term_j) = -\sum_{k=1}^{m}(P(k|term_j).\log_2 P(k|term_j)) \tag{10}$$

where k is one of the values of W and $P(k|term_j)$ is the probability of having class label k for instances containing $term_j$.

AntMiner2 and AntMiner3 [15]–[17] use

$$\frac{|term_j, majority\_class(term_j)|}{|term_j|} \tag{11}$$

and AntMiner+ [18] uses

$$\frac{|term_j, class\_chosen\_by\_ant|}{|term_j|}. \tag{12}$$

Equation (11) is calculated only once for each term for the discovery of a rule in AntMiner2 and AntMiner3. In AntMiner+, (12) is used which is class dependent and has to be calculated as many times as the number of classes (minus the default class) for each term and each discovered rule.

All these versions of AntMiner calculate the heuristic value of a candidate $term_j$ without considering any other previously chosen term. Our novel heuristic function takes into account the correlation of the last selected term with the candidate term along with its potential for maximizing correct coverage.

4) *Correlation:* The first portion of the heuristic function is derived as follows. The actual correlation between the committed $class_k$ and $term_i$ and the candidate $term_j$ is

$$Corr_{k,i,j} = \frac{P(term_i, term_j, class_k)}{P(class_k).P(term_i, term_j)}. \tag{13}$$

Correlation cannot be directly used for comparison purposes because the only information it gives is about negative, positive, or no correlation. It does not specify the degree of correlation. The secondary reason is that it is not bounded between 0 and 1. Hence we simplify it in the following manner.

The value of $P(class_k)$ is same for all competing terms, because it has been previously chosen and is fixed. Hence we may safely ignore it for comparison purposes between the competing terms. Thus, instead of (13) we can use

$$\eta'_{k,i,j} = \frac{P(term_i, term_j, class_k)}{P(term_i, term_j)}. \tag{14}$$

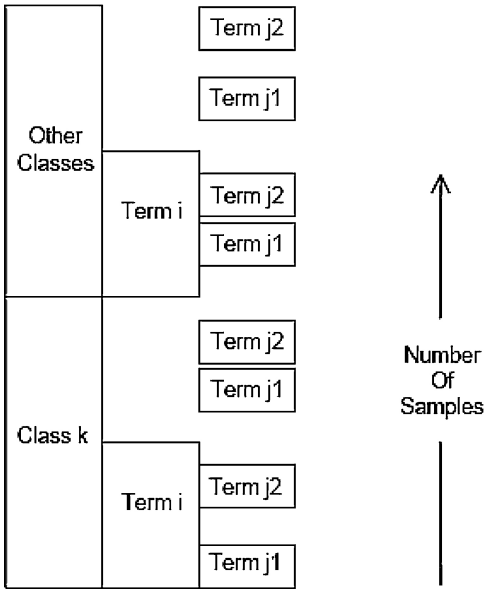

Fig. 3. Example of how the heuristic function facilitates better inter class discrimination and larger correct coverage of the rule.

From the definition of probability, we can rewrite (14) as

$$\eta'_{k,i,j} = \frac{|term_i, term_j, class_k|\ /N}{|term_i, term_j|\ /N} = \frac{|term_i, term_j, class_k|}{|term_i, term_j|}. \tag{15}$$

This gives us the first component of our heuristic function.

5) *Coverage:* We add a second component to encourage the selection of $term_j$ which increases the correct coverage

$$\eta''_{k,i,j} = \frac{|term_i, term_j, class_k|}{|term_i, class_k|}. \tag{16}$$

6) *Overall:* Hence the overall heuristic function is

$$\eta_{k,i,j} = \frac{|term_i, term_j, class_k|}{|term_i, term_j|}.\frac{|term_i, term_j, class_k|}{|term_i, class_k|} = \frac{|term_i, term_j, class_k|^2}{|term_i, term_j|\ .\ |term_i, class_k|}. \tag{17}$$

The first portion gives a higher value to that term that is strongly correlated with the already committed $class_k$ and $term_i$. This means the discovery of a rule with higher confidence ($term_i \wedge term_j => class_k$), if only these two terms are considered. It also encourages early termination of rule construction (described later). The second portion gives a higher value to that term which has larger coverage given $class_k$ and $term_i$.

An example is shown in Fig. 3. Suppose the specified class label is $class_k$. An ant has recently added $term_i$ in its rule. It is now trying to add another term from the set of available terms. We take the case of two competing terms ($term_{j1}$ and $term_{j2}$) which can be generalized to more terms. The heuristic value for $term_{j1}$ given $term_i$ and $class_k$ due to the first portion of the heuristic function is

$$|term_i, term_{j1}, class_k|\ /\ |term_i, term_{j1}| \tag{18}$$

and for $term_{j2}$ given $term_i$ and $class_k$ is

$$|term_i, term_{j2}, class_k|\ /\ |term_i, term_{j2}|. \tag{19}$$

Thus the first portion encourages the inclusion of that term which has better potential for inter class discrimination. We also want the chosen term to maximize correct coverage of the rule as much as possible. This is made possible by the second portion of the heuristic function which assigns the values of

$$\left|term_i, term_{j1}, class_k\right| / \left|term_i, class_k\right| \tag{20}$$

and

$$\left|term_i, term_{j2}, class_k\right| / \left|term_i, class_k\right| \tag{21}$$

for the two competing terms.

Thus our heuristic function quantifies the relationship of the term to be added with the most recently added term and also takes into consideration the overall discriminatory capability of the term to be added. Since the class label of the rule is committed before the construction of the rule antecedents hence our heuristic function is dependent on the class chosen by the ant.

The heuristic values calculated according to (17) are normalized before usage. The heuristic values on edges originating from a $term_i$ are normalized by the summation of all heuristic values on the edges between $term_i$ and other terms.

7) *Heuristic Function for the First Term:* The heuristic value when considering the first term of the rule antecedent is calculated on the basis of the following equation:

$$\eta_{k,start,j} = \frac{\left|term_j, class_k\right|}{\left|term_j\right|} \cdot \frac{\left|term_j, class_k\right|}{\left|class_k\right|} = \frac{\left|term_j, class_k\right|^2}{\left|term_j\right| . \left|class_k\right|}. \tag{22}$$

This equation is obtained by excluding $term_i$ from (17), because it does not exist in this case. Equation (22) provides the advantage of penalizing those terms that would lead to very specific rules and thus helps in avoiding over-fitting. For example, if a term occurs in just one training sample and its class is the chosen class ($class_k$), its heuristic value is low (1/total training samples of $class_k$).

An example heuristic matrix is shown in Fig. 4(a) and (b). The elements of Fig. 4(a) are the heuristic values on edges from Start vertex to vertices of the first layer of terms. They are obtained by using (22). The elements of Fig. 4(b) are heuristic values [obtained by using (17)] on edges between terms in any two consecutive layers. For example, the first row elements of Fig. 4(b) are the heuristic values for edges from the term T_{11} to all other terms in the next layer. These values are the same for edges between 1st and 2nd layer, 2nd and 3rd layer, etc. In other words, the same matrix is used for finding heuristic values for any two consecutive layers. Furthermore, the matrix shown in Fig. 4(b) is asymmetric. The heuristic value for selecting a term given another term is not the same in reverse order of these terms. The heuristic values are calculated once for a batch of ants and are dependent on the class selected for that batch of ants and the samples present in the current training set (i.e., those samples of original training set which are not yet covered by any rule in the discovered rule set). Before placement in this matrix, each raw heuristic value is normalized by division with the summation of other raw heuristic values in its row.

	T_{11}	T_{12}	T_{13}	T_{21}	T_{22}	T_{31}	T_{32}	T_{33}	T_{41}	T_{42}
S	$\eta_{S,11}$	$\eta_{S,12}$	$\eta_{S,13}$	$\eta_{S,21}$	$\eta_{S,22}$	$\eta_{S,31}$	$\eta_{S,32}$	$\eta_{S,33}$	$\eta_{S,41}$	$\eta_{S,42}$

(a)

	T_{11}	T_{12}	T_{13}	T_{21}	T_{22}	T_{31}	T_{32}	T_{33}	T_{41}	T_{42}
T_{11}	0	0	0	$\eta_{11,21}$	$\eta_{11,22}$	$\eta_{11,31}$	$\eta_{11,32}$	$\eta_{11,33}$	$\eta_{11,41}$	$\eta_{11,42}$
T_{12}	0	0	0	$\eta_{12,21}$	$\eta_{12,22}$	$\eta_{12,31}$	$\eta_{12,32}$	$\eta_{12,33}$	$\eta_{12,41}$	$\eta_{12,42}$
T_{13}	0	0	0	$\eta_{13,21}$	$\eta_{13,22}$	$\eta_{13,31}$	$\eta_{13,32}$	$\eta_{13,33}$	$\eta_{13,41}$	$\eta_{13,42}$
T_{21}	$\eta_{21,11}$	$\eta_{21,12}$	$\eta_{21,13}$	0	0	$\eta_{21,31}$	$\eta_{21,32}$	$\eta_{21,33}$	$\eta_{21,41}$	$\eta_{21,42}$
T_{22}	$\eta_{22,11}$	$\eta_{22,12}$	$\eta_{22,13}$	0	0	$\eta_{22,31}$	$\eta_{22,32}$	$\eta_{22,33}$	$\eta_{22,41}$	$\eta_{22,42}$
T_{31}	$\eta_{31,11}$	$\eta_{31,12}$	$\eta_{31,13}$	$\eta_{31,21}$	$\eta_{31,22}$	0	0	0	$\eta_{31,41}$	$\eta_{31,42}$
T_{32}	$\eta_{32,11}$	$\eta_{32,12}$	$\eta_{32,13}$	$\eta_{32,21}$	$\eta_{32,22}$	0	0	0	$\eta_{32,41}$	$\eta_{32,42}$
T_{33}	$\eta_{33,11}$	$\eta_{33,12}$	$\eta_{33,13}$	$\eta_{33,21}$	$\eta_{33,22}$	0	0	0	$\eta_{33,41}$	$\eta_{33,42}$
T_{41}	$\eta_{41,11}$	$\eta_{41,12}$	$\eta_{41,13}$	$\eta_{41,21}$	$\eta_{41,22}$	$\eta_{41,31}$	$\eta_{41,32}$	$\eta_{41,33}$	0	0
T_{42}	$\eta_{42,11}$	$\eta_{42,12}$	$\eta_{42,13}$	$\eta_{42,21}$	$\eta_{42,22}$	$\eta_{42,31}$	$\eta_{42,32}$	$\eta_{42,33}$	0	0

(b)

Fig. 4. Heuristic values for the example shown in Fig. 1.

