Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 2232000, 17 pages
https://doi.org/10.1155/2022/2232000

Review Article

Rule-Based Classification Based on Ant Colony Optimization: A Comprehensive Review

Sayed Kaes Maruf Hossain, Sajia Afrin Ema, and Hansuk Sohn

Department of Industrial Engineering, New Mexico State University, Las Cruces, NM 88003, USA

Correspondence should be addressed to Hansuk Sohn; hsohn@nmsu.edu

Received 19 January 2022; Revised 8 March 2022; Accepted 25 March 2022; Published 8 April 2022

Academic Editor: Manikandan Ramachandran

The Ant Colony Optimization (ACO) algorithms have been well-studied by the Operations Research community for solving combinatorial optimization problems. A handful of researchers in the Data Science community have successfully implemented various ACO methodologies for rule-based classification. This family of ACO algorithms is referred to as AntMiner algorithms. Due to the flexibility of the framework, and the availability of alternative strategies at the modular level, a systematic review on the AntMiner algorithms can benefit the broader community of researchers and practitioners interested in highly interpretable classification techniques. In this paper, we provided a comprehensive review of each module of the AntMiner algorithms. Our motivation is to provide insight into the current practices and future research scope in the context of the rule-based classification. Our discussions address ACO methodologies, rule construction strategies, candidate selection metrics, rule quality evaluation functions, rule pruning strategies, methods to address continuous attributes, parameter selection, and experimental settings. This review also reports a summary of real-life implementations of the rule-based classifiers in diverse domains including medical, genetics, portfolio analysis, geographic information system (GIS), human-machine interaction (HMI), autonomous driving, ICT, quality, and reliability engineering. These implementations demonstrate the potential application domains that can be benefitted from the methodological contributions to the rule-based classification technique.

1. Introduction

The rule-based classification method appeals to the data mining community due to the high interpretability of the classifier. While widely used classification techniques such as Support Vector Machines and Artificial Neural Network are admired for their robustness and accuracy, they are often criticized for the lack of interpretability. In contrast, the decision rules in a rule-based classifier are readily interpretable to humans. Naturally, the rule-based classification is popular in the application areas where the interpretability of the classifier to domain experts is deemed crucial such as cancer research, genetics, and financial analytics [1, 2].

A rule-based classifier consists of a set of "IF-THEN" rules obtained by statistically apprehending the training data. Each rule of the classifier consists of an antecedent and a consequent. The antecedent part contains one or more terms, where each term is comprised of a variable name, an operator, and a value. In the cases where an antecedent contains more than one term, the terms are joined by the "AND" conjunction. On the other hand, the consequent part of the rule represents the class label associated with the rule. The key components of a generic classification rule are shown in Figure 1.

The process of exploring such rules requires a decision on which attributes and corresponding values to consider for classification. This is aligned with the idea of combinatorial optimization, in the sense that the goal is to find a set of attributes and corresponding instance values connected by conjunctions, which maximizes the objective measures and accuracy. If the dataset under consideration is large, the computational effort for extracting such a set of rules can be considerably expensive. Given this context, using intelligent heuristic search algorithms can be helpful in reducing the

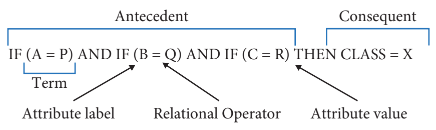

Figure 1: The structure of a generic classification rule. The example demonstrates the key components of a classification rule. The antecedent contains one or more attribute-value pairs connected by AND conjunction. The suggested class is shown in the consequent on the right side of the THEN clause.

computational effort. It is important to note that such algorithms will not guarantee finding the best classifier, but one of the top-performing ones. Thus, it is critical to implement the algorithm with effective strategies.

Ant Colony Optimization (ACO) algorithms are a family of heuristic optimization algorithms inspired by the food foraging behavior of ants in the natural system. ACO algorithms have been evolving over the last few decades in terms of search strategies, parameter settings, and new application areas. Various combinatorial optimization problems in the area of Operations Research including the Traveling Salesman Problem [3], Vehicle Routing Problem [4], Facility Layout Problem [5], and Facility Location Problem [6] have been efficiently solved by ACO algorithms.

A specialized family of ACO algorithms, AntMiner, has successfully been implemented to address the rule-based classification problem. During later years, further extension of AntMiner algorithms was suggested. Freitas et al. provided an overview of AntMiner algorithms until that time and some critical future directions [7]. While this work was an admirable contribution at the time, the narrations are brief and may not include the most recent developments. To the best of our knowledge, their work is the only dedicated review article on this topic. In this article, we aim to extend their work by identifying potential application areas and reporting detailed and updated information on AntMiner algorithms at the modular level. This review covers 83 relevant scientific articles indexed in the Web of Science and/or Google Scholar. We have included articles that (a) provide some methodological contribution to AntMiner algorithms, (b) demonstrate real-life applications of rule-based classifier, and (c) report developments in the ACO domain that are relevant to AntMiner algorithms and/or provide use cases demonstrating successful implementation of ACO. Note that, in the interest of manifesting the potential disciplines that can be benefitted from future research contribution in this area, the listing of applications of rule-based classifiers is not limited to any particular methodology. When reviewing existing methods, we sought information following the framework shown in Table 1.

This review article is organized as follows. Section 1 provides an introduction to the rule-based classification problem, an oversight on the connection of the rule-based classification, and the research framework; Section 2 lists some existing application domains for rule-based classification; Section 3 provides a background of AntMiner algorithm, discusses how concepts in ACO can be transferred in the context of rule-based classification, and provides a high-level description of the AntMiner algorithm; Section 4 provides an extensive survey of existing methods for implementing different modules of AntMiner; Section 5 outlines most common experimental setting in terms of measurement metrics and validation approaches and the datasets used by earlier researchers for benchmarking purpose; Section 6 discusses the research gaps and recommendations for future research; Section 7 provides some concluding remarks and closes our article.

2. Real-Life Applications of Rule-Based Classification

In this section, a list of real-life applications of rule-based classification is presented. The goal is to identify application areas that will potentially be benefiting from the research involving rule-based classification. In Table 2, we included the reported applications regardless of the rule discovery methods used which are categorized into eight primary domains, i.e., medical, genetics, portfolio analysis, geographic information system (GIS), human-machine interaction (HMI), autonomous driving, ICT, and quality and reliability engineering. A summary of these applications is provided in the following table. It is interesting to notice that most applications of rule-based classification reported in the literature are in the medical domain.

3. Overview of the AntMiner Algorithms

3.1. Background. The AntMiner algorithms are descendants of ACO algorithms that mimic the natural food foraging behavior of ants. In the natural system, when ants seek food, they leave a pheromone trail for the successor ants to follow. Once they find a food source, ants return to the nest, while depositing pheromone. Since the deposited pheromone is exposed to the environment, it continuously evaporates while the ants travel. Understandably, the ants taking a shorter distance would return to the nest sooner and the remaining pheromone level on their path would be higher. While the following ant makes a decision to select a path from a set of alternatives, the decision is dictated by the amount of pheromone deposited on the candidate alternatives. The higher the pheromone level on a path, the higher the chance that it will be selected. However, this process is stochastic and a higher level of deposited pheromone on a path does not guarantee that the path would be selected. It rather means that the path would have a higher chance of getting selected over alternate options. As the process continues, more ants would be inclined to take the shorter path and possibly make the pheromone level on that path even stronger [50, 51].

This idea of progressively finding the shortest path is used in the ACO algorithms. To mimic the natural system, artificial ants are carefully programmed for tour construction, pheromone updating, etc. such that the agents collectively work towards finding the shortest path. The original algorithm developed by Dorigo et al. was applied to solve the Traveling Salesman Problem (TSP) [51]. Soon after the first application, the wider combinatorial optimization

Table 1: Framework for reviewing existing methods.

Methodology	Core modules	Classification specific modules
Mapping AntMiner variants to a comparable ACO methodology	Identifying reported strategies for implementing core AntMiner modules 1. Transition probability function 2. Heuristic value function 3. Pheromone updating strategy 4. Evaluation of rule quality 5. Handling continuous attributes	What are different ways to 6. Evaluate rule quality 7. Prune constructed rules 8. Classify an unseen instance 9. How to handle multiclass problems

Table 2: Applications of rule-based classifier.

