DISCOVERING UNORDERED RULE SETS FOR MIXED VARIABLES USING AN ANT-MINER ALGORITHM

C. Nalini[1*] and P. Balasubramanie[2]

*[*1]Department of Information Technology, Kongu Engineering College, Erode, India*
Email: nalinikec@gmail.com
[2]Dept. of Computer Science & Engineering, Kongu Engineering College, Erode, India

ABSTRACT

This work proposes a data mining algorithm called Unordered Rule Sets using a continuous Ant-Miner algorithm. The goal of this work is to extract classification rules from data. Swarm intelligence (SI) is a technique whereby rules may be discovered through the study of collective behavior in decentralized, self-organized systems, such as ants. The Ant-Miner algorithm, first proposed by Parpinelli and his colleagues (2002), applies an ant colony optimization (ACO) heuristic to the classification task of data mining to discover an ordered list of classification rules. Ant-Miner is a rule-induction algorithm that uses SI techniques to form rules. Ant-Miner uses a discretization process to deal with continuous attributes in the data. Discretization transforms numeric attributes into nominal attributes. Discretization may suffer from a loss of information, as the real relationship underlying individual values of a numeric attribute is unknown. The objective of this work is to apply ACO heuristic techniques to discover unordered rule sets for mixed variables in a data set. The proposed algorithm handles both nominal and continuous attributes using multimodal functions. It has the advantage of discovering more modular rules, i.e., rules that can be interpreted independently from other rules – unlike the rules in an ordered list, where the interpretation of a rule requires knowledge of the previous rules in the list. The results provide evidence that the accuracy of the Unordered Rule Set Continuous Ant-Miner algorithm is competitive with other Ant-Miner versions and generates simpler rule sets.

Key words: Ant Colony Optimization, Data mining, Classification rules

1 INTRODUCTION

Data mining refers to the extraction of knowledge from large amounts of data. Classification is one of the most frequently occurring tasks of human decision-making. A classification problem encompasses the assignment of an object to a predefined class according to its characteristics. Many decision problems in a variety of domains, such as engineering, medical sciences, human sciences, and management science, can be considered as classification problems. Rule discovery is an important data mining task, as it generates a set of symbolic rules that describe each class or category in a natural way. However, these rules need to be simple and comprehensive; otherwise, a human is unable to comprehend them. To our knowledge, Parpinelli, Lopes, and Freitas (2002) were the first to propose Ant Colony Optimization (ACO) for discovering classification rules using the system Ant-Miner. They argue that an ant-based search is more flexible and robust than traditional approaches. Their method uses a heuristic value based on entropy measure and generates ordered rule sets. Ant-Miner uses a discretization process to deal with continuous attributes in the data. Even though the discretization process reduces the overhead of handling large sets of values occurring in a data set, the real relationship among numerical values is unknown. In our work, we handle continuous attributes directly using multimodal functions and also generate unordered rule sets.

The remainder of the paper is organized as follows. In Section 2, we present the basic idea of the ant colony systems, and the Ant-Miner algorithm (Parpinelli et al, 2000) is introduced. In Section 3, the ACO in continuous space is discussed. In Section 4, the ACO algorithm for a mixed variable, i.e., the proposed algorithm, is discussed. Then the computational results are reported in Section 5. Finally, we conclude with general remarks on this work and directions for future research in Section 6.

2 ANT COLONY SYSTEM (ACS)

Ant Colony Optimization (Dorigo, Colorni, & Maniezzo, 1996) is a branch of a newly developed form of artificial intelligence called swarm intelligence. Swarm intelligence is a field that studies “the emergent collective intelligence of groups of simple agents” (Bonabeau, Dorigo, & Theraulaz, 1999). In groups of insects living in colonies, such as ants and bees, an individual can do only simple tasks on its own, while the colony's

cooperative work is the main reason for the intelligent behavior it exhibits. Most real ants are blind. However, each ant, while it is walking, deposits a chemical substance on the ground called a pheromone (Dorigo et al., 1996). Pheromones encourage the following ants to stay close to previous moves. The pheromone evaporates over time to allow search exploration.

A number of experiments presented in Dorigo et al. (1996) illustrate the complex behavior of ant colonies. For example, a set of ants built a path to some food. An obstacle with two ends was placed in their way such that one end of the obstacle was more distant than the other. In the beginning, equal numbers of ants had spread around the two ends of the obstacle. Because all ants had almost the same speed, the ants going around the nearer end of the obstacle returned before the ants going around the farther end (differential path effect). With time, the amount of pheromones the ants deposited increased more rapidly on the shorter path, and so more ants preferred this path. This positive effect is called autocatalysis. The difference between the two paths is called the preferential path effect; it is the result of the differential deposition of pheromones between the two sides of the obstacle. The ants following the shorter path made more visits to the source than those following the longer path. Because of pheromone evaporation, pheromones on the longer path vanished with time.