The correlation-based heuristic function has the potential to be effective in large dimensional search spaces. The heuristic values are calculated only once for an iteration of the WHILE loop of Fig. 2 and remains same for a batch of ants. It assigns a zero value to the combination of those terms which do not occur together for a given class label; thus efficiently restricting the search space for the ants.

Equations (17) and (22) have not been used before in any ACO-based classification rule discovery algorithm. In contrast to our approach, each ant of AntMiner [14] continues its attempts to add terms in its rule until it is sure that there is no term whose addition would not violate the condition of minimum number of covered samples. In AntMiner+ [18], every ant has to traverse the whole search space and there are no such short cuts.

8) *Stopping Criterion for a Single Ant Run:* Our basic criterion for the stoppage of rule construction is that one condition from every attribute has been included in the rule [shown as Stop on the right-hand side in Fig. 1(b)]. In addition to this, we use the criterion of homogenous class label to stop the construction of a rule. The rule construction stops when all the samples covered by the rule have homogenous class labels. The third scenario in which rule construction stops is that the probability of transition to all remaining permissible terms is zero. The last two criteria are shown as Stop vertices at the bottom of Fig. 1(b).

AntMiner [14] tries to add one term from each attribute, but stops the rule construction process if only those terms are left unused whose addition will make the rule cover a number of training samples smaller than a user-defined threshold called minimum samples per rule. In AntMiner+ [18], one term from each attribute is added. However, each attribute has a "don't care" option that allows for its non-utilization in the rule.

D. Rule Fitness and Pheromone Update

1) *Fitness or Quality of a Rule:* ACO requires the evaluation of generated solutions and uses it as a feedback for generation of new solutions. The correctness of this evaluation has direct consequences on the search process. For classification rule discovery problem we would like to have rules with high confidence and large coverage. We measure the quality, Q, of a rule by adding its confidence and coverage

$$Q = \frac{TP}{Covered} + \frac{TP}{N} \tag{23}$$

where TP is the number of samples covered by the rule that have the same class label as that of the rule's consequent, *Covered* is the total number of samples covered by the rule, and N is the number of samples in the current training set (i.e., those samples of original training set which are not yet covered by any rule in the discovered rule set).

Equation (23) has been used in AntMiner+ [18], whereas AntMiner [14] uses sensitivity multiplied by specificity as the quality measure

$$Q = \frac{TP}{TP + FN} \cdot \frac{TN}{FP + TN} \tag{24}$$

where FP is the number of samples covered by the rule that have a class label different from that of the rule's consequent (false positives), FN is the number of samples that are not covered by the rule but that have the same class label as that of the rule's consequent (false negatives), TN is the number of samples that are not covered by the rule and that have a class label different from that of the rule's consequent (true negatives).

2) *Updating of Pheromone Values on Edges:* The quality of the generated rule is made available to the future ants by means of pheromone update on edges. The pheromone values of the whole system can be represented in the form of a matrix. An example pheromone matrix is shown in Fig. 5. The pheromone values are associated with the edges of the search space. The elements of Fig. 5(a) are the pheromone values on edges from Start vertex to vertices of the first layer of terms. The elements of Fig. 5(b) are pheromone values on edges between terms in any two consecutive layers. In other words, the matrix values are valid for any two consecutive layers and the pheromone values on edges originating from a term to other terms are the same in all the layers of the search space. For example, the first row elements of Fig. 5(b) are the pheromone values for edges from the term T_{11} to all other terms in the next layer. These values are the same for edges between first and second layers, second and third layers, etc. If an ant chooses the terms T_{31}, T_{13}, T_{41} and T_{21} for its rule, then the elements $\tau_{S,31}$, $\tau_{31,13}$, $\tau_{13,41}$ and $\tau_{41,21}$ are updated according to (25) (given below). Elements of rows S, T_{31}, T_{13}, T_{41}, and T_{21} are then normalized. The remaining rows remain unchanged. The pheromone matrix is asymmetric.

The amount of pheromone on the edges between consecutive terms occurring in the generated rule is updated according to the equation

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \left(1 - \frac{1}{1+Q}\right)\tau_{ij}(t) \tag{25}$$

	T_{11}	T_{12}	T_{13}	T_{21}	T_{22}	T_{31}	T_{32}	T_{33}	T_{41}	T_{42}
S	$\tau_{S,11}$	$\tau_{S,12}$	$\tau_{S,13}$	$\tau_{S,21}$	$\tau_{S,22}$	$\tau_{S,31}$	$\tau_{S,32}$	$\tau_{S,33}$	$\tau_{S,41}$	$\tau_{S,42}$

(a)

	T_{11}	T_{12}	T_{13}	T_{21}	T_{22}	T_{31}	T_{32}	T_{33}	T_{41}	T_{42}
T_{11}	0	0	0	$\tau_{11,21}$	$\tau_{11,22}$	$\tau_{11,31}$	$\tau_{11,32}$	$\tau_{11,33}$	$\tau_{11,41}$	$\tau_{11,42}$
T_{12}	0	0	0	$\tau_{12,21}$	$\tau_{12,22}$	$\tau_{12,31}$	$\tau_{12,32}$	$\tau_{12,33}$	$\tau_{12,41}$	$\tau_{12,42}$
T_{13}	0	0	0	$\tau_{13,21}$	$\tau_{13,22}$	$\tau_{13,31}$	$\tau_{13,32}$	$\tau_{13,33}$	$\tau_{13,41}$	$\tau_{13,42}$
T_{21}	$\tau_{21,11}$	$\tau_{21,12}$	$\tau_{21,13}$	0	0	$\tau_{21,31}$	$\tau_{21,32}$	$\tau_{21,33}$	$\tau_{21,41}$	$\tau_{21,42}$
T_{22}	$\tau_{22,11}$	$\tau_{22,12}$	$\tau_{22,13}$	0	0	$\tau_{22,31}$	$\tau_{22,32}$	$\tau_{22,33}$	$\tau_{22,41}$	$\tau_{22,42}$
T_{31}	$\tau_{31,11}$	$\tau_{31,12}$	$\tau_{31,13}$	$\tau_{31,21}$	$\tau_{31,22}$	0	0	0	$\tau_{31,41}$	$\tau_{31,42}$
T_{32}	$\tau_{32,11}$	$\tau_{32,12}$	$\tau_{32,13}$	$\tau_{32,21}$	$\tau_{32,22}$	0	0	0	$\tau_{32,41}$	$\tau_{32,42}$
T_{33}	$\tau_{33,11}$	$\tau_{33,12}$	$\tau_{33,13}$	$\tau_{33,21}$	$\tau_{33,22}$	0	0	0	$\tau_{33,41}$	$\tau_{33,42}$
T_{41}	$\tau_{41,11}$	$\tau_{41,12}$	$\tau_{41,13}$	$\tau_{41,21}$	$\tau_{41,22}$	$\tau_{41,31}$	$\tau_{41,32}$	$\tau_{41,33}$	0	0
T_{42}	$\tau_{42,11}$	$\tau_{42,12}$	$\tau_{42,13}$	$\tau_{42,21}$	$\tau_{42,22}$	$\tau_{42,31}$	$\tau_{42,32}$	$\tau_{42,33}$	0	0

(b)

Fig. 5. Pheromone values for the example shown in Fig. 1.

where $\tau_{ij}(t)$ is the pheromone value encountered by Ant_t (the tth ant of the REPEAT-UNTIL loop) between $term_i$ and $term_j$. The pheromone evaporation rate is represented by ρ and Q is the quality of the rule constructed by Ant_t.

The equation updates pheromone values by first evaporating a percentage of the previously occurring pheromone and then adding a percentage of the pheromone dependent on the quality of the recently discovered rule. If the rule is good then the pheromone added is greater than the pheromone evaporated and the terms become more attractive for subsequent ants. The evaporation in the equation improves exploration, otherwise in the presence of a static heuristic function the ants tend to converge quickly to the terms selected by the first few ants. Equation (25) is according to AS and has been previously used in AntMiner3, but with a different equation for determining Q. It can also be shown to be equivalent to the ACS nonlocal pheromone update equation.

Note that our pheromone matrix is asymmetric whereas the matrix used in original AntMiner [14] is symmetric. The matrix becomes asymmetric because for a constructed rule with $term_i$ occurring immediately before $term_j$, the pheromone on edge between $term_i$ and $term_j$ is updated but the pheromone on edge between $term_j$ and $term_i$ is not updated. This is done to encourage exploration and discourage early convergence.