Domain	Work	Application
Medical	[8]	Detection of epilepsy based on ECG
	[9]	Predicting radiation toxicity in prostate cancer treatment
	[10]	Heart disease prediction
	[11]	Lung nodule detection based on thoracic computed tomography images
	[12]	Lung nodule detection based on thin section CT images
	[13]	Predicting diabetes
	[14]	Classification of premature ventricular complexes
	[15]	Mammography classification
	[16]	EEG signal classification
	[17]	Classification of lung cancer stages from free text pathology reports
	[18]	Analysis of dermatology databases
	[19]	Medical decision support
	[20]	Computer aided diagnosis
	[21]	Classification of Arabic medical texts
	[22]	Classification of diseases in discharge summaries
Genetics	[23]	Analysis of gene-gene and gene-environment interactions in genetic association study
Portfolio analysis	[24, 25]	Bankruptcy prediction
GIS	[26]	Landslide prediction
	[27]	Mapping wetlands
	[28, 29]	Land cover classification
HMI	[30, 31]	Body poses and gesture recognition
	[32]	Handwriting recognition
Autonomous driving	[33]	Classification of lighting conditions for driving scenes
ICT	[34]	User behavior prediction on website
	[35]	Event driven messaging system
	[36]	Sports video indexing
	[37, 38]	Network anomaly detection
	[39]	Image processing
	[40]	Sentiment analysis on microblog
	[41]	Finding people with emotional distress in social media
	[42]	Detecting travel modes
	[43]	Classification of data streams
Quality and reliability	[44]	Fault detection of bearings
	[45]	Fault detection and diagnosis of vapor compression air conditioners
	[46]	Fault diagnosis of rotating machinery
	[47]	Fault detection in semiconductor wafers
	[48]	Analysis of bank service quality
	[49]	Recognition of power quality disturbances

community has implemented ACO algorithms and their variants to solve problems in diverse areas including Quadratic Assignment Problem, Job-Shop Scheduling Problem, Vehicle Routing Problem, Graph Coloring Problem, and Network Routing Problem, to name a few [52]. Over the following years, there have been several versions of the algorithm developed by the ACO practitioners incorporating various search strategies, pheromone updating strategies, heuristic function values, and local updating rules [53, 54].

3.2. An Example of Ant Colony Optimization for TSP. We introduce a short demonstrative example of how the ACO algorithm is used to solve the classical TSP. For further details, the readers may refer to Dorigo, Caro, and Gambardella [52]. In the TSP, our interest is to find the shortest route that a salesman can take to cover all cities in a given set. While the salesman travels, he/she may not travel to the same city more than once but will return to the origin city. The available information is the distance between each pair of cities. For demonstration purpose, let us consider a TSP with four cities: A, B, C, and D, with A as the origin city. The distances between the cities are arbitrarily chosen and are illustrated in Figure 2. It is assumed that there is an artificial ant colony consisting of two ants and they probabilistically choose one edge at a time to construct tour solutions. This probability is a function of heuristic value and pheromone value. The heuristic value in this case would be determined by the distances between each pair of cities. On the other hand, the pheromone level will be increased or decreased over the iterations on each edge, depending on how often an edge is used by the ants to construct a tour solution. In the beginning, the pheromone level on each edge is equally distributed. As the population only consists of two ants, two solutions in the first iteration exist (see Figures 2(a) and 2(b)). By examining these two solutions, it is easy to see that the edges A-B and C-D were used by both ants. Hence, the pheromone levels on these paths are reinforced by a predetermined increment value. To avoid saturation of pheromone, the pheromone levels on all edges are reduced by a certain rate at each iteration. Figure 2(c) shows an analogy representing the value of pheromone on each edge by the levels of darkness and thickness of the edges. Darker and thicker lines would mean relatively higher levels of pheromone on corresponding edges. The edges that possess higher levels of pheromone will have higher probabilities of getting chosen in the following iterations. A common modification of this strategy is to allow the best ant in each iteration to deposit pheromone rather than allowing every ant to do so. Over the iterations, quality solutions get higher probabilities of getting explored and exploited. Depending on a termination criterion (such as the number of iterations or saturation), the algorithm stops and returns the incumbent solution. While there is no guarantee that such solutions will be the optimal solutions, the literature supports the claim that they often provide solutions with satisfactory quality at a reasonable amount of time.

3.3. Bridging ACO and AntMiner. Bennett and Parrado-Hernández methodologically show some interesting interplay between machine learning and optimization in the general sense [55]. In line with the idea, the rule induction process in a rule-based classifier can be modeled as a combinatorial optimization problem. Thus, a customized version of ACO algorithms can be justifiably used for rule discovery. In the context of rule-based classification, each unique attribute-value pair can be considered a virtual node of a graph. However, in this case, a modified constraint will be that the nodes associated with every attribute can appear at maximum once. Also, depending on the model, it may not be required to include all attributes in every rule.

Given this context, Figure 3 shows an illustrative example of how such a rule can be discovered. This is an imaginary dataset replicating the credit default dataset. An ant can construct a rule by visiting a maximum of one value from each attribute. Thus, each node of the graph in this case is represented as a two-dimensional index (labeled by attribute and corresponding value). The particular rule shown as an example path in the figure can be expressed as IF (Age = Group_2) AND IF (Employment = Full-time) AND IF (Credit_Score = Mid-High) THEN Class = Not_Default. In the upcoming sections, we will provide details on the mechanisms of discovering such classifiers.

Parpinelli et al. proposed the first strategy to use an ACO-based algorithm and named it AntMiner. The AntMiner algorithm was tested on publicly available medical domain datasets [1]. In the following years, a number of researchers have proposed and analyzed various strategies to improve the performance and expand the capabilities of the algorithm.

In this paper, to minimize ambiguity, when referring to the original version of AntMiner, we will use the name 'AntMiner'. The phrase 'AntMiner algorithms' will also refer to the family of all rule-based classification algorithms that are developed based on the ACO algorithms. Finally, the term 'AntMiner version' would refer to a certain version of AntMiner algorithms.

3.4. High-Level Description of AntMiner. The AntMiner algorithms start with a preprocessed full training set provided by the user. The user also needs to input some parameters before the initialization. During the first iteration, the algorithm has coverage-related information for every potential term. However, the ultimate impact each term will have on the rule quality can only be evaluated once a complete rule is constructed. The heuristic values and pheromone values are initialized using a predetermined mechanism. Every ant will use a probability function to progressively select strictly one value from each attribute. The order of selection does not affect the rule quality. The best rule constructed by the ant is selected as the iteration best rule. The terms used in this iteration best rule are rewarded by allowing addition of pheromone to the terms used in the iteration best rule. On the other hand, the terms not used in the best rule are penalized by reducing their pheromone level, metaphorically known as pheromone evaporation. As the algorithm progresses, the terms associated with high-quality rules will get a higher pheromone level, leading to a higher probability of selection in the succeeding iterations. If the same rule is suggested over several iterations, the search process is converged. After a predetermined number of iterations are performed or the algorithm has converged, the best rule among all the iteration best rules is selected and added to the discovered rules list. This means the newly added rule to the discovered rules list has become a member of the rules in the classifier. A pseudocode for this process is shown in Figure 4.

Before moving forward, the instances covered by the latest discovered rule are removed from the training set. This

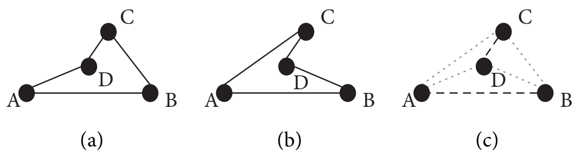

Figure 2: An example of ACO algorithm being used to find the optimal solution of TSP problem. In a population of two ants, (a) and (b) are the paths constructed by the two ants. The choice of an edge is probabilistically done. (c) shows how the pheromone levels on the edges are updated following an iteration.

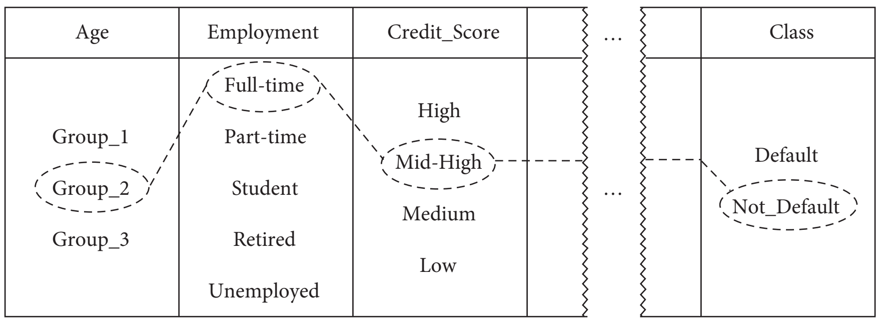

Figure 3: An example of rule construction by ants on the problem graph. The example replicates a credit default setting where the classification rules will suggest whether a given instance should be classified as a potential credit default risk or not.

means the previous heuristic and pheromone-related information will no longer be useful as the number of instances in the training data has changed. Thus, all the ACO parameters are reinitialized, and the algorithm is executed to discover the next rule. Once a new rule is discovered and added to the discovered rules list, the training data is further reduced by removing the covered instances. This process continues until the number of instances in the training data is equal to or less than a user-selected parameter, called maximum uncovered case. Table 3 gives some useful definitions of parameters commonly used in AntMiner algorithms.