Ant-Miner is a sequential covering algorithm that incorporates the concepts and principles of ACO and rule induction. The goal of Ant-Miner is to extract classification rules from data (Parpinelli et al., 2002). In Ant-Miner, an artificial ant conducts loops of three procedures, rule construction, rule pruning, and pheromone updating, to induce rules from a current training data set. An artificial ant then constructs a rule by visiting a set of possible nodes in the graph and forms a path that ends at a class term node. A complete path is a constructed rule. Ant-Miner has the limitation that the numerical attributes have to be preprocessed. The Ant-Miner algorithm is presented as follows.

Ant-Miner algorithm:

Training set = all training cases;
WHILE (No. of cases in the Training set > max_uncovered_cases)
i=0;
    REPEAT
    i=i+1;
    Ant_i incrementally constructs a classification rule;
    Prune the just constructed rule;
    Update the pheromone of the trail followed by Ant_i;
    UNTIL (i = No_of_Ants) or (Ant_i constructed the same rule as the
        previous No_Rules_Converg-1 Ants)
Select the best rule among all constructed rules;
Remove the cases correctly covered by the selected rule from the training set;
END WHILE

3 ACO IN CONTINUOUS OPTIMIZATION

The original ant system (AS) introduced by Dorigo et al. (1999) was applicable only to a discrete problem space. ACO is an efficient tool for solving various combinatorial optimization problems. As a result, Bilchev and Parmee (1995) suggested a method to extend ACO to continuous spaces in their work called Continuous Ant Colony Optimization (CACO). They suggested considering a finite set of regions at each iteration of the AS algorithm. Agents (ants) are sent to these regions, from which they explore randomly selected directions within a radius of exploration. Agents reinforce their paths according to their performance. However, CACO does not perform an incremental construction of solutions, which is one of the main characteristics of the ACO meta heuristics. Further, CACO is dedicated strictly to continuous optimization problems and is not used for mixed variables. In the Pachycondyla Apicalis (API) optimization algorithm (Monmarche, Venturini, & Slimane, 2000), ants perform their search independently and do not use any artificial pheromone information. Continuous Interacting Ant Colony (CIAC) (Dreo & Siarry 2002) uses two types of communication among ants. However, CIAC does not perform an incremental construction of the solution, which is a virtual part of the standard ACO algorithm. Another notable work in this area is that of Krzysztof Socha (2004) who suggests a way to extend the generic ACO to the continuous domain by considering a mixture of several Probability Density Functions (PDFs). The Probability Density Function(PDF) of a continuous random variable is a function which can be integrated to obtain the probability that the random variable takes a value in a given interval. PDF is used to find the point of Normal Distribution curve. Continuous Probability Density Function of the Normal Distribution is called the Gaussian Function.

4 ACO IN MIXED VARIABLE OPTIMIZATION

In the original Ant-Miner (Parpinelli et al, 2000), ants choose terms for a rule with the goal of decreasing entropy in the class distribution of examples matching the rule in construction. The consequent of the rule is then assigned afterwards by determining the class value that would produce the highest quality rule. The goal of the original Ant-Miner algorithm was to produce an ordered list of rules, which was then applied to test data in the order in which they were discovered. This makes it difficult to interpret the rules at the end of the list, as their conditions make sense only in the context of all the previous rules in the ordered list of rules. In Unordered Rule Set Ant-Miner (Smaldon, & Freitas, 2006), the consequent for the rule is known by the ant during rule construction at the initial stage and leads to faster convergence on good rules in comparison with the original Ant-Miner. The reason for this is that in Unordered Rule Set Ant-Miner each term's pheromone value directly represents that term's relevance for predicting a fixed target class value. By contrast, in the original Ant-Miner each term's pheromone is associated with that term's relevance in reducing the entropy associated with the entire class distribution, a less focused relevance. In Ant-Miner and Unordered Rule Set Ant-Miner any continuous attributes in the dataset have to be discretized prior to the rule induction process. Continuous Ant-Miner (Swaminathan, 2006) handled continuous attributes in the datasets without any preprocessing. It chose the terms for a rule based on entropy value and generated an ordered rule list. The proposed algorithm handles both nominal and continuous attributes. In this case, ants choose terms for a rule based on Laplace-corrected confidence (Clark & Boswell, 1991) and generated an unordered rule set.

In Continuous Ant-Miner (Swaminathan, 2006), the discretization process is handled within the algorithm. As a result, statistical information about the data, such as the mean and standard deviation of the ranges, can be calculated and used in the rule induction process. In this process, any continuous attribute present in the original dataset is split into ranges by the C4.5 discretization procedure (Bonabeau, Dorigo, & Theraulaz, 1999). C4.5 discretization procedure follows a Divide and Conquer strategy. C4.5 first sorts the numeric values and then uses the mid point between two numeric values as the branching value. The information gain is calculated for each of the branching points in a similar manner to nominal attributes, and the branching point with the highest information gain is chosen to split the data set into smaller subsets. The minimum and the maximum values of the ranges are stored for further use during the calculation of the mean and standard deviation of the ranges. After preprocessing, the Continuous Ant-Miner algorithm generates k runs, where k is the number of folds in a k-fold cross validation. In a k-fold cross validation, the dataset is randomly shuffled and split into k approximately equal-sized, mutually-exclusive subsets. Each of the k subsets forms the testing dataset, while the other (k-1) subsets are used as the training dataset. Each set of training and testing sets forms a fold of data, and each fold of data is applied to an algorithm to test the performance of the algorithm (Kohavi, 1995).