The next step is to normalize the pheromone values. Each pheromone value is normalized by dividing it by the summation of pheromone values on edges of all its competing terms. In the pheromone matrix (Fig. 5), normalization of the elements is done by dividing them with the summation of values of the row to which they belong. Note that for those rows for which no change in the values has occurred for the ant run under consideration, the normalization process yields the same values as before. Referring to Fig. 5, normalization

of pheromone value of an edge originating from a term is done by dividing it by the summation of all pheromone values of the edges originating from that term. This process changes the amount of pheromone associated with those terms that do not occur in the most recently constructed rule but are competitors of the selected term. If the quality of rule has been good and there has been a pheromone increase on the terms used in the rule then the competing terms become less attractive for the subsequent ants. The reverse is true if the rule found is not of good quality. The normalization process is an indirect way of simulating evaporation of pheromone. Note that in original AntMiner [14] every element of the pheromone matrix is normalized by dividing it with the sum of all elements. This is unnecessary and tends to discourage exploration and favor early convergence.

E. Stopping Criterion for a Batch of Ants

For the discovery of one rule, a batch of ants is sent. The ants are sent one by one and each ant constructs one candidate rule. After the construction of a candidate rule its quality is determined and the pheromone values on the trail are updated accordingly. The pheromone values (along with the heuristic values) guide the construction of next candidate rule. Theoretically there may be as many candidate rules as the user defined number of ants. An early termination of a batch run (REPEAT-UNTIL loop) is possible if the last few ants have constructed the same candidate rule. This implies that the pheromone values on a trail have become very high and convergence has been achieved. Since the heuristic values for a batch of ants are static, any further rule construction will most probably yield the same rule again. Hence the batch of ants can be terminated prematurely. For this purpose, each constructed rule is compared with the last rule and a counter is incremented if both the rules are the same. If the value of this counter exceeds a threshold "No. of rules converged," then the loop is terminated. In our experiments, we use a value of 10 for this threshold.

This method of early termination of REPEAT-UNTIL loop is also used by AntMiner. In AntMiner+ there is no provision for early termination of this loop and it terminates when the pheromone values on one path converge to $\tau_{\max}$ and all other paths have $\tau_{\min}$, as required by the MMAS.

The best one among the candidate rules constructed by a batch of ants is selected, postprocessed and added in the discovered rule list. The dataset is emptied of the training samples covered by that rule. Then the whole process is repeated with the reduced dataset.

F. Postprocessing of Discovered Rules

After a candidate rule has been discovered we are faced with the question: can we further improve its quality? One way to improve a rule's quality is by trying to prune it. Rule pruning is the process of finding and removing any irrelevant terms that might have been included in the constructed rule. Rule pruning has two advantages. First, it increases the generalization of the rule thus potentially increasing its predictive accuracy. Second, a shorter rule is usually simpler and more comprehensible.

The rule pruning procedure starts with the full rule. It temporarily removes the first term and determines the quality of the resulting rule. It then replaces the term back and temporarily removes the second term and again calculates the quality of the resulting rule. This process continues until all the terms present in the rule are dealt with. After this assessment, if there is no term whose removal improves or maintains the quality then the original rule is retained. However, if there is a term whose removal improves (or maintains) the quality of the rule then that term is permanently removed. If there are two or more such terms then the term whose removal most improves the quality of the rule is removed. The shortened rule is again subjected to the procedure of rule pruning. The process continues until any further shortening is impossible (removal of any term present in the rule leads to decrease in its quality) or if there is only one remaining term in the rule.

It is pertinent to note that we again use the confidence and coverage as the rule's quality measure (23) for pruning purpose, but another quality measure could also have been used (e.g., confidence only). We also recall that the construction of a rule continues until all the samples covered by it are of homogenous class label or if there are no more terms to add (Section III-C). In the first case of homogenous class label samples, the confidence of the rule is already 100%. Any decrease in confidence must be compensated by an increase in coverage. A rule may thus get pruned on the basis of (23) with lower confidence but better coverage than before. Pruning cannot result in a decrease in coverage.

Rule pruning is a costly procedure. That is why we select and try to prune only one, top quality, rule after all rules have been discovered by a batch of ants. This means that pheromone updates are done with un-pruned rules. The ideal option would have been to check the rules for eligibility of pruning and pruning them as soon as they are discovered, before updating the pheromone values for the next ant.

The best rule is placed in the discovered rule set after pruning and the training samples correctly covered by the rule are flagged (i.e., removed) and have no role in the discovery of other rules.

Rule pruning is also used by AntMiner [14] and AntMiner+ [18]. However, the rule quality criterion used in AntMiner is different. Furthermore, in AntMiner each discovered rule is subjected to rule pruning (prior to pheromone update) whereas we prune only the best rule among the rules found during the execution of the REPEAT-UNTIL loop of the algorithm and our pheromone update is prior to and independent of rule pruning. AntMiner+ also prunes the best rule. However, it is pertinent to note that AntMiner+ has several iterations of the REPEAT-UNTIL loop and in each iteration there are a 1000 ant runs. The best rule from these 1000 rules is pruned. In other words, AntMiner+ prunes several rules for each rule inserted in the rule set whereas we prune only one rule.

G. Final Rule Set

The best rule found after a complete execution of WHILE loop of Fig. 2 is placed in the discovered rule set after pruning. The training samples correctly covered by the rule are flagged

(i.e., removed) and they have no role in the subsequent rule discovery process.

1) *Stopping Criterion for Addition of Rules in the Rule Set:* After several iterations of the WHILE loop of Fig. 2, the training set is almost empty and only a few samples are left. Now we are faced with the question of when to stop this rule discovery process. There are four major options.

1) The rule discovery process can be continued until the training dataset is empty.
2) A separate validation set can be used to monitor the training. The rule discovery process can be stopped when the accuracy on the validation set starts to dip.
3) Specify a threshold on the number of training samples present in the data set. If, after an iteration of the REPEAT-UNTIL loop, the remaining samples in the training set are equal to or below this specified threshold, then the algorithm can be stopped. The threshold can be defined as a percentage of the samples present in the initial training set.
4) Specify a threshold on the maximum number of rules to be found. The rule discovery process is stopped when the number of discovered rules becomes equal to the specified maximum limit.

All of these options have some drawbacks. The first option usually leads to rules discovered at the tail-end covering only one or two of the remaining training samples. Such rules possibly over-fit the data and usually do not have generalization capabilities. The validation set option is not appropriate for small datasets because we have to divide the total data samples into training, validation, and test sets. The third and fourth options both necessitate a user-defined threshold parameter. The correct specification of such parameters is usually problematic because they are dependent on the dataset.

All the options, except the first one, result in residual training samples uncovered by any of the discovered rules. Usually, the information contained in these samples is partially harnessed by adding a default rule at the bottom of the rule set according to class label in majority among them.

All AntMiners employ early stopping of the rule discovery process. AntMiner [14] uses a user-defined threshold called "Max_uncovered_samples." The algorithm checks whether the uncovered training samples are still above the value for this threshold. If that is the case, a new iteration of the WHILE loop starts for the discovery of the next rule. If not, the rule discovery process is terminated. AntMiner+ [18] uses the validation set technique for large datasets (greater than 250 samples) and a threshold of 1% remaining samples for small datasets.

We recommend the following approach, taking into account the fact that the validation set option is not feasible for small- and medium-sized datasets, and also that we need the default rule to represent the majority class whose rules we do not discover. We continue to discover rules till the training set is completely empty except for the samples belonging to the majority class. In this way, we do not need any user-defined parameter for stopping the discovery of rules. Furthermore, there is no information lost due to the nonhomogeneity of residual samples.

2) *Default Rule:* After the completion of rule discovery process, a default rule may be added at the bottom of the rule set. The default rule is without any conditions and has a consequent part only. This default rule accounts for those test cases which are not covered by any of the discovered rules.

Since we do not discover rules for samples of the majority class label, we add a default rule with the class label of majority class in the training set samples.

The default rule is used by all AntMiners. Since AntMiner [14] uses a threshold for terminating the rule discovery process, there are residual uncovered training samples in the training set. The majority class label of these remaining uncovered samples is assigned as the class label for the default rule. In AntMiner+ [18], the default rule class label is assigned on the basis of majority class of the complete set of training samples. AntMiner2 and AntMiner3 also assign default rule class label on the basis of majority class of the complete set of training samples. However, the rule discovery process of AntMiner2 and AntMiner3 includes discovery of rules for the majority class, whereas AntMiner+ does not discover them.

3) *Use of Discovered Rule Set for Classifying Unseen Samples:* After the addition of default rule, the rule set is ready to be used for classifying unseen samples. A new test sample, unseen during training, is classified by checking the rules in order of their discovery. The first rule whose antecedents match the test sample is fired and the class predicted by the rule's consequent is assigned to the sample. If none of the discovered rules are fired then the final default rule gets activated and fired.

H. Comprehensibility of the Rule Set

One issue regarding comprehensibility of the rule set discovered by AntMiner-CC is the absence of rules for the majority class. A second, and bigger, issue is that the discovered rule set is an ordered one, and each rule has to be understood in the context of the rules preceding it. For rectifying these two problems, we propose two variants of AntMiner-CC.

1) *Rules for Majority Class:* If we wish to discover rules for the majority class also, then the algorithm of Fig. 2 can be modified to allow the majority class to compete with other classes while selecting a class before the execution of the REPEAT-UNTIL loop. Apart from this change, the rest of the algorithm remains the same. The default class at the bottom of the rule set is still according to the class label of the majority of the samples in the training set. We call this algorithm MAntMiner-CC.

2) *Unordered Rule Set:* An AntMiner has been proposed in [19] for discovering unordered set of rules. For an unordered rule set there is no need to apply the discovered rules in order and therefore they can be interpreted independently of each other.