4. AntMiner Algorithms–Framework and Modules

4.1. ACO Methodologies Implemented in AntMiner Algorithms. Given the modular features available to customize the well-known ACO algorithms, in some cases, it is difficult to say all the features of certain AntMiner versions are solely inherited from one version of the ACO algorithm. However, in this section, we would refer to an ACO algorithm as the base of an AntMiner version, which shares the most resemblance in terms of implementation.

The original AntMiner algorithm was implemented based on the concept of the Ant System (AS) algorithm. Although in the case of AS algorithm pheromone is updated once all ants in the population have constructed respective complete solutions, in the AntMiner algorithm pheromone is updated each time an ant constructs a solution. This enables the immediate next ant to use the updated pheromone information. The authors have referred to this process as having a population of a single ant. While they have discussed the idea of having a population of more than a single ant in the population, it was not demonstrated in the AntMiner algorithm [56]. The algorithm proposed by Liu et al. is also based on the AS algorithm [57].

Liu et al., on the other hand, implemented an interesting concept of controlling the balance between the magnitude of exploration and exploitation to be used [58]. This idea was originally proposed in the Ant-Q family of algorithms for TSP [59]. The authors claimed that the Ant-Q algorithm strengthened the connection between reinforcement learning and AS. A similar idea was incorporated into the ACS algorithms [3, 60]. In short, a parameter is used to control whether a term will be selected based on AntMiner's probability function or just random selection. As apparent from the definition, the probability function uses information from the previously discovered rules by the means of pheromone value (encouraging exploitation) but the random selection is independent of any influence from previous information (encouraging exploration). In a later work, Liu et al. provided a theoretical demonstration of how their earlier work can provide more diversity in the search process compared to the original AntMiner [61].

The AntMiner + algorithm implements the MAX-MIN ACO algorithm for classification. The main idea is to constrain the minimum and maximum level of pheromone on the discovered path as the ants continue to construct solutions. The AntMiner + also introduced two dynamic parameters in the probability function itself, to control the weight of heuristic value and pheromone level. This provides a mechanism to select the weight of the exploration and exploitation operators as part of the search process [2, 62].

While the remaining AntMiner versions have some other forms of contributions, the ACO algorithms they used are limited to the abovementioned methodologies, in principle.

```
training_set ← training_data
discovered_rules ← []

while   (|training_set| > max_uncovered_cases):
    initialize(user_input)
    calculate_heuristic ()
    best_rule ← []

    while   (iteration <= max_iteration) and (rule_converge == False):
        it_best_rule ← []
        it_best_Q = 0

        while   (ant_id < max_ant):
            rule ← []

            while   (|rule| < max_length) and (min_cases_rule == True):
                rule.append(choice(term_ij))

            class = conseqent ()
            Q = evaluate(rule)

            if(Q > it_best_Q):
                it_best_rule = rule

            ant_id += 1

        it_best_rule = prune (it_best_rule)
        it_best_Q = evaluate (it_best_rule)

        if (it_best_Q > best_Q) :
            best_rule = it_best_rule
            best_Q = it_best_Q

        update_pheromone ()
        update_parameters ()
        iteraton += 1

    discovered_rules. append (best_rule)
    training_set ← training_set\covered_by (best_rule)

return discovered_rules
```

Figure 4: A generic high-level pseudocode for AntMiner algorithms.

Table 3: Definitions of commonly used parameters in AntMiner algorithms.

Parameter	Definition
Maximum uncovered cases (max_uncovered_cases)	The maximum number of cases in the training data that can be left out without being covered by any rule
Minimum cases per rule (min_cases_rule)	The minimum number of cases that must be covered by a rule when a new term is added to the rule
Number of iterations (max_iteration)	The maximum number of ant cycles allowed if other convergence criteria are not met
Number of ants (max_ant)	The maximum number of ants representing how many rules are to be constructed in each ant cycle

4.2. Rule Construction Strategies. The task of rule construction for a rule-based classifier involves traversing through the attributes and values to construct the antecedent and select an appropriate class label. In all of the existing AntMiner algorithms, each ant constructs a rule where exactly one value from each of the attributes is included in the antecedent unless an early termination criterion is applied. Later, using some pruning strategy, the length of the rule is shortened when possible.

In AntMiner, the first ant starts exploring the entire training set. It keeps adding one feasible value from each attribute unless adding that new term decreases the coverage of the rule below a predetermined threshold, min_cases_covered. The following ants construct rules in a similar way until all ants have constructed a rule or a predetermined number of consecutive ants have constructed the same rule. Once all ants finish construction, the best rule in this phase is added to the discovered_rules list. Also, the training instances covered by this rule are removed from the training dataset. The process keeps repeating until the maximum number of uncovered training instances (max_uncovered_cases) is reached. We see a similar rule construction strategy implemented in Liu et al. with some changes as discussed in the following sections [57, 58].

The AntMiner + starts by selecting a class for which rules will be discovered. This provides the advantage of having to calculate heuristic values associated with one class value only. Martens et al. also reformulated a graph that allows ants to deposit pheromone on the edges rather than the vertices. The algorithm uses a population of ants in each ant cycle (iteration) instead of a single ant for constructing rules. The best ant in each ant cycle is allowed to reinforce pheromone while all trails are subject to pheromone evaporation. Only the best rule in each ant cycle is passed on to the pruning procedure. The algorithm also keeps track of the error measure on a validation set to deal with overfitting. If the error measure in the validation set starts increasing the algorithm stops [2].

Baig, Shahzad, and Khan suggested an approach where the ant selects the class label before the main ACO loop begins. The class is chosen probabilistically weighted in proportion to their frequencies in the uncovered training data. They argued in favor of using well-recognized discretization methods for handling continuous attributes at the preprocessing step. They also suggested a heuristic function that takes into account correlation between the class label, last selected term, and candidate terms [63].

The algorithm by Smaldon and Freitas also starts by assigning a consequent before constructing an antecedent of the rule, however, aiming at discovering an unordered set of rules. They used a new heuristic value function and pheromone updating strategy which could potentially be extended to the ordered rule set as well [64].

Salama et al. proposed μAntMiner which uses separate pheromone information for each class. While in this algorithm the consequent is selected before the rule is constructed, the ant may construct rules containing different consequents in the same rule discovery stage like AntMiner+. However, it is different from the view that information on terms with respect to each consequent label is kept independent of each other [65].

4.3. Transition Probability Functions

4.3. Transition Probability Functions. In most AntMiner variants, each attribute-value pair (i.e., term) is represented as a node on the problem graph. A probability function defines the probability of a term to be included in the partial rule that is being constructed. Although there are functional similarities between the transition probability functions used for AntMiner algorithms and general ACO algorithms, there are some differences in the interpretations. In general ACO, P_{ij} refers to the transition probability from node i to j via arc ij whereas in most AntMiner versions it represents the probability of selection of node ij representing the value j of attribute i, regardless of the departing node. A weighted random selection process is used to pick a $term_{ij}$, based on the corresponding P_{ij} value [1, 56].

$$p_{ij} = \frac{\eta_{ij}\tau_{ij}}{\sum_{k=1}^{a}\sum_{l=1}^{b_i} x_i \eta_{kl}\tau_{kl}}. \tag{1}$$

Another transition probability function is used by Liu et al. which is similar to the strategy in ANT-Q and ACS [58]. This function provides an explicit mechanism to keep the exploration process active, regardless of the level of pheromone available on the path. A random number $q[0,1]$ is generated and checked against a threshold value (q_0). If q is less than the threshold, the term is chosen probabilistically using the conventional probability function shown in equation (1). Otherwise, the term with the maximum P_{ij} value is selected deterministically. In the following equation, S indicates a weighted probabilistic choice of $term_{ij}$ using equation (1) for P_{ij}.

$$T_{ij} = \begin{cases} S, & \text{if } (q \leq q_0), \\ \arg\max\left(p_{ij}\right), & \text{otherwise.} \end{cases} \tag{2}$$

Martens et al. redefined the problem graph into a version that is more in line with the conventional ACO models [2], where the pheromone is deposited on the arcs between a pair of vertices. The probability of selecting the edge leading to a vertex v_{ij} is given by the following equation. Also, in this case, the normalization takes place over the values in the next available variable only. This is sufficient because in AntMiner+ the order of variables for an ant to traverse is predetermined. They also included weights (α,β) on the pheromone level and heuristic value, providing a direct way to control the weight of exploration and exploitation in the probability function itself.

$$P_{ij}(t) = \frac{\left[\tau_{v_{i-1,k},v_{i,j}}(t)\right]^{\alpha}\left[\eta_{v_{i,j}}(t)\right]^{\beta}}{\sum_{l=1}^{b_i}\left[\tau_{v_{i-1,k},v_{i,l}}(t)\right]^{\alpha}\left[\eta_{v_{i,l}}(t)\right]^{\beta}}. \tag{3}$$

4.4. Heuristic Value Functions. There are two major approaches for evaluating heuristic values reported in the literature. The first one is an information-theoretic measure of the entropy of a discovered rule, based on the information theory [66]. The original AntMiner algorithm used this method to compute the heuristic value of a discovered rule [56].

$$\eta_{ij} = \frac{\log_2 c - H\left(W|A_i = V_{ij}\right)}{\sum_{k=1}^{a}\sum_{l=1}^{b_k}(x_i)\left(\log_2 c - H\left(W|A_k = V_{kl}\right)\right)}, \tag{4}$$

$$H\left(W|A_i = V_{ij}\right) = -\sum_{w=1}^{c}\left(P\left(w|A_i = V_{ij}\right)\log_2 P\left(w|A_i = V_{ij}\right)\right). \tag{5}$$

Here, $P(w|A_i = V_{ij})$ represents the conditional probability of selecting a class label w given the term $A_i = V_{ij}$ is selected. A higher value of $H(w|A_i = V_{ij})$ means the classes are more uniformly distributed and selecting $A_i = V_{ij}$ will add less value; in turn, it should have less probability of being included in the current partial rule. The value of $H(w|A_i = V_{ij})$ varies in the range of $0 \le H(w|A_i = V_{ij}) \le \text{Log}_2 C$, where C represents the number of class labels in the class attribute W.