AntMiner-CC has been modified to obtain an unordered rule set. A general description of the resulting algorithm, called UAntMiner–CC, is shown in Fig. 6. The FOR loop iterates once for each class value in the class attribute domain. For all class values, it starts with the full training set. This ensures availability of a maximal number of negative examples to the

algorithm. The samples of the selected class are considered as positive samples and all other samples serve as negative samples. The algorithm creates a best rule that covers a subset of the training data, adds the best rule in the discovered rules list and removes the training samples that are correctly classified by this rule. This process continues until the training set is empty of samples belonging to the class value being considered. The training set is then again reinstated for another class value.

The algorithm terminates when the FOR loop has iterated over all the possible class values. The output of the algorithm is an unordered set of rules. A default rule is added at the bottom of the rule set. It has no conditions and has only a consequent. The consequent is the class label in majority among training set samples. This set can then be used to classify unseen data.

Since the rules can be applied in any order, several rules may get activated for a given test sample and, in such cases, a conflict resolution strategy has to be used. In our experiments, we use the conflict resolution strategy that fires the rule with the highest confidence. In case none of the discovered rules is activated, the default rule is fired.

UAntMiner-CC has another advantage. Since discovery of rules for a class is independent of the other classes, we can parallelize the rule discovery process.

IV. Experiments and Analysis

In this section, we describe our experimental setup and report the performance of the proposed algorithms. We also present our experiments for validating some of the design choices of these algorithms.

A. Comparisons on Basis of Predictive Accuracy

1) *Datasets:* In our experiments, we use a suite of 26 datasets obtained from the UCI repository [28] to report the performance of the proposed algorithms and their comparison with other algorithms. The main characteristics of the suite of datasets are summarized in Table I. These datasets are commonly used by researchers and the suite has reasonable variety in terms of number of attributes, number of samples, and number of classes.

The proposed algorithms work with categorical attributes and continuous attributes need to be discretized in a preprocessing step. We use unsupervised discretization filter of Weka-3.4 machine learning tool [12] for discretizing continuous attributes in the datasets. This filter first computes the intervals of continuous attributes and then discretizes them on the basis of these intervals.

2) *Performance Metric and Choice of Parameters:* There are many performance metrics that can be used when evaluating classification algorithms. Many of them are based on the number of correct and incorrect class label predictions by the resulting classifier. For classification rule discovery algorithms, we can also use number of discovered rules and number of conditions per rule as an indirect measure of the classifier comprehensibility (the smaller, the better). Rule interestingness measures can also be used for comparison on the basis of novelty of discovered rules. In this paper, we use predictive accuracy as our main performance metric. It is defined as the percentage of testing samples correctly classified by the classifier. This is a popular performance metric for general comparison between classification algorithms and has been used in all the previous AntMiner research.

TABLE I
Suite of Datasets Used in Our Experiments

Dataset	Attributes	Samples	Classes
Balance-scale	4	625	3
Breast Cancer–Wisconsin (BC-W)	9	683	2
Car	6	1728	4
Congress House Votes	17	435	2
Credit (Australia)	15	690	2
Credit (Germany)	19	1000	2
Dermatology	33	366	6
Ecoli	7	336	8
Glass	9	214	7
Haberman	3	307	2
Hayes Roth	6	132	3
Heart	13	270	2
Hepatitis	19	155	2
Image Segmentation	19	210	7
Ionosphere	34	351	2
Iris	4	150	3
Mammographic—Mass	5	961	2
Pima Indian Diabetes	8	768	2
SPECT (Heart)	22	267	2
Teacher Assistant Evaluation (TAE)	6	151	3
Tic-tac-toe	9	958	2
Transfusion	4	748	2
Vehicle	18	282	4
WDBC	31	569	2
Wine	13	178	3
Zoo	16	282	7

The experiments are performed using a tenfold cross-validation procedure. A dataset is divided into 10 equally sized, mutually exclusive subsets. Each of the subset is used once for testing while the other nine are used for training. The results of the 10 runs are then averaged and this average is reported as the final result. Standard deviation is also reported along with the averages.

AntMiner-CC has the following user-defined parameters:

1) number of ants (this is the short form for maximum size of a batch of 1-ant colonies);
2) number of rules converged (for early termination of REPEAT-UNTIL loop);
3) powers of pheromone and heuristic values (α, β) used in (7);
4) pheromone evaporation rate used in (25).

According to our experiments, described later (Table VI, Section IV-B), the number of 1000 ants is an adequate value because it is used in conjunction with the early termination of the REPEAT-UNTIL loop if rule convergence is achieved. The parameter "number of rules converged" is an indirect indication of pheromone saturation and does not seem to be a sensitive parameter as long as it is not a very small number. We use the value of 10 for this parameter; a number which has

```
1. Initialize user-defined parameters (number of ants, evaporation rate, number of rules converged, alpha, and beta);
2. TrainingSet = {all training samples};
3. No_of_Classes = Total classes present in the training set;
4. DiscoveredRuleList = {};
5. FOR (jj = 1 to No_of_Classes)
6.   TrainingSet = {all training samples};
7.   WHILE (TrainingSet has samples of current class jj)
8.    t = 1;          /* counter for ants */
9.    rcc = 1;        /* counter for rule convergence test */
10.      Set up the search space and initialize all edges with the same amount of
                        pheromone (i.e. initialize pheromone matrix);
11.      Calculate heuristic values for all edges taking the current class label into
                        consideration (initialize heuristic matrix);
12.      REPEAT
13.         Send an Ant_t which constructs a classification rule R_t for the current
                                class;
14.         Assess the quality of the rule and update the pheromone of all trails;
15.         IF (t > 1 AND R_t is equal to R_(t-1))   /* update for convergence test */
16.            THEN rcc = rcc + 1;
17.            ELSE rcc = 1;
18.         END IF
19.         t = t + 1;
20.      UNTIL (t ≥ No_of_ants) OR (rcc ≥ No_rules_converg);
21.      Prune the best rule R_best;
22.      Add the pruned best rule R_best to DiscoveredRuleList;
23.      Remove the training samples correctly classified by the pruned best rule R_best;
24.   END WHILE
25. END FOR
26. Add a default rule in the DiscoveredRuleList
```

Fig. 6. UAntMiner-CC algorithm.

TABLE II
PARAMETERS USED IN EXPERIMENTS

Parameter	Value
Number of ants	1000
Evaporation rate	0.10
No. of rules converged	10
Alpha	1
Beta	1

also been used by the previous AntMiner versions reported in [14] and [18].

The relationship between α, β, and ρ is a complex one and needs to be determined experimentally. For a given dataset, accuracy results can be obtained using different combinations of these three parameters, and the best ones can be retained. In the experiments reported in this paper, we make no efforts at such an optimization and arbitrarily use $\alpha = \beta = 1$ and $\rho = 0.1$ for all datasets. The complete list of parametric values used in our experiments is shown in Table II.

3) *Predictive Accuracy Results:* We compare the results of our algorithm with those for AntMiner, AntMiner2, AntMiner3, AntMiner+, C4.5 [29], [30], Ripper, AdaBoost, *k*-nearest neighbor, logistic regression, naive Bayes, and support vector machines (SVM) [31]. AntMiner-CC and its variants have been implemented using C-Sharp language. AntMiner, AntMiner2, and AntMiner3 algorithms have been implemented in MATLAB 7.0. AntMiner is also available from [32]. For AntMiner+, we have used the software available from [33]. For the rest of the algorithms we use the Weka machine learning tool [12]. The values of two parameters, maximum uncovered cases and minimum cases per rule, used in AntMiner, AntMiner2, and AntMiner3, are both set as 10 [14]–[17]. The predictive accuracies of the compared algorithms are shown in Table III(a)–(c). Tenfold cross-validation is used to obtain the results.

The results indicate that AntMiner-CC and its variants achieve higher accuracy rate than the compared algorithms for most of the datasets. In order to analyze whether the performance of AntMiner-CC is significantly better than the others we have employed the Friedman test. The Friedman test is a nonparametric statistical test and is chosen because it does not make any assumptions about the distribution of the underlying data and is a suitable test for comparing a set of classifiers over multiple data sets [34], [35].