The other major approach to measure heuristic value is based on density estimation [57]. While the authors acknowledged that this measure may not be as accurate as the information-theoretic measure, they claimed this compromise is not large and can be potentially compensated by the pheromone updating strategy. Considering the reduced computational effort in the density-based method, the most recent works in this area are inclined to use this method [2, 58]. The mathematical expression for measuring density-based heuristic value is shown in

$$\eta_{ij} = \frac{\left|\text{Term} = T_{ij} \text{ AND Class} = \text{Majority\_class}\left(T_{ij}\right)\right|}{\left|\text{Term} = T_{ij}\right|}. \tag{6}$$

The heuristic value measure used in AntMiner+ is very similar to the above. However, since the class is selected before the rule is constructed, the *Majority_class(T*$_{ij}$*)* is replaced by the class selected by the ant *class_ant* [2].

$$\eta_{ij} = \frac{\left|\text{Term} = T_{ij} \text{ AND Class} = \text{class\_ant}\right|}{\left|\text{Term} = T_{ij}\right|}. \tag{7}$$

In the previous two heuristic value functions, we evaluate the metric based on a ratio where there may be potential special cases of having small numbers in both denominator and numerator leading to high heuristic values, whereas there may be a considerably higher coverage with a small error in class prediction. To address this situation, Smaldon et al. proposed the following heuristic value function [64]. The same function was later adopted by Liang et al. [67]. Here, k represents the number of class labels.

$$\eta_{ij} = \frac{\left|\text{Term} = T_{ij} \text{ AND Class} = \text{class\_ant}\right| + 1}{\left|\text{Term} = T_{ij}\right| + k}. \tag{8}$$

Baig et al. reported another heuristic value function that looks into the correlation of the class label and the last selected term to the candidate term. The other part of the function contains the coverage by this triple. This method is expressed by equation (9), where T_{i*j*} represents the last selected term, T_{ij} represents a candidate term, and *Class*$_{committed}$ represents the committed class [63].

$$\eta_{ij} = \frac{\left|T_{i^*j^*} \text{ AND } T_{ij} \text{ AND Class}_{\text{committed}}\right|^2}{\left|T_{i^*j^*} \text{ AND } T_{ij}\right|\left|T_{i^*j^*} \text{ AND Class}_{\text{committed}}\right|}. \tag{9}$$

4.5. Pheromone Updating Strategies. The pheromone updating policy of AntMiner algorithms contains two aspects. First, which terms should get the pheromone update? And second, at what rate should the pheromone deposition and/or evaporation take place? Based on the philosophy of the ACO algorithm, the set of terms associated with each constructed rule should be evaluated by means of the relative quality of the rule. The quality of the rule dictates which terms to retain higher pheromone levels after an iteration. This is to be noted that all terms in the same rule will have the same degree of change in pheromone.

In AntMiner, only the best ant is allowed to deposit pheromone. The increase in pheromone is quantified by the product of Q (see Evaluation of Rule Quality section) and the current pheromone level. As there is only one ant in each ant cycle, the ant deposits pheromone to the terms used in the rule constructed. There is no direct evaporation factor considered in this approach. However, after pheromone deposition, the pheromone values are normalized over all terms. This passively reduces the pheromone level on the unused terms. During initialization, all of the terms are assigned with the same amount of pheromone [56].

$$\begin{aligned} \tau_{ij}(0) &= \frac{1}{\sum_{i=1}^{a} b_i}, \\ \tau_{ij}(n+1) &= \tau_{ij}(n) + \tau_{ij}(n) \cdot Q. \end{aligned} \tag{10}$$

In Liu et al.'s work, an exclusive pheromone evaporation factor ρ is introduced. Also, the use of Q is transformed in the pheromone updating function [58]. While the purpose of using ρ is easily understood from our previous discussion on ACO methodologies, the authors did not explicitly describe the motivation of using the transformed function of Q. Similar to AntMiner, the pheromone level for unused terms is updated by normalization; however, the pheromone level for terms used in the constructed rule is updated using

$$\tau_{ij}(n+1) = (1-\rho) \cdot \tau_{ij}(n) + \left(1 - \frac{1}{1+Q}\right) \cdot \tau_{ij}(n). \tag{11}$$

The AntMiner + algorithm initializes the pheromone values for each term with the maximum allowed pheromone value $\tau_{\max}$ [2]. In subsequent ant cycles, pheromone levels on all trails are subject to reduction due to evaporation. However, the pheromone on the best ant's path is reinforced (see (11)). The definition of coverage and confidence is provided in the Evaluation of Rule Quality section.

$$\tau_{\left(v_{ij},v_{i+1,k}\right)}(t+1) = \rho\tau_{\left(v_{ij},v_{i+1,k}\right)}(t) + \Delta\tau_{\left(v_{ij},v_{i+1,k}\right)}, \tag{12}$$

$$\Delta\tau_{\left(v_{ij},v_{i+1,k}\right)} = \begin{cases} \dfrac{(\text{confidence} + \text{coverage})}{10}, & \text{if arc}\left(v_{ij}, v_{i+1,k}\right)\text{is used by best ant} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Smaldon and Freitas used a preference operator to select the ants to deposit pheromone. The argument is to allow only the ants that constructed a rule that meets some acceptable threshold to deposit pheromone. The threshold is determined by the following function. If the Laplace corrected confidence is greater than the threshold, Q amount of pheromone is deposited (see equations (14) and (15)). Otherwise, no pheromone is added [64]. The function representing transitioning pheromone is the same as in AntMiner.

$$\text{Rule\_Confidence\_Threshold} = \max\left(0.5, \frac{\left|\text{Class} = \text{class}_{\text{ant}}\right|}{|\text{Training}|}\right), \tag{14}$$

$$\text{Laplace\_Corrected\_Confidence} = \frac{|\text{TP}| + 1}{|\text{TP} + \text{FP}| + k}. \tag{15}$$

4.6. Evaluation of Rule Quality

4.6. Evaluation of Rule Quality. The traditional method for evaluating the quality of a rule is to use the following metric which is the product of sensitivity and specificity [56, 58].

$$Q = \left(\frac{\text{TP}}{\text{TP} + \text{FN}}\right)\left(\frac{\text{TN}}{\text{TN} + \text{FP}}\right), \tag{16}$$

where TP: |Cases Covered by rule AND Class = Predicted Class|, FN: |Cases NOT Covered by rule AND Class = Predicted Class|, TN: |Cases NOT Covered by rule AND Class ≠ Predicted Class|, FP: |Cases Covered by rule AND Class ≠ Predicted Class|

Table 4 provides the definition of TP, FN, TN, and FP in a matrix form.

Two other means of rule quality as used in AntMiner + are coverage and confidence. In the context of a rule, confidence refers to the ratio of correctly classified instances over total instances covered by the rule; and coverage refers to the ratio of total instances covered by the rule over the total number of instances in the training data. Note that the training set size dynamically shrinks as new rules are discovered. These relations are defined in equations (17)–(20).

$$Q = \text{rule\_confidence} + \text{rule\_coverage}, \tag{17}$$

$$\text{rule\_confidence} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{18}$$

$$\text{rule\_coverage} = \frac{\text{TP} + \text{FP}}{|\text{Training}|}, \tag{19}$$

$$|\text{Training}| = \text{TP} + \text{FP} + \text{TN} + \text{FN}. \tag{20}$$

The AntMiner$_{\text{mbc}}$ proposed a different rule quality evaluation function using the same parameters as shown below [67].

$$Q = \left(\frac{\text{TP} + \text{FP}}{|\text{Training}|}\right)\left(\frac{\text{TP}}{\text{TN} + \text{FP}} - \frac{\text{TP} + \text{FN}}{|\text{Training}|}\right). \tag{21}$$

Further, Salama and Abdelbar conducted a study on various rule quality measures in the context of the μAntMiner algorithm. The suggested use of the Kappa function provides a better balance of average size and average accuracy (readers are referred to [68] for further information about this metric).

4.7. Rule Pruning Strategies. The rule pruning strategy in AntMiner takes place after each rule is constructed. This involves iteratively removing one term at a time from the rule and evaluating for improvement. The term whose removal results in the most improvement is removed from the rule. This process of removing one term at a time continues until the point where removal of no term results in improvement or there is only one term left in the rule. This is to be noted that, in AntMiner, every time a term is removed from the rule, the class label may be reassigned [1, 56].

The rule pruning in AntMiner+ is similar to this except it uses a different metric, confidence to evaluate the improvement of rules. Also, only the best rule from each ant cycle is allowed to go through the pruning process [2]. While

Table 4: Matrix defining TP, FN, FP, and TN in the context of classification rules.

	Covered (P)	*NOT Covered (N)*
Class = Predicted Class	TP	FN
Class ≠ Predicated Class	FP	TN

using a single ant population approach like AntMiner, the AntMiner-CC also allows only the best-so-far ant to go through pruning. This means the pheromone updating takes place without accounting for pruning.

Smaldon and Freitas used a pruning method where the class label is not changed during the pruning process and argued this reduces some computational effort as rule quality is to be evaluated for the selected class only [64].

In a later work, Chan and Freitas criticized the above methods for being computationally expensive and identified the pruning stage as a bottleneck for the algorithm. The algorithm would reportedly perform poorly for datasets with a larger number of attributes [69]. The authors proposed a "Faster Rule Pruning Procedure" to tackle this problem, inspired by [70]. In the essence, in the proposed method the original AntMiner's rule pruning operator is still used to reduce the length of a rule, but only on a stochastically reduced number of terms in the rule. The user is required to select the maximum number of terms (r) to be passed on to the original pruning operator. If the length of the currently constructed rule is greater than r, the algorithm reduces the length to r number of terms using roulette wheel selection. The rule pruning process of AntMiner is executed on the reduced rule containing selected terms only. The probability of selecting a term in the reduced rule is weighted according to the information gain achieved by that term. This is important to note that the pruning process gets the information gain achieved for terms precalculated by another procedure in AntMiner (see equation (5)). If the current rule contains less than r terms, the rule is directly passed to the rule pruning process of AntMiner.

In cAntMiner, the use of entropy-based discretization allows for a simpler rule pruning process. The author suggests that, due to the nature of rule construction, the continuous attributes can be removed in the reverse order they were added (see the Handling Continuous Attributes section for more information on the discretization process) [71].

In μcAntMiner, the rule pruning procedure is similar to the original AntMiner except that, for μcAntMiner, the consequent is preselected and does not change due to the pruning [72].

4.8. Handling Continuous Attributes.

4.8. Handling Continuous Attributes. In Parpinelli et al.'s method, the continuous attributes are discretized at the preprocessing step, where the C4.5 Disc discretization method is used for discretization. First, for each continuous attribute, a pair containing the continuous attribute and the class attribute data is fed to C4.5. Based on the output decision tree the continuous values of the attribute are replaced with categorical labels [1, 56].

Swaminthan used the Mixed Normal Kernel approach to handle continuous attributes. Although this approach still uses the C4.5 Disc discretization method for generating intervals, the intervals do not replace the original numeric values. After getting the intervals, the mean and standard deviations for each interval are calculated. These values are used to generate a Gaussian distribution. Then a multimodal mixed kernel function representing a probability distribution function is generated adding each kernel distribution corresponding to intervals. The area within each interval of the mixed kernel represents the pheromone value for that interval. When an interval is picked by an ant in a rule, a new kernel is added to the mixed kernel with the mean and standard deviation of the selected range. This represents an increase in pheromone level for the selected term [73].

In AntMiner+ the ants construct attribute intervals on the fly. The problem graph is first modified to accommodate for handling ordinal attributes. For each ordinal attribute, two vertices are considered. Each vertex holds a value from the ordinal attribute and the range between them is considered an interval. The algorithm forbids the second vertex to become the same or less than the first vertex in each attribute. For consistency, the categorical attributes also contain two vertices, where one is a dummy containing no data. The author suggests some discretization and/or chi-square-based approach to address data with high dimensionality [2].

The cAntMiner is another strategy to handle continuous attributes on the fly. The algorithm targets at finding the best split in the domain of the continuous attribute. The best split value is calculated based on an entropy measure as shown below in equation (22). Only the threshold values (v) from an attribute a_i that form boundaries between classes are evaluated. The definition of boundary values is given in [74]. Once a split is achieved, the half that gives less entropy is selected as a term [71]. This is important to note that the calculations for threshold value involve only the examples covered by the current partial solution. For continuous attributes, the pheromone can no longer be updated in the conventional manner. The pheromone is added to the attribute vertex (T_i) instead of a particular attribute-value pair. In equation (22), $|S_{a_i < v}|$ represents the number of cases where $a_i < v$, $|S_{a_i < v}|$ represents the number of cases where $a_i \geq v$, and $|S|$ represents the total number of cases. The best threshold value v_{best} will correspond to the threshold value that minimizes the entropy of the partition. After v_{best} is selected, the entropy of the term is given by equation (23).

$$ep_v(a_i, v) = \frac{|S_{a_i < v}|}{|S|}\text{entropy}(a_i < v) + \frac{|S_{a_i \geq v}|}{|S|}\text{entropy}(a_i \geq v), \quad (22)$$

$$\text{entropy}(T_i) = \min\{\text{entropy}(a_i < v_{\text{best}}), \text{entropy}(a_i \geq v_{\text{best}})\}. \quad (23)$$

Later on, Otero et al. published two further extensions of cAntMiner [75]. The first method is based on the Minimum Description Length (MDL) approach [74]. In this method, the previously mentioned splitting technique based on the threshold is recursively applied to find multiple intervals. Instead of referring to only one-half partition, the intervals can be more specific by giving a lower bound and an upper bound. The process still uses the threshold selection model from cAntMiner but in the next step, the threshold is passed through another criterion (see equations (23)–(26)) to accept or reject the threshold for an interval. Once a threshold value is selected the discretization is recursively repeated for each partition. In the end, the MDL approach may provide multiple intervals. The interval corresponding to the lowest entropy is selected. In equation (24), *c* represents the number of different classes contained in the training cases covered by the condition indicated in the subscripts of *c*.

$$\text{Gain}(a_i, v; S) > \frac{\log_2(|S| - 1)}{|S|} + \frac{\Delta(a_i, v; S)}{|S|}, \quad (24)$$

$$\text{Gain}(a_i, v; S) = \text{entropy}(S) - \frac{|S_{a_i < v}|}{|S|}\text{entropy}(S_{a_i} < v) + \frac{|S_{a_i \geq v}|}{|S|}\text{entropy}(S_{a_i} \geq v), \quad (25)$$

$$\Delta(a_i, v; S) = \log_2(3^c - 2) - \left[c \cdot \text{entropy}(S) - c_{a_i < v}\text{entropy}(S_{ai<v}) - c_{a_i \geq v}\text{entropy}(S_{ai<v})\right]. \quad (26)$$

The other extension in this work suggests that depositing pheromone on the edges instead of vertex could account for the interactions. Such an approach is also used in AntMiner+. However, in AntMiner+ the order of selection of attributes is predetermined. The influence of the order of selection of terms in the context of dynamic intervals is taken care of in this version of cAntMiner. Another modification suggested is for rule pruning. The author mentioned that once an attribute is removed from the rule disregarding the order as in rule pruning of AntMiner, that may change the context based on which the threshold value was selected. The author suggests that removing the terms from the rule in the reverse order of construction can provide more useful information. Recently, Helal and Otero suggested a probabilistic approach of discretizing continuous variables on the go [76].

Taking advantage of the preselected class, in μcAntMinter, the threshold values are calculated in the context of the selected class. The algorithm picks the threshold value providing maximum quality discrimination which is a function of support and confidence provided by each partition with respect to that threshold value [72].

4.9. Classification of Unseen Data. Once a classifier consisting of a set of decision rules is constructed, it can be used to classify new unseen instances of data. In this case, a new question arises, that is, which rule we should use to classify the new instance.

4.9.1. Ordered Rule Set. The most conventional way of constructing a rule-based classifier is to produce an ordered rule set, organized in the order of their discovery. When an unseen instance is to be classified, it is checked against the ordered rule set. The first rule to cover the instance will be used to classify it.

4.9.2. Unordered Rule Set. Smaldon and Freitas proposed an AntMiner methodology for using unordered rule sets for classification. In this case, while classifying an unseen instance, multiple situations are possible. Firstly, if no rule covers the instance the default class is assigned. Secondly, if there is only one rule that covers the instance, the class label is assigned based on the rule. Thirdly, if there are multiple rules which cover the instance but all of the rules suggest the same class, the suggested class is assigned. Finally, if there are multiple rules which cover the instance but a disagreement exists in regard to the consequent, a selection strategy is required. One of the two such selection strategies is to select the rule in agreement with the rule with the highest quality. The other is to select the class based on the class distribution [64].