Table IV presents the comparisons of the best algorithm (the algorithm with the best average rank, considered as control algorithm: AntMiner-CC, in our case) with the remaining algorithms according to the Friedman test in terms of predicative accuracy values for all the datasets. For each algorithm, the average rank (the lower the average rank the better the algorithm's performance), the *p*-value (when the average rank is compared to the average rank of the algorithm with the best

TABLE III (a)

AVERAGE PREDICTIVE ACCURACIES OBTAINED USING TENFOLD CROSS-VALIDATION FOR THE ALGORITHMS PROPOSED IN THIS PAPER

Datasets	**AntMiner–CC**	**MAntMiner-CC**	**UAntMiner-CC**
Balance-scale	**91.84 ± 4.55**	90.4 ± 5.22	90.57 ± 4.55
BC-W	**97.71 ± 1.80**	96.43 ± 1.55	96.57 ± 1.93
Car	93.06 ± 1.64	**97.74 ± 1.11**	94.56 ± 1.61
Congress House Votes	95.65 ± 3.45	95.89 ± 4.65	**96.8 ± 2.88**
Credit (Aus)	**90.43 ± 2.47**	85.51 ± 3.13	89.28 ± 3.70
Credit (Ger)	**86.59 ± 3.26**	72.18 ± 5.28	73.47 ± 9.17
Dermatology	**94.26 ± 3.98**	93.17 ± 3.19	88.24 ± 8.57
Ecolli	86.58 ± 5.53	**87.78 ± 6.93**	77.59 ± 8.60
Glass	**72.45 ± 11.38**	71.56 ± 6.39	65.06 ± 10.44
Haberman	79.53 ± 6.86	**82.76 ± 9.61**	82.44 ± 9.36
Hayes Roth	**88.68 ± 7.37**	85 ± 12.02	85.6 ± 12.17
Heart	**90.74 ± 6.10**	84.81 ± 4.43	85.19 ± 7.81
Hepatitis	**94.21 ± 4.78**	88.38 ± 5.79	87.17 ± 10.28
Image Segmentation	**89.52 ± 6.67**	85.71 ± 4.49	72.86 ± 10.31
Ionosphere	**94 ± 3.13**	85.14 ± 6.29	90.29 ± 8.85
Iris	**98.67 ± 2.81**	98 ± 3.23	91.33 ± 5.50
Mammographic—Mass	68.27 ± 5.15	76.48 ± 4.74	**83.66 ± 3.76**
Pima Indian Diabetes	**87.9 ± 2.88**	80.76 ± 7.15	82.31 ± 6.67
SPECT (Heart)	84.99 ± 8.16	80.94 ± 8.22	**86.81 ± 7.37**
TAE	**84.75 ± 13.38**	78.17 ± 15.37	79.46 ± 11.10
Tic-tac-toe	98.22 ± 1.83	97.08 ± 2.69	**99.27 ± 0.85**
Transfusion	47.66 ± 7.30	78.91 ± 5.34	**79.57 ± 5.28**
Vehicle	**82.34 ± 7.56**	82.29 ± 9.27	79.45 ± 5.19
WDBC	**95.61 ± 2.06**	91.74 ± 5.50	90.87 ± 5.79
Wine	**98.86 ± 2.40**	98.3 ± 2.75	96.6 ± 4.81
Zoo	**95.09 ± 5.19**	95.09 ± 5.19	90.09 ± 9.42

The highest accuracy for a given dataset is shown in bold.

TABLE III (b)

AVERAGE PREDICTIVE ACCURACIES OBTAINED USING TENFOLD CROSS-VALIDATION FOR ALGORITHMS THAT DISCOVER COMPREHENSIBLE RULE SETS

Datasets	**AntMiner**	**AntMiner2**	**AntMiner3**	**AntMiner+**	**C4.5**	**Ripper**
Balance-scale	75.32 ± 8.86	72.78 ± 9.23	75.06 ± 6.91	35.88 ± 15.00	83.02 ± 3.24	80.93 ± 3.35
BC-W	94.64 ± 2.74	92.70 ± 2.82	93.56 ± 3.45	49.1 ± 20.72	94.84 ± 2.62	95.57 ± 2.17
Car	82.38 ± 2.42	81.89 ± 2.63	78.82 ± 3.76	56.93 ± 19.90	96.0 ± 2.13	89.17 ± 2.52
Congress House Votes	94.54± 2.27	95.76 ± 2.43	94.72 ± 3.42	79.25 ± 10.89	95.31 ± 2.57	95.66 ± 2.75
Credit (Aus)	86.09 ± 4.69	84.20 ± 4.55	86.67 ± 5.46	46.52 ± 17.04	81.99 ± 7.78	86.07 ± 2.27
Credit (Ger)	71.62 ± 2.71	73.16 ± 5.21	72.07 ± 4.32	41.13 ± 14.43	70.73 ± 6.71	70.56 ± 5.96
Dermatology	58.72 ± 7.36	59.18 ± 14.91	67.47 ± 9.57	28.27 ± 20.94	95.07 ± 2.80	93.96 ± 3.63
Ecolli	47.52 ± 11.32	43.09 ± 9.76	44.61 ± 10.32	21.16 ± 10.26	82.99 ± 7.72	79.18 ± 2.22
Glass	53.33 ± 4.38	56.15 ± 10.32	46.36 ± 10.96	15.28 ± 11.47	68.90 ± 8.98	70.48 ± 8.19
Haberman	71.99 ± 7.57	73.94 ± 5.33	74.72 ± 4.62	66.08 ± 12.99	73.88 ± 4.66	72.47 ± 6.74
Hayes Roth	75.05 ± 10.62	70.88 ± 15.47	85.49 ± 7.76	40.93 ± 12.14	80.93 ± 8.24	78.57 ± 14.47
Heart	80.74 ± 4.94	78.15 ± 10.25	87.78 ± 6.77	54.81 ± 16.82	78.43 ± 6.26	73.59 ± 9.57
Hepatitis	80.67 ± 8.67	81.46 ± 11.89	80.17 ± 10.23	72.38 ± 13.01	68.25 ± 11.63	73.46 ± 8.21
Image Segmentation	70.48 ± 10.95	72.33 ± 9.34	72.98 ± 9.54	23.33 ± 12.18	88.82 ± 8.04	83.74 ± 7.72
Ionosphere	68.0 ± 11.09	66.0 ± 7.31	64.57 ± 8.85	62.29 ± 15.58	89.98 ± 5.25	90.29 ± 6.90
Iris	95.33 ± 4.50	94.67 ± 6.89	96.0 ± 4.66	56 ± 24.59	94.0 ± 6.63	94.76 ± 5.26
Mammographic—Mass	78.25 ± 3.48	79.40 ± 3.64	82.33 ± 3.56	64.82 ± 13.44	82.44 ± 4.56	83.52 ± 4.52
Pima Indian Diabetes	74.63 ± 6.65	72.56 ± 4.46	70.88 ± 5.05	65.93 ± 3.59	72.11 ± 6.96	77.96 ± 7.47
SPECT (Heart)	75.38 ± 5.50	83.56 ± 8.34	78.68 ± 5.17	63.8 ± 26.35	80.03 ± 51.85	79.29 ± 19.43
TAE	50.67 ± 6.11	53.58 ± 7.33	53.04 ± 10.67	32.29 ± 11.01	51.33± 9.45	44.67 ± 10.35
Tic-tac-toe	74.95 ± 4.26	72.54 ± 5.98	72.02 ± 4.50	51.46 ± 14.60	94.03 ± 2.44	97.57 ± 1.44
Transfusion	77.30 ± 6.37	79.44 ± 3.64	77.56± 3.31	53.43 ± 28.77	77.71 ± 12.32	76.11 ± 14.54
Vehicle	56.79 ± 9.56	53.55 ± 11.07	56.0 ± 10.24	25.11 ± 11.45	65.86 ± 6.76	66.52 ± 11.86
WDBC	85.26± 4.22	86.66 ± 4.56	88.05 ± 5.40	66.09 ± 14.59	93.31 ± 2.72	94.03 ± 3.72
Wine	90.0 ± 9.22	90.49 ± 10.13	94.44 ± 5.24	49.48 ± 19.31	96.60 ± 3.93	94.90 ± 5.54
Zoo	81.36 ± 11.30	79.36 ± 15.51	75.27 ± 11.69	32.64 ± 16.23	94.0 ± 9.17	90.0 ± 11.55

TABLE III (c)
AVERAGE PREDICTIVE ACCURACIES OBTAINED USING TENFOLD CROSS-VALIDATION FOR THE ALGORITHMS THAT DO NOT PRODUCE RULE SET MODELS

Datasets	AdaBoost	kNN	Logit	NaiveBayes	SVM
Balance-scale	72.33 ± 2.39	89.26 ± 1.59	88.30 ± 2.69	91.04 ± 2.55	87.98 ± 1.80
BC-W	94.83 ± 1.67	96.42 ± 1.54	96.56 ± 1.21	96.13 ± 1.19	96.70 ± 0.69
Car	69.79 ± 0.48	93.75 ± 1.87	93.22 ± 2.10	86.04 ± 2.32	93.74 ± 2.65
Congress House Votes	96.08 ± 3.27	92.28 ± 3.86	95.14 ± 3.0	90.54 ± 3.59	96.09 ± 3.06
Credit (Aus)	85.76 ± 3.40	82.44 ± 7.31	85.77 ± 4.75	79.37 ± 4.57	85.17 ± 2.06
Credit (Ger)	76.12 ± 2.01	74.43 ± 7.87	75.82 ± 4.24	74.87 ± 5.96	75.11 ± 3.63
Dermatology	96.45 ± 2.88	95.62 ± 3.22	96.71 ± 2.83	96.98 ± 2.75	97.54 ± 2.99
Ecolli	81.53 ± 7.25	80.86 ± 6.45	86.31 ± 5.38	85.66 ± 2.83	82.69 ± 5.04
Glass	44.78 ± 1.89	70.95 ± 5.83	63.65 ± 6.72	51.69 ± 8.31	57.70 ± 8.10
Haberman	69.76 ± 7.21	70.11 ± 11.95	73.08 ± 3.99	74.05 ± 4.76	73.44 ± 0.97
Hayes Roth	83.30 ± 10.46	70.93 ± 9.59	53.52 ± 11.18	77.20 ± 11.08	58.13 ± 15.33
Heart	82.33 ± 6.08	80.71 ± 6.17	77.0 ± 5.05	85.19 ± 8.27	80.32 ± 6.25
Hepatitis	74.71 ± 15.02	67.62 ± 9.06	64.25 ± 8.87	74.79 ± 6.90	75.37 ± 8.62
Image Segmentation	88.02 ± 6.15	86.57 ± 7.99	85.71 ± 9.28	78.48 ± 5.29	88.52 ± 5.70
Ionosphere	90.0 ± 3.37	86.0 ± 6.38	86.57 ± 7.63	82.75 ± 4.94	87.14 ± 5.75
Iris	95.33 ± 3.22	95.33 ± 4.50	97.33 ± 5.62	95.33 ± 3.22	96.67 ± 3.52
Mammographic—Mass	81.88 ± 4.54	78.07 ± 5.99	82.92 ± 3.91	82.69 ± 3.82	80.32 ± 4.37
Pima Indian Diabetes	74.33 ± 6.49	69.75 ± 7.13	77.98 ± 5.91	75.49 ± 4.50	77.06 ± 4.09
SPECT (Heart)	84.52 ± 15.29	81.20 ± 16.77	81.94 ± 15.92	76.87 ± 12.72	87.20 ± 11.48
TAE	41.34 ± 6.13	64.67 ± 7.73	53.33 ± 11.33	57.33 ± 10.91	58.67 ± 10.98
Tic-tac-toe	98.23 ± 0.50	98.75 ± 0.66	98.23 ± 0.50	70.09 ± 5.78	98.33 ± 0.53
Transfusion	75.84 ± 16.0	71.01 ± 14.98	77.17 ± 14.49	72.37 ± 17.22	76.24 ± 15.33
Vehicle	69.36 ± 8.96	67.62 ± 5.63	77.18 ± 9.35	42.79 ± 2.54	66.57 ± 7.40
WDBC	94.71 ± 2.89	95.06 ± 2.88	95.24 ± 2.07	93.48 ± 4.49	97.72 ± 1.45
Wine	92.01 ± 4.93	96.08 ± 4.59	96.60 ± 4.03	98.30 ± 2.74	98.30 ± 2.74
Zoo	91.0 ± 9.94	97.0 ± 6.75	97.0 ± 4.83	97.0 ± 6.75	95.0 ± 7.07

rank, i.e., control algorithm) and Holm critical value obtained by Holm's post-hoc test are reported. If the p-value of an algorithm is lower than the critical value (at 5% significance level), the values in a row are shown in bold. This indicates that there is a significant difference between the average ranks of the compared algorithm and the control algorithm, and that the control algorithm has significantly outperformed the counterpart algorithm in that row.

According to Table IV, AntMiner-CC is the best performing algorithm in terms of predictive accuracy with the lowest average rank of 3.2115. The two variants of AntMiner-CC and SVM can be placed in the same group as AntMiner-CC and these three algorithms can be considered to perform as well as AntMiner-CC. However, we note that AntMiner-CC and its variants develop comprehensible classification models which can be validated, understood, and verified by the end users, as compared to the black box classification model generated by SVM.

Martens *et al.* [18] reported results for AntMiner+ that have used datasets subjected to chi-square attribute selection. This is a preprocessing step and not a part of the AntMiner+ algorithm. In order to make fair comparison between all the algorithms, we have not used chi-square attribute selection for AntMiner+ in our experiment. We have also not used validation set technique to stop the classifier learning process, as is done in [18], for the same reason.

The results of AntMiner-CC are better than those of MAntMiner-CC. The reason for this reduction in accuracy is that in an ordered rule set with more rules (as is the case for MAntMiner-CC), there are more chances of a sample being falsely covered by a rule placed higher in the hierarchy. Furthermore, the reason for UAntMiner-CC having higher number of incorrect classifications is more rules in the classifier combined with the need of using a conflict resolution strategy. However, while comparing these three variants, the advantage of better comprehensibility has also to be taken into consideration.

B. Roles of Pheromones and the Heuristic Function

In this section, we report the results of our investigation of the respective roles played by the pheromone values and the heuristic function. For this purpose, we conducted two experiments. In the first experiment, AntMiner-CC and its variants are modified such that the ants perform a search in the absence of the heuristic function and are guided only by pheromone values. In the second experiment, AntMiner-CC and its variants are modified such that pheromone values are not used and only heuristic function is used.

The results of these experiments are shown in Table V for AntMiner-CC. We observe that accuracy of AntMiner-CC is generally better, even though not prominently, when both pheromone and heuristic values are used together as compared to the cases when only one of them is being used. However, it is the speed of convergence, manifested by the number of ants used, which gives us an insight into the benefit of using both pheromone and heuristic values. AntMiner-CC uses considerably fewer ants than the user-defined number of 1000. This is due to the early exit from the REPEAT-UNTIL loop

TABLE IV

SUMMARY OF THE COMPARISONS OF THE ALGORITHMS BASED ON PREDICTIVE ACCURACY, ACCORDING TO THE NONPARAMETRIC FRIEDMAN TEST WITH THE HOLM'S POST-HOC TEST ON 5% SIGNIFICANCE LEVEL

Comparison basis	Algorithm	Avg. rank	*p*	*Holm*
Predictive accuracy	AntMiner-CC (control)	3.2115	–	–
	UAntMiner-CC	4.5769	0.2392	0.0500
	MAntMiner-CC	4.9231	0.1401	0.0250
	SVM	5.4231	0.0566	0.0167
	Logit	**6.1154**	**0.0123**	**0.0125**
	Knn	**7.4038**	**3.0230E–4**	**0.0100**
	NaiveBayes	**7.5000**	**2.188E–4**	**0.0083**
	AdaBoost	**7.5769**	**1.6823E–4**	**0.0071**
	C4.5	**7.6538**	**1.2877E–4**	**0.0063**
	Ripper	**8.0192**	**3.4173E–5**	**0.0056**
	AntMiner3	**9.0769**	**4.2965E–7**	**0.0050**
	AntMiner2	**9.6923**	**2.3273E–8**	**0.0045**
	AntMiner	**9.9808**	**5.4002E–9**	**0.0042**
	AntMiner+	**13.8462**	**4.9141E–20**	**0.0038**

TABLE V

AVERAGE PREDICTIVE ACCURACIES AND NUMBER OF ANTS FOR ANTMINER-CC AND ITS MODIFIED FORMS, USING TENFOLD CROSS-VALIDATION

Datasets	Max. Ants	AntMiner-CC			Pheromone Only (No Heuristic)			Heuristic Only (No Pheromone)		
		Accuracy	Ants Used	Pr. Eqs	Accuracy	Ants Used	Pr. Eqs	Accuracy	Ants Used	Pr. Eqs
Balance-scale	1000	91.84 ± 1.44	155.48 ± 21.43	2865.23 ± 57.75	90.24 ± 1.47	313.3 ± 17.05	3227.94 ± 52.34	91.03 ± 1.46	991.27 ± 1.65	2789.79 ± 49.59
BC-W	1000	97.71 ± 0.57	77.54 ± 2.41	1238.14 ± 39.52	97.57 ± 0.57	262.19 ± 5.75	1969.88 ± 49.19	97.71 ± 0.57	1000 ± 0	1129.19 ± 47.37
Car	1000	93.06 ± 0.52	240.34 ± 12.99	7088.67 ± 91.96	91.78 ± 0.72	547.03 ± 2.4	6875.35 ± 97.72	92.59 ± 0.65	998.33 ± 0.9	6498.83 ± 85.44
Cong. H. Votes	1000	95.65 ± 1.09	254.27 ± 42.4	1222.07 ± 90.92	95.19 ± 1.04	623.88 ± 32.31	737.09 ± 81.23	95.41 ± 0.9	880.97 ± 21.68	1121.5 ± 67.48
Credit (Aus)	1000	90.43 ± 0.78	306.77 ± 20.62	5624.62 ± 347.55	81.01 ± 2	580.84 ± 10.28	3545.54 ± 621.03	89.71 ± 1.25	950.27 ± 3.5	3677.23 ± 311.11
Credit (Ger)	1000	86.59 ± 1.03	404.43 ± 8.3	22079.47 ± 309.8	83.18 ± 0.99	748.72 ± 16.85	19454.78 ± 413.64	86.48 ± 0.84	1000 ± 0	16476.82 ± 227.91
Dermatology	1000	94.26 ± 1.26	136.72 ± 6.15	3476.24 ± 224.8	95.38 ± 1.07	447.1 ± 11.47	6087.29 ± 120.43	94.23 ± 1.2	1000 ± 0	3107.54 ± 127.76
Ecolli	1000	86.58 ± 1.75	91.46 ± 8.86	2988.62 ± 110.05	85.65 ± 2.83	257.6 ± 4.07	4584.21 ± 113.68	86.56 ± 1.66	963.99 ± 3.6	2505.26 ± 60.98
Glass	1000	72.45 ± 3.6	104.77 ± 6.27	2325.92 ± 39.2	71.9 ± 2.19	310.48 ± 9.37	3860.94 ± 96.65	77.55 ± 2.64	898.83 ± 7.47	1934.5 ± 42.42
Haberman	1000	79.53 ± 2.17	186.46 ± 18.39	1481.29 ± 36.31	75.91 ± 2.46	341.45 ± 9.99	2471.12 ± 56.84	79.2 ± 2.23	960.6 ± 2.25	1405.82 ± 36.17
Hayes Roth	1000	88.68 ± 2.33	147.61 ± 22.02	540.83 ± 19.69	84.07 ± 3.53	228.05 ± 13.45	779.64 ± 25.17	85.6 ± 2.91	907.54 ± 12.32	526.11 ± 12.99
Heart	1000	90.74 ± 1.93	118.43 ± 11.89	1824.32 ± 101.91	88.89 ± 2.7	414.58 ± 14.8	2033.38 ± 149.36	91.11 ± 1.93	1000 ± 0	1480.4 ± 32.85
Hepatitis	1000	94.21 ± 1.51	108.08 ± 4.87	1038.6 ± 43.84	90.96 ± 2.44	369.47 ± 15.68	1567.04 ± 74.9	91.67 ± 2.38	987.68 ± 12.33	824.17 ± 28.68
Image Seg.	1000	89.52 ± 2.11	114.41 ± 3.88	3801.66 ± 186.53	87.62 ± 1.27	302.94 ± 4.25	8247.38 ± 322.4	85.71 ± 2.35	1000 ± 0	2974.7 ± 139.34
Ionosphere	1000	94.00 ± 0.99	136.29 ± 4.37	3540.64 ± 93.46	94.86 ± 1.33	704.84 ± 16.53	6909.48 ± 196.04	94.29 ± 0.95	991.41 ± 6.26	3388.63 ± 73.53
Iris	1000	98.67 ± 0.89	35.54 ± 0.59	140.47 ± 7.06	97.33 ± 1.09	107.81 ± 9.9	310.8 ± 12.04	96.67 ± 1.11	978.02 ± 14.65	143.1 ± 8.42
Mammographic—Mass	1000	68.27 ± 1.63	680.34 ± 23.29	2549.88 ± 74.47	67.22 ± 1.52	539.08 ± 11.49	2277.04 ± 86.09	68.37 ± 1.64	954.76 ± 2.74	2162.84 ± 65.41
Pima Indian D.	1000	87.90 ± 0.91	164.75 ± 10.99	7562.9 ± 103.79	80.11 ± 1.5	481.39 ± 6.96	9435.3 ± 160.86	86.34 ± 1	987.53 ± 2.09	6626.22 ± 65.33
SPECT (Heart)	1000	84.99 ± 2.58	206.18 ± 31.96	4413.39 ± 171.21	85.77 ± 2.47	964.76 ± 7.71	2363.66 ± 88.01	86.52 ± 2.77	633.42 ± 29.66	3854.69 ± 211.68
TAE	1000	84.75 ± 4.23	182.93 ± 49.78	1760.7 ± 82.21	82.75 ± 3.34	220.75 ± 6.17	2536.06 ± 41.46	81.46 ± 4.42	909.5 ± 8.85	1425.88 ± 41.2
Tic-tac-toe	1000	98.22 ± 0.58	286.04 ± 16.76	1163.69 ± 17.15	97.81 ± 0.43	663.01 ± 17.16	1209.55 ± 36.02	98.54 ± 0.39	1000 ± 0	1187.18 ± 16.74
Transfusion	1000	47.66 ± 2.31	310.75 ± 27.04	1435.96 ± 42.45	46.86 ± 2.44	630.07 ± 18.52	1747.69 ± 37.33	47.66 ± 2.31	892.12 ± 10	1385.87 ± 49.7
Vehicle	1000	82.34 ± 2.39	132.47 ± 3.15	6343.39 ± 126.97	80.53 ± 2.8	400.85 ± 7.99	11641.08 ± 122.69	81.92 ± 1.21	1000 ± 0	5092.28 ± 79.45
WDBC	1000	95.61 ± 0.65	163.88 ± 8.26	5506.35 ± 139.96	94.56 ± 1.56	627.8 ± 10.67	14037.87 ± 432.15	95.61 ± 0.91	960.28 ± 8.83	4405.29 ± 46.05
Wine	1000	98.86 ± 0.76	54 ± 2.63	182.09 ± 8.7	100 ± 0	225.08 ± 56.32	323.51 ± 8.69	98.86 ± 0.76	1000 ± 0	207.7 ± 4.3
Zoo	1000	95.09 ± 1.64	69.66 ± 11.35	370.57 ± 16.17	95.09 ± 1.64	212.72 ± 17.49	623.65 ± 19.25	95 ± 1.67	1000 ± 0	454.85 ± 10.97