4.9.3. Voting. Due to the stochastic nature of the AntMiner algorithms, the classifiers generated on the same training data at different times are likely to be different. To tackle this instability Liang et al. suggested AntMiner$_{\text{mbc}}$ which uses multiple rule sets in the classifier such that each rule set can

Table 5: List of datasets used in previous AntMiner implementations.

Name	Number of Instances	Number of Classes	Number of Attributes	Attribute types	Implementation
Ljubljana breast cancer	282–286	2	9	Categorical	[1, 2, 56, 62, 64, 67, 73]
Wisconsin breast cancer	683–699	2	9	Continuous	[1, 2, 56–58, 61–64, 73]
Hepatitis	155	2	19	Mixed	[1, 56, 63, 64, 67, 71–73, 75]
Dermatology	358–366	6	34	Mixed	[1, 56, 63–65, 67, 73]
Tic-tac-toe	958	2	9	Categorical	[2, 56, 58, 61–65, 67, 68, 73]
Cleveland heart disease	303	5	8	Mixed	[56, 64, 73]
Chess	3196	2	36	Categorical	[69]
House-votes-84	434	2	16	Categorical	[69]
Web-mining 1	124	3	159	*	[69]
Web-mining 2	124	3	293	*	[69]
Web-mining 3	124	3	339	*	[69]
Uniprot 1	540	2 × 2	153	Categorical	[77]
Uniprot 2	1343	2 × 2	156	Categorical	[77]
Uniprot 3	1872	2 × 2	102	Categorical	[77]
Uniprot 4	1826	2 × 2	101	Categorical	[77]
Uniprot 5	622	2 × 2	34	Categorical	[77]
German credit scoring	1000	2	30	Mixed	[62, 73]
Ripley's		2	2	Continuous	[2, 62]
Diabetes	768	2	8	Continuous	[73]
Thyroid	3772	4	30	Mixed	[73]
Ionosphere	350–351	2	34	Continuous	[63, 65, 67, 71–73, 75]
Contraceptive method choice	1473	3	9	Mixed	[2, 65, 67]
Iris	150	3	4	Continuous	[2, 67, 68, 72]
Teacher assistant evaluation	151	3	5	Mixed	[2, 63, 65, 68]
Balance scale	625	3	4	Categorical	[2, 63, 65, 67, 68]
car	1728	4	6	Categorical	[2, 63, 65, 67, 68]
Wine	178	3	13	Continuous	[2, 65, 68, 71, 75, 78]
Wdbc	569	2	30	Continuous	[63, 71, 75]
Crx	690	2	15	Mixed	[71, 75]
Glass	213–214	7	9	Continuous	[63, 65, 67, 68, 71, 75]
Australian	690	2	14	Mixed	[71, 75]
Heart	270	2	13	Mixed	[63, 71, 72, 75]
Congress house votes	435	2	17	Categorical	[63]
Credit history Australia	690	2	14	Mixed	[63, 65, 67, 68, 72]
Credit history Germany	1000	2	19–20	Mixed	[63, 65, 72]
Ecoli	336–366	8	7	Continuous	[63, 67, 72]
Haberman	307	2	3	Continuous	[63]
Hayes Roth	132–160	3	5	Categorical	[63, 65, 67, 68]
Image segmentation	2310	7	19	Continuous	[63]
Mammographic mass	961	2	5	Continuous	[63]
Pima Indian diabetes	768	2	8	Continuous	[63, 67]
SPECT heart	267	2	22	Categorical	[63, 65]
Transfusion blood	748	2	4	Continuous	[63, 72]
Vehicle	282	4	18		[63]
Zoo	282	7	16	Mixed	[63, 65, 68]
Annealing	896	6	38	Mixed	[72]
Automobile	205	7	25	Mixed	[72]
Breast cancer W	569	2	30	Continuous	[67, 72]
Cylinder bands	540	2	35	Mixed	[72]
Heart-c	303	5	12	Mixed	[65, 67, 72]
Heart-h	294	5	13	Mixed	[72]
Horse colic	365	2	22	Mixed	[72]
Parkinsons	195	2	22	Continuous	[72]
Segmentation	2269	7	19	Continuous	[72]
Hill-valley	606	2	100	Continuous	[67]
Liver-disorder	345	2	6	Continuous	[67]
Tae	151	3	5	Mixed	[67]
Monks	432	2	6	Categorical	[65, 68]
Post-operative patient	90	3	8	Mixed	[65, 68]

Table 5: Continued.

Name	Number of Instances	Number of Classes	Number of Attributes	Attribute types	Implementation
Voting records	435	2	16	Categorical	[65, 68]
Audiology	266	24	69	Categorical	[65]
Breast cancer (Wisconsin)	286	2	9	Continuous	[65]
Mushrooms	8124	2	22	Categorical	[65]
Soybean	307	19	35	Categorical	[65]
CSTR	360	6	3	Continuous	[78]
CSTR_ distillation	252	12	3	Continuous	[78]

Note: In [67, 71], the original size of the dataset is shown in the table. In the experiment duplicate values were removed and the number of instances was reduced. * Information not available.

complement the others when classifying unseen data [67]. Each of the rule sets is trained on a different subset of the original training data. While classifying unseen data, the class label is determined based on voting from each rule set. They also suggested a new heuristic function which was discussed in the heuristic value function section.

5. Experimental Settings

Typically, AntMiner algorithms are tested against their predecessors, decision tree counterparts, and high-accuracy yielding algorithms such as SVM, ANN. In this section, we discuss the performance criteria for algorithm evaluation in the context of rule-based classifiers and provide a list of datasets used in the previous implementations.

5.1. Performance Criteria. The two most common criteria for evaluating rule-based classifiers are predictive accuracy and interpretability. The predictive accuracy metric is commonly reported in terms of mean accuracy and standard deviation. The interpretability is measured using the number of rules in the classifier and the average length of the rules. The length of the rules is calculated using the number of terms used in the rule. In many implementations, 10-fold cross-validation is used for performance evaluation. Hence the average and standard deviations for each of accuracy, no. of rules, and no. of terms per rule are reported.

5.2. Data Sets. In Table 5, we have listed the datasets used for testing different AntMiner versions for data classification. Most datasets are available on the UCI machine learning repository except for Web-mining and Uniprot. The authors of this review article neither own nor maintain these datasets. Any data related to questions may be forwarded to the corresponding author of the referenced article. Note that the goal of this paper is to provide a map to assist future researchers for the benchmarking purpose.

6. Future Research Directions

There are several ACO strategies in the general optimization domain which have not been utilized in the AntMiner family yet, for example, the dynamic balance of diversification and intensification by Yan et al. [54] and using ACO for problems with continuous domains by Socha and Dorigo [79]. This would be interesting to see how the latest developments of ACO in the optimization domain perform when adopted for the classification problem.

The existing AntMiner implementations mostly focus on extending on Ant System and Max-Min Ant System. As reported in the review there are other ACO methodologies that are yet to be implemented for classification. For example, one of the oldest ACO techniques Ant Colony System (ACS) is not yet implemented for classification. Although Liu et al. used the probability function of ACS, the rule construction strategy is still built on the Ant System.

We have limited information with regard to the performance of AntMiner algorithms and their modules. Ali and Shahzad provided a comparative analysis of several AntMiner versions. However, their experiments used the default parameter setting [80]. This is imperative to optimize the parameters for all of the variations before conducting such experiments for a fair comparison. The algorithmic framework allows easy adaptation of modules of one algorithm version into another. Further analyses of computational complexity and experimental studies involving different ACO modules such as probability function, heuristic, and pheromone updating strategy are needed to make conclusive remarks on their performance.

Parameter tuning is a big contributor to the successful implementation of AntMiner algorithms. In the general optimization area, there have been studies that report a suggested range for some parameters. However, the classification problem may benefit from specialized parameter settings for AntMiner algorithms. An extensive study on optimal parameter setting for AntMiner algorithms is yet to be done.

Several scholars identify the rule pruning procedure as the most computationally expensive procedure in the AntMiner algorithms. While some improvements are suggested in terms of performance, all of them come with some trade-off. Further research on rule pruning is needed to find more efficient as well as high-accuracy yielding procedures.

One particular rule pruning by Chan and Freitas suggests using the conventional rule pruning on a randomly selected part of a constructed rule [69]. Their strategy involves a user decision on how many terms to be passed on to the rule pruning procedure. Without informed choice, this may result in a trade-off in accuracy for faster operation. A suggested range of values for r that works well for all rules is yet to be determined. Another potential path for handling this is to find a dynamic strategy for selecting *r*.

Given the computational intelligence used in AntMiner algorithms, they have the potential to be useful for datasets with a high number of attributes and values. Also, systems requiring online classification can be particularly benefitted from such heuristics. There are some experiments reported on large datasets involving limited versions of AntMiner methodologies, reporting competitive results. There is no application for online classification reported in the literature.

In recent times, there is a rise in research on parallel implementation strategies in both of the ACO and rule-based classification domains [81–83]. However, none of such strategies were adopted for AntMiner algorithms yet. We believe this will be a timely contribution to explore effective parallel implementation strategies for AntMiner algorithms.

7. Conclusions

Many real-life applications of machine learning tools demand easy interpretations of the classifiers to humans, in addition to yielding high accuracy. This could be due to reliability, accountability, ethics, or other concerns. Rule-based classifiers have been a successful tool for such applications. We have provided a list of such applications to outline the application areas that already benefitted from the rule-based classification technique. Due to the high computational complexity and combinatorial nature of the construction process of rule-based classifiers, heuristic optimization algorithms such as AntMiner are considered to be a helpful method. In this article, an extensive review of the state-of-the-art AntMiner algorithms was conducted, and the relevant future research directions were suggested. Our suggestions emphasize exploring ACO methodologies including parallel implementation strategies, conducting experimental studies to find recommended parameter settings, comparative experimental studies to systematically evaluate the performance of existing strategies at the modular level, developing new strategies for rule pruning, and exploring the potential of AntMiner for online classification. This work will be beneficial to the researchers devoting their effort to improving or deploying the metaheuristic for rule-based classification.

Data Availability

This is a review article that does not deal with any datasets. To access the datasets cited in this article, the readers are referred to the source articles' authors.

Conflicts of Interest

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

Acknowledgments

This research was partially supported by the US Department of Agriculture (USDA) under Grant no. 2021-67037-34163.

References

[1] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "An ant colony based system for data mining: applications to medical data," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, San Francisco, CA, USA, July 2001.

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

[3] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[4] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[5] R. A. Hasan, M. A. Mohammed, N. Ţăpuş, and O. A. Hammood, "A comprehensive study: ant Colony Optimization (ACO) for facility layout problem," in *Proceedings of the 2017 16th RoEduNet conference: Networking in Education and Research (RoEduNet)*, pp. 1–8, Targu-Mures, Romania, September 2017.

[6] A. Kole, P. Chakrabarti, and S. Bhattacharyya, "An ant colony optimization algorithm for uncapacitated facility location problem," in *Proceedings of the 38th International Conference on Computers and Industrial Engineering*, Beijing, China, October-November 2013.

[7] A. Freitas, R. Parpinelli, and H. Lopes, "Ant colony algorithms for data classification," *Encyclopedia of Information Science and Technology*, vol. 1, pp. 154–159, 2009.

[8] G. Wang, Z. Deng, and K.-S. Choi, "Detection of epilepsy with Electroencephalogram using rule-based classifiers," *Neurocomputing*, vol. 228, pp. 283–290, 2017.

[9] J. L. Domínguez-Olmedo, J. Mata, V. Pachón, and J. L. L. Guerra, "Application of a rule-based classifier to data regarding radiation toxicity in prostate cancer treatment," in *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies*, pp. 384–398, Funchal, Madeira, Portugal, January 2018.

[10] S. Narayan and J. Gobal, "Optimal decision tree fuzzy rule-based classifier for heart disease prediction using improved cuckoo search algorithm," *International Journal of Business Intelligence and Data Mining*, vol. 15, no. 4, pp. 408–429, 2019.

[11] M. N. Gurcan, B. Sahiner, N. Petrick et al., "Lung nodule detection on thoracic computed tomography images: preliminary evaluation of a computer-aided diagnosis system," *Medical Physics*, vol. 29, no. 11, pp. 2552–2558, 2002.

[12] Q. Li, F. Li, and K. Doi, "Computerized detection of lung nodules in thin-section CT images by use of selective enhancement filters and an automated rule-based classifier," *Academic Radiology*, vol. 15, no. 2, pp. 165–175, 2008.

[13] G. Thippa Reddy and N. Khare, "FFBAT-optimized rule based fuzzy logic classifier for diabetes," *International Journal of Engineering Research in Africa*, vol. 24, pp. 137–152, 2016.

[14] O. Wieben, V. X. Afonso, and W. J. Tompkins, "Classification of premature ventricular complexes using filter bank features, induction of decision trees and a fuzzy rule-based system," *Medical, & Biological Engineering & Computing*, vol. 37, no. 5, pp. 560–565, 1999.

[15] O. R. Zaıane, M.-L. Antonie, and A. Coman, "Mammography classification by an association rulebased classifier," in *Proceedings of the Third International Workshop on Multimedia*

Data Mining, MDM/KDD'2002, pp. 62–69, Edmonton, Alberta, Canada, July 2002.

[16] M. Sabeti, M. Sadreddini, and J. T. Nezhad, "EEG signal classification using an association rule-based classifier," in *Proceedings of the 2007 IEEE International Conference on Signal Processing and Communications*, pp. 620–623, Dubai, UAE, November 2007.

[17] A. N. Nguyen, M. J. Lawley, D. P. Hansen et al., "Symbolic rule-based classification of lung cancer stages from free-text pathology reports," *Journal of the American Medical Informatics Association*, vol. 17, no. 4, pp. 440–445, 2010.

[18] T. Deepa, B. Sathiyabhama, J. Akilandeswari, and N. P. Gopalan, "Action fuzzy rule based classifier for analysis of dermatology databases," *International Journal of Biomedical Engineering and Technology*, vol. 15, no. 4, pp. 360–379, 2014.

[19] J. M. Alonso, C. Castiello, M. Lucarelli, and C. Mencar, "Modeling interpretable fuzzy rule-based classifiers for medical decision support," in *Data Mining: Concepts, Methodologies, Tools, and Applications*, pp. 1064–1081, IGI Global, Hershey, PA, USA, 2013.

[20] Q. Li and K. Doi, "Analysis and minimization of overtraining effect in rule-based classifiers for computer-aided diagnosis," *Medical Physics*, vol. 33, no. 2, pp. 320–328, 2006.

[21] Q. A. Al-Radaideh and S. S. Al-Khateeb, "An associative rule-based classifier for Arabic medical text," *International Journal of Knowledge Engineering and Data Mining*, vol. 3, no. 3-4, pp. 255–273, 2015.

[22] I. Solt, D. Tikk, V. Gal, and Z. T. Kardkovacs, "Semantic classification of diseases in discharge summaries using a context-aware rule-based classifier," *Journal of the American Medical Informatics Association*, vol. 16, no. 4, pp. 580–584, 2009.

[23] T. Lehr, J. Yuan, D. Zeumer, S. Jayadev, and M. D. Ritchie, "Rule based classifier for the analysis of gene-gene and gene-environment interactions in genetic association studies," *BioData Mining*, vol. 4, no. 1, p. 4, 2011.

[24] P. R. Kumar and V. Ravi, "Bankruptcy prediction in banks by fuzzy rule based classifier," in *Proceedings of the 2006 1st International Conference on Digital Information Management*, pp. 222–227, IEEE, Bangalore, India, December 2006.

[25] H. Lei, C. Chan, and J. Cheh, "Rule-based classifier for bankruptcy prediction," in *Proceedings of the Fourteenth Midwest Artificial Intelligence and Cognitive Sciences Conference*, pp. 74–81, Cincinnati, OH, USA, April 2003.

[26] B. T. Pham, D. Tien Bui, I. Prakash, and M. B. Dholakia, "Rotation forest fuzzy rule-based classifier ensemble for spatial prediction of landslides using GIS," *Natural Hazards*, vol. 83, no. 1, pp. 97–127, 2016.

[27] J. Li and W. Chen, "A rule-based method for mapping Canada's wetlands using optical, radar and DEM data," *International Journal of Remote Sensing*, vol. 26, no. 22, pp. 5051–5069, 2005.

[28] D. G. Stavrakoudis, G. N. Galidaki, I. Z. Gitas, and J. B. Theocharis, "A genetic fuzzy-rule-based classifier for land cover classification from hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 1, pp. 130–148, 2011.

[29] P. Bolstad and T. Lillesand, "Rule-based classification models-Flexible integration of satellite imagery and thematic spatial data," *Photogrammetric Engineering & Remote Sensing*, vol. 58, no. 7, pp. 965–971, 1992.

[30] T. Hachaj and M. R. Ogiela, "Rule-based approach to recognizing human body poses and gestures in real time," *Multimedia Systems*, vol. 20, no. 1, pp. 81–99, 2014.

[31] A. Riad, H. K. Elminir, and S. M. Shohieb, "Hand gesture recognition system based on a geometric model and rule based classifier," *British Journal of Applied Science & Technology*, vol. 4, no. 9, pp. 1432–1444, 2014.

[32] X. Fang and A. Alouani, "Unconstrained handwritten numeral recognition using fuzzy rule-based classifier," in *Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No. 02EX540)*, pp. 256–260, Huntsville, AL, USA, March 2002.