if the saturation of pheromone values occurs (indicated by the last 10 ants discovering the same rule). However, we use a large number, such as 1000, to ensure that sufficient ants are available for searching before the saturation occurs. When the same search is performed in the absence of any heuristic function and is guided only by the pheromone values, the average number of ants used is much higher as compared to AntMiner-CC but still much lower than the maximum value of 1000. The ants utilize their assigned number almost fully for the case when the search is performed based only on the heuristic values.

An important aspect of classifier learning is that it should terminate in reasonable time even for large, real-world datasets. Our heuristic function (17) combined with class selection prior to the rule construction reduces the search space considerably, making AntMiner-CC attractive even for high dimensional datasets. A heuristic value of zero is assigned to all those terms that do not occur together for a given class.

A costly computation in all ACO-based algorithms, which is done repeatedly, is the calculation of probability equation (7). The counter of its utilization gives a direct indication of the amount of search space which an ant has had to consider. Table V also shows average number of probability calls in an iteration of the main WHILE loop. The best values are for AntMiner-CC using heuristic values only. This is understandable because the exploration of search space is less in the absence of pheromone values. However, the main

TABLE VI
AVERAGE PREDICTIVE ACCURACIES OBTAINED USING TENFOLD CROSS-VALIDATION WITH VARIOUS DESIGN OPTIONS

Datasets	AM-CC	HF: Density Estimation	Condition of Minimum Rule Coverage	Symmetric	Pruning All Generated Rules	Pruning None of the Rules
Balance-scale	91.84 ± 4.55	89.21 ± 2.38	82.01 ± 8.39	89.74 ± 3.11	91.90 ± 4.39	88.89 ± 4.15
BC-W	97.71 ± 1.80	95.43 ± 1.59	94.59 ± 4.25	97.52 ± 2.03	97.71 ± 1.80	97.15 ± 1.89
Congress	95.65 ± 3.45	94.62 ± 2.59	83.88± 8.75	95.57 ± 3.19	95.23 ± 3.56	92.72 ± 4.6
Dermatology	94.26 ± 3.98	93.71 ± 2.24	91.76 ± 5.65	92.71 ± 4.93	95.21 ± 3.45	91.51 ± 3.51
Ecoli	86.58 ± 5.53	82.98 ± 6.71	75.31 ± 7.58	87.86 ± 6.28	86.58 ± 5.53	86.58 ± 5.53
Glass	72.45 ± 11.38	71.08 ± 9.32	64.77 ± 9.88	74.72 ± 6.48	72.45 ± 11.38	77.32 ± 9.15
Haberman	79.53 ± 6.86	77.53 ± 6.09	77.92 ± 8.35	72.49 ± 6.33	80.69 ± 6.37	75.08 ± 7.14
Hayes Roth	88.68 ± 7.37	86.17 ± 7.25	72.98 ± 11.55	86.42 ± 9.77	88.68 ± 7.37	84.93 ± 6.67
Heart	90.74 ± 6.10	87.18 ± 4.94	87.09 ± 4.26	89.32 ± 4.82	90.74 ± 6.10	85.98 ± 6.73
Hepatitis	94.21 ± 4.78	87.33 ± 6.23	86.83 ± 9.12	95.78 ± 5.26	93.59 ± 4.97	87.13 ± 6.29
Ionosphere	94 ± 3.13	90.95 ± 4.59	80.63 ± 9.79	95.89 ± 3.87	94 ± 3.13	94 ± 3.13
Tic-tac-toe	98.22 ± 1.83	97.26 ± 2.59	91.33± 8.15	96.47 ± 2.28	98.22 ± 1.83	97.47 ± 1.77

observation is that the results lend support to the belief of reduction of search space when our heuristic function is used.

The same trends for accuracies, average ants, and average probability equation usage are observed for MAntMiner-CC and UAntMiner-CC. For brevity the results for these algorithms are not shown.

C. Experimental Validation of Algorithmic Design Choices

Several design choices had to be made while developing AntMiner-CC and its variants. All of these have been discussed in Section III. We have also validated our choices by experiments. Some of the experiments are discussed below. The results are presented in Table VI. For brevity, we present results of only a representative subset suite of datasets and only for AntMiner-CC. The observed trends and conclusions are valid for the larger dataset suite and for the two variants of AntMiner-CC. All algorithmic parameters used are shown in Table II.

1) *Effectiveness of Correlation Based Heuristic Function:* In this experiment, we replace our heuristic function with the one used in AntMiner+ (12). The heuristic functions of AntMiner, AntMiner2, and AntMiner3 (9)–(11) cannot be used in AntMiner-CC and its variants, because they do not make use of preselected class label. If we modify these heuristic functions to incorporate the selected class label, then they become similar to (12). The obtained results (Table VI) indicate that the better performance of our algorithms is indeed dependent on our correlation-based heuristic function and degrades if it is replaced by the density estimation heuristic of (12).

2) *Number of Discovered Rules and Terms per Rule:* Apart from predictive accuracy, two other commonly used performance metrics for rule discovery algorithms are the number of rules in a discovered rule set and the average number of terms/rule. If the accuracy of two discovered rule sets is the same, then we can judge them on the other two metrics. Shorter rules and rule sets are better as they are more comprehensible. AntMiner-CC and its variants are at a disadvantage on these accounts. Sometimes the discovered rules are several times more than the compared algorithms. This is due to the absence of minimum sample coverage restriction on discovered rules in our algorithms. In the worst case, a constructed rule sometimes covers only one sample. This disadvantage of our algorithms is compensated by their better predictive accuracy results.

We have conducted an experiment in which AntMiner-CC and its variants are used with the incorporation of a restriction that a rule being created should cover at least a minimum number of samples. We set this threshold as 10, a value used in [14] for the same purpose. A term is added to a rule only if the rule still covers at least 10 samples after its addition. The predictive accuracies for all the datasets decreased when this restriction was imposed (Table VI). The reason is that this condition may prohibit a very effective term from being added, which would have yielded a high confidence rule. Another problem with this approach is that it is difficult to determine a universal correct value for the minimum coverage parameter which is appropriate for all datasets. It may also be noted that our heuristic function (17) already encourages the inclusion of term that has better coverage among the competing terms.