[33] E. Soares, P. Angelov, B. Costa, and M. Castro, "Actively semi-supervised deep rule-based classifier applied to adverse driving scenarios," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Budapest, Hungary, July 2019.

[34] Q. Yang, T. Li, and K. Wang, "Building association-rule based sequential classifiers for web-document prediction," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 253–273, 2004.

[35] K. C. Gross, C. J. Digate, and E. H. Lee, "Event-driven rule-based messaging system," *Google Patents*, 1994.

[36] W. Zhou, A. Vellaikal, and C. J. Kuo, "Rule-based video classification system for basketball video indexing," in *Proceedings of the 2000 ACM Workshops on Multimedia*, pp. 213–216, Los Angeles, CA, USA, October-November 2000.

[37] N. Duffield, P. Haffner, B. Krishnamurthy, and H. A. Ringberg, "Systems and methods for rule-based anomaly detection on IP network flow," *Google Patents*, 2016.

[38] S. Ghosh, A. Pal, A. Nag, S. Sadhu, and R. Pati, "Network anomaly detection using a fuzzy rule-based classifier," *Computer, Communication and Electrical Technology*, pp. 61–65, 2017.

[39] T. Nakashima, G. Schaefer, Y. Yokota, and H. Ishibuchi, "A weighted fuzzy classifier and its application to image processing tasks," *Fuzzy Sets and Systems*, vol. 158, no. 3, pp. 284–294, 2007.

[40] U. A. Siddiqua, T. Ahsan, and A. N. Chy, "Combining a rule-based classifier with ensemble of feature sets and machine learning techniques for sentiment analysis on microblog," in *Proceedings of the 2016 19th International Conference on Computer and Information Technology (ICCIT)*, pp. 304–309, Dhaka, Bangladesh, December 2016.

[41] M. Chau, T. M. Li, P. W. Wong, J. J. Xu, P. S. Yip, and H. Chen, "Finding people with emotional distress in online social media: a design combining machine learning and rule-based classification," *MIS Quarterly*, vol. 44, no. 2, 2020.

[42] G. Xiao, Q. Cheng, and C. Zhang, "Detecting travel modes using rule-based classification system and Gaussian process classifier," *IEEE Access*, vol. 7, pp. 116741–116752, 2019.

[43] H. Shahparast, S. Hamzeloo, and M. Z. Jahromi, "A self-tuning fuzzy rule-based classifier for data streams," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 22, no. 02, pp. 293–303, 2014.

[44] M. Heidari, "Fault detection of bearings using a rule-based classifier ensemble and genetic algorithm," *International Journal of Engineering*, vol. 30, no. 4, pp. 604–609, 2017.

[45] T. Rossi and J. Braun, "A statistical, rule-based fault detection and diagnostic method for vapor compression air conditioners," *HVAC & R Research*, vol. 3, no. 1, pp. 19–37, 1997.

[46] D. Dou, J. Jiang, Y. Wang, and Y. Zhang, "A rule-based classifier ensemble for fault diagnosis of rotating machinery,"

Journal of Mechanical Science and Technology, vol. 32, no. 6, pp. 2509–2515, 2018.

[47] R. Nakagaki, M. Kurihara, and T. Honda, "Defect classifier using classification recipe based on connection between rule-based and example-based classifiers," *Google Patents*, 2011.

[48] Y. Bidulya and E. Brunova, "Sentiment analysis for bank service quality: a rule-based classifier," in *Proceedings of the 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–4, Baku, Azerbaijan, October 2016.

[49] R. Kumar, B. Singh, D. Shahani, A. Chandra, and K. Al-Haddad, "Recognition of power-quality disturbances using S-transform-based ANN classifier and rule-based decision tree," *IEEE Transactions on Industry Applications*, vol. 51, no. 2, pp. 1249–1258, 2014.

[50] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant," *Journal of Insect Behavior*, vol. 3, no. 2, pp. 159–168, 1990.

[51] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[52] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[53] O. Cordón García, F. Herrera Triguero, and T. Stützle, "A review on the ant colony optimization metaheuristic: basis, models and new trends," *Mathware and Soft Computing*, vol. 9, 2002.

[54] Y. Yan, H.-s. Sohn, and G. Reyes, "A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem," *Applied Soft Computing*, vol. 60, no. C, pp. 256–267, 2017.

[55] K. P. Bennett and E. Parrado-Hernández, "The interplay of optimization and machine learning research," *Journal of Machine Learning Research*, vol. 7, pp. 1265–1281, 2006.

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

[57] B. Liu, H. A. Abbass, and B. McKay, "Density-based heuristic for rule discovery with ant-miner," in *Proceedings of the 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System*, Canberra, Australia, November 2002.

[58] B. Liu, H. A. Abbas, and B. McKay, "Classification rule discovery with ant colony optimization," in *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, pp. 83–88, Halifax, NS, Canada, October 2003.

[59] L. M. Gambardella and M. Dorigo, "Ant-Q: a Reinforcement Learning approach to the traveling salesman problem," in *Machine Learning Proceedings 1995*, A. Prieditis and S. Russell, Eds., 252–260, Morgan Kaufmann, San Francisco (CA), 1995.

[60] R. Sun, S. Tatsumi, and G. Zhao, "Multiagent reinforcement learning method with an improved ant colony system," vol. 3, pp. 1612–1617, 2001.

[61] B. Liu, H. A. Abbass, and R. I. McKay, "Classification rule discovery with ant colony optimization," *IEEE Intelligent Informatics Bulletin*, vol. 3, no. 1, pp. 31–35, 2004.

[62] D. Martens, M. De Backer, R. Haesen, B. Baesens, and T. Holvoet, "Ants constructing rule-based classifiers," in *Swarm Intelligence in Data Mining*, pp. 21–43, Springer, Berlin, Germany, 2006.

[63] A. R. Baig, W. Shahzad, and S. Khan, "Correlation as a heuristic for accurate and comprehensible ant colony optimization based classifiers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 686–704, 2012.

[64] J. Smaldon and A. A. Freitas, "A new version of the ant-miner algorithm discovering unordered rule sets," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 43–50, Seattle, WA, USA, July 2016.

[65] K. M. Salama, A. M. Abdelbar, and A. A. Freitas, "Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 149–182, 2011.

[66] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Hoboken, NY, USA, 1991.

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

[68] K. M. Salama and A. M. Abdelbar, "Exploring different rule quality evaluation functions in aco-based classification algorithms," in *Proceedings of the 2011 IEEE Symposium on Swarm Intelligence*, pp. 1–8, Paris, France, April 2011.

[69] A. Chan and A. Freitas, "A new classification-rule pruning procedure for an ant colony algorithm," in *Proceedings of the International Conference on Artificial Evolution (Evolution Artificielle)*, pp. 25–36, Lille, France, October 2005.

[70] D. R. Carvalho and A. A. Freitas, "A hybrid decision tree/genetic algorithm method for data mining," *Information Sciences*, vol. 163, no. 1-3, pp. 13–35, 2004.

[71] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes," in *Ant Colony Optimization and Swarm Intelligence*, pp. 48–59, Springer Berlin Heidelberg, Berlin, Germany, 2008.

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

[73] S. Swaminathan, *Rule Induction Using Ant Colony Optimization for Mixed Variable Attributes*, Texas Tech University, Lubbock, TX, USA, 2006.

[74] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," *IJCAI*, pp. 1022–1027, 1993.

[75] F. E. Otero, A. A. Freitas, and C. G. Johnson, "Handling continuous attributes in ant colony classification algorithms," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 225–231, Nashville, TN, USA, March 2009.

[76] A. Helal and F. E. Otero, "A mixed-attribute approach in ant-miner classification rule discovery algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 13–20, Denver CO, USA, July 2016.

[77] A. Chan and A. A. Freitas, "A new ant colony algorithm for multi-label classification with applications in bioinfomatics," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 27–34, Seattle, WA, USA, July 2016.

[78] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony classifier system: application to some process engineering problems," *Computers & Chemical Engineering*, vol. 28, no. 9, pp. 1577–1584, 2004.

[79] K. Socha and M. Dorigo, “Ant colony optimization for continuous domains,” *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[80] Z. Ali and W. Shahzad, “Comparative analysis and survey of ant colony optimization based rule miners,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 1, pp. 49–60, 2017.

[81] J. Peake, M. Amos, N. Costen, G. Masala, and H. Lloyd, “PACO-VMP: parallel ant colony optimization for virtual machine placement,” *Future Generation Computer Systems*, vol. 129, pp. 174–186, 2022.

[82] J. Yu, “A novel parallel ant colony optimization algorithm for warehouse path planning,” *Journal of Control Science and Engineering*, vol. 2020, 2020.

[83] A. Govada, V. S. Thomas, I. Samal, and S. K. Sahay, “Distributed multi-class rule based classification using RIPPER,” in *Proceedings of the 2016 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 303–309, Nadi, Fiji, December 2016.