3) *Symmetric or Asymmetric Pheromone Matrix:* Our pheromone matrix is asymmetric [Fig. 5(b)]. If an ant chooses $term_j$ after $term_i$, then the pheromone on the edge between $term_i$ and $term_j$ is updated but the pheromone value on the edge between $term_j$ and $term_i$ is not modified. This is done to encourage exploration. We have conducted experiments in which pheromone values on both the edges between two chosen consecutive terms are updated. This pheromone update method yields a symmetric pheromone matrix. Experimental results, shown in Table VI, indicate a decrease in accuracy when symmetric update is used. The ants have a lesser chance for exploration and converge quickly to a solution and it may be an inferior one.

4) *Rule Pruning: All Rules, No Rules, or Best Rule?:* In AntMiner-CC and its variants we prune only the best rule found during an execution of the REPEAT-UNTIL loop before inserting it in the final rule set. We have done two experiments in this regard. In the first experiment we prune all the rules discovered during the REPEAT-UNTIL loop (prior to pheromone update) and in the second experiment none of the rules is pruned.

Experimental results are shown in Table VI. From these results we note that rule pruning almost always has a positive effect on the accuracy. It may be noted that we use the confidence and coverage as the rule's quality measure (23) for pruning purpose. Sometimes this equation sacrifices a term contributing to the higher confidence of a rule because its removal increases the coverage of the rule. Due to this reason we sometimes observe a decrease in accuracy of the classifier when rule pruning is used. However, these occurrences are rare. Furthermore, we note that pruning only the best rule and pruning all the rules give almost similar results. Since pruning is a costly procedure, we retain pruning of only the best rule in our algorithm.

V. Conclusion and Future Work

In this paper, we presented and analyzed three variants of a novel ACO-based classification algorithm. Its main feature is a new heuristic function that takes into account the relationship between the previously selected term and the candidate term, given a preselected class label. The experimental results indicated that it has potential to compete with other contemporary classification algorithms. The main advantage of our approach was high accuracy combined with comprehensibility of the discovered rule set.

As an extension of the main idea, we are currently working on a new heuristic function that takes into account the relationship between the last two selected terms and the candidate term. We also intend to investigate the extension of this idea to last n selected terms. In this regard, the Apriori algorithm [11] used in association rule mining for finding frequent item sets may prove to be helpful. The Apriori property prunes a large number of candidate combinations if their subset combinations are not frequent (or do not occur at all). Instead of specifying a minimum support threshold, we can use the frequency with which terms occur together as a heuristic guide. Terms that do not occur together at all, and their superset combinations, will be automatically pruned from the search space.

Work on the same lines can be done for the problem of hierarchical multilabel classification [36]. In such problems, samples are associated with more than one class label at a time and the class labels are organized in a hierarchical structure. Our heuristic function would be useful in capturing the relationships between terms and classes, and the Aprori property may be applied to reduce the search space.

Another idea worth investigating is that of using domain knowledge in the proposed algorithm. One simple way to improve the discovered rule set with the help of domain knowledge is by looking for missing, redundant, and misleading rules. The second way is to incorporate hard constraints of the domain knowledge by modifying the search space and soft constraints by influencing the heuristic values as is done in [37] for AntMiner+.

Finally, the algorithm can be altered to accommodate multiobjective rule mining. Alatas and Akin [38] experiment with the biobjectives of accuracy and comprehensibility in the framework of a new PSO algorithm. The idea can also be applied in the context of ACO.

References

[1] A. P. Engelbrecht, *Computational Intelligence, An Introduction*, 2nd ed. New York: Wiley, 2007.

[2] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. New York: Wiley, 2005.

[3] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann/Academic, 2001.

[4] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing* (Natural Computing Series, 2nd ed.). Berlin, Germany: Springer, 2007.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[7] T. Stutzle and H. H. Hoos, "MAX–MIN ant system," *Future Generation Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[8] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[9] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization: Artificial ants as a computational intelligence technique," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[10] A. Abraham, C. Grosan, and V. Ramos, *Swarm Intelligence in Data Mining* (Studies in Computational Intelligence, vol. 34). Berlin, Germany: Springer, 2006.

[11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2006.

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

[13] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2000.

[14] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 321–332, Aug. 2002.

[15] B. Liu, H. A. Abbass, and B. McKay, "Density-based heuristic for rule discovery with ant-miner," in *Proc. 6th Australasia-Japan Joint Workshop Intell. Evol. Syst.*, 2002, pp. 180–184.

[16] B. Liu, H. A. Abbass, and B. McKay, "Classification rule discovery with ant colony optimization," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.*, Oct. 2003, pp. 83–88.

[17] B. Liu, H. A. Abbass, and B. McKay, "Classification rule discovery with ant colony optimization," *IEEE Comput. Intell. Bull.*, vol. 3, no. 1, pp. 31–35, Feb. 2004.

[18] D. Martens, M. de Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 651–665, Oct. 2007.

[19] J. Smaldon and A. A. Freitas, "A new version of the ant-miner algorithm discovering unordered rule sets," in *Proc. GECCO*, 2006, pp. 43–50.

[20] N. Holden and A. A. Freitas, "A hybrid PSO/ACO algorithm for classification," in *Proc. GECCO Workshop Particle Swarms: The Second Decade*, 2007, pp. 2745–2750.

[21] S. Swaminathan, "Rule induction using ant colony optimization for mixed variable attributes," M.Sc. thesis, Texas Tech. Univ., Lubbock, 2006.

[22] F. Otero, A. Freitas, and C. Johnson, "cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes," in *Proc. ANTS*, LNCS 5217. 2008, pp. 48–59.

[23] A. Chan and A. Freitas, "A new ant colony algorithm for multilabel classification with applications in bioinformatics," in *Proc. GECCO*, 2006, pp. 27–34.

[24] U. Boryczka and J. Kozak, "New algorithms for generation decision trees: Ant-miner and its modifications," in *Foundations of Computational Intelligence, vol. 6* (Studies in Computational Intelligences, vol. 206). Berlin, Germany: Springer, 2009.

[25] D. Martens, B. Baesens, and T. Fawcett, "Editorial survey: Swarm intelligence for data mining," *Mach. Learning*, vol. 82, no. 1, pp. 1–42, 2011.

[26] A. R. Baig and W. Shahzad, "A correlation based Ant Miner for classification rule discovery," *Neural Comput. Appl. J.*, vol. 21, no. 2, pp. 219–235, Mar. 2012.

[27] W. Shahzad and A. R. Baig, "Compatibility as a heuristic for construction of rules by artificial ants," *J. Circuits Syst. Comput.*, vol. 19, no. 1, pp. 297–306, Feb. 2010.

[28] S. Hettich and S. D. Bay. (1996). *The UCI KDD Archive* [Online]. Available: http://kdd.ics.uci.edu

[29] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[30] J. R. Quinlan, "Generating production rules from decision trees," in *Proc. Int. Joint Conf. Artif. Intell.*, San Francisco, CA, USA, 1987, pp. 304–307.
[31] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[32] F. Meyer and R. S. Parpinelli. (2009). *GUI Ant-Miner* [Online]. Available: http://sourceforge.net/projects/guiantminer/
[33] D. Martens. (2012). *Classification With Ant Colony Optimization* [Online]. Available: http://www.antminerplus.com/
[34] S. Garcia and F. Herrera, "An extension on statistical comparisons of classifiers over multiple datasets for all pairwise comparisons," *J. Mach. Learning Res.*, vol. 9, pp. 2677–2694, Dec. 2008.
[35] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learning Res.*, vol. 7, pp. 1–30, Jan. 2006.
[36] S. N. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining Knowledge Discovery*, vol. 22, nos. 1–2, pp. 31–72, 2011.
[37] D. Martens, M. de Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen, "Ant-based approach to the knowledge fusion problem," in *Proc. ANTS*, LNCS 4150. 2006, pp. 84–95.
[38] B. Alatas and E. Akin, "Multiobjective rule mining using a chaotic particle swarm optimization algorithm," *Knowledge Based Syst.*, vol. 22, no. 6, pp. 455–460, 2009.

Abdul Rauf Baig (M'07) received the B.E. degree in electrical engineering from the NED University of Engineering and Technology, Karachi, Pakistan, in 1987, the Diplôme de Spécialisation degree in computer science from Supélec, Rennes, France, in 1996, and the Ph.D. degree in computer science from the University of Rennes-I, Rennes, France, in 2000.

He was with the National University of Computer and Emerging Sciences, Islamabad, Pakistan, as a Faculty Member (Assistant Professor, Associate Professor, and then Professor) from 2001 to 2010, and is currently on leave. Since 2010, he has been a Professor with Imam Muhammad bin Saud Islamic University, Riyadh, Saudi Arabia. He has more than 90 publications in journals and international conferences. His current research interests include data mining and evolutionary algorithms.

Dr. Baig is a member of the IEEE Computational Intelligence Society.

Waseem Shahzad received the M.S. and Ph.D. degrees in computer science from the National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2007 and 2010, respectively.

Since 2010, he has been an Assistant Professor with the National University of Computer and Emerging Sciences. His current research interests include data mining, computational intelligence, machine learning, theory of computation, and soft computing. He has several publications to his credit.

Salabat Khan received the B.S. degree in computer science from Virtual University, Lahore, Pakistan, in 2007, and the M.S. degree in computer science from the National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2009. He is currently pursuing the Ph.D. degree in computer science with the National University of Computer and Emerging Sciences, Islamabad.

His current research interests include data mining, pattern analysis, medical image processing, bioinformatics, and bio-inspired algorithms.