4.1 Algorithm for Continuous Ant-Miner for Unordered Rule Set

RuleSet = [ ] // initialized as empty set
FOR EACH Class
TrainingSet = {all training cases}
PositiveSet = {training cases of current class}
NegativeSet = TrainingSet – PositiveSet
Calculate the mean and the standard deviations of each range
Initialize the pheromones; in case of continuous attributes pheromone is a mixed kernel PDF with each range representing a PDF with the corresponding mean and standard deviation
WHILE (|PositiveSet| > max_uncovered_cases)
t = 1;
j = 1;
REPEAT // iteration for constructing a rule

Ant_t starts with an empty rule and incrementally constructs a classification rule R_t by adding one term at a time to the current rule;
Prune rule R_t;
IF(LaplaceCorrectedConfidence(R_t)>ruleConfidenceThreshold
THEN increase pheromone of terms in rule R_t
In case of a continuous attribute add a PDF corresponding to the range
END IF

Update pheromones in all other terms by normalising the pheromone values (simulating evaporation)
IF (R_t equals R_t -1) **THEN** j = j + 1;
ELSE j = 1;
END IF
t = t+1;
UNTIL (t = No_of_ants) OR (j = No_rules_converge)
Select the best rule among all constructed rules;
Add the best rule to RuleSet;
TrainingSet = TrainingSet – {set of positive cases covered by R_{best}};
PositiveSet = PositiveSet – {set of positive cases covered by R_{best}};
END WHILE
Output RuleSet;
END FOR

In unordered rule sets for mixed variables using an ant-miner algorithm, an extra For-Each loop is added as the outer loop of the algorithm, iterating over the values in the class attribute domain. Each iteration of the For-Each loop discovers an unordered set of rules, all of which predict the current class value. At the beginning of each iteration, the entire training set is reinstated, so that a maximal number of negative examples are available to the algorithm. Ants discover rules from the training data until the number of positive examples (belonging to the current class) remaining in the dataset that have not been covered by a discovered rule is less than or equal to the value determined by the *max_uncovered_cases* parameter. Note that in the original Ant-Miner, *max_uncovered_cases* referred to all examples in the training set, rather than to the positive examples only as in the proposed algorithm. For each iteration of the WHILE loop, the amount of pheromone for each term is set to an initial value. This initialization is the same as the one used in the original Ant-Miner. An ant starts with a rule containing the known consequence and an empty antecedent. The rule is constructed incrementally by selecting terms with a probabilistic method that favors terms with a large amount of pheromone and a high Laplace-corrected confidence value. The ant stops constructing a rule if all the attributes have been used in the rule antecedents constructed so far or if there are no terms available that, when added to the rule antecedent, would not make the rule cover fewer cases than the limit *min_cases_per_rule*. The rule is then pruned in an attempt to increase its quality. If the rule is of a high enough confidence, the terms making up the rule have their pheromone levels increased. The best rule discovered during the REPEAT UNTIL loop is added to the unordered set of discovered rules.

4.2 Term Selection-Probability Transition rule

The probability that a term will be added to the current rule is given by the following formula:

$$P_{ij} = \frac{\eta_{ij} \cdot \tau_{ij}(t)}{\sum_{i=1}^{a} x_i \cdot \sum_{j=1}^{b_i} \left(\eta_{ij} \cdot \tau_{ij}(t)\right)} \qquad (1)$$

where η_{ij} is the value of a problem-dependent heuristic function to select attribute value pairs used to form the conditions of the rule, $\tau_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration t, and a is the total number of attributes.

4.3 Problem dependent heuristic function

The problem dependent heuristic function chosen is the Laplace-corrected confidence (Clark & Boswell, 1991) for each term. The Laplace correction is also used in other rule induction algorithms such as CN2 (Clark et al., 1991). It generates rules that are specific and helpful in reducing over fitting.

$$\eta_{ij} = \frac{|term_{ij},k| + 1}{|term_{ij}| + no\_of\_classes} \qquad (2)$$

where $|term_{ij}, k|$ is the number of training cases having $term_{ij}$ and the current positive class k; $|term_{ij}|$ is the number of training cases having $term_{ij}$; and No_of_classes is the number of values in the class attribute's domain.

4.4 Pheromone initialization and maintenance

The initial pheromone levels for both nominal and continuous attributes must be set at the initialization step. In case of a nominal attribute, the pheromone level for each attribute value is initialized to 1/n, where n is the total number of attribute values for all of the attributes including the continuous attributes. Adding as many kernels as the number of ranges into which the continuous attribute is split initializes the mixed kernel PDF. Each kernel is initialized with the corresponding mean and standard deviation of the range. The pheromone level of a particular range is obtained by calculating the area under the mixed kernel PDF curve between the minimum and the maximum values of the range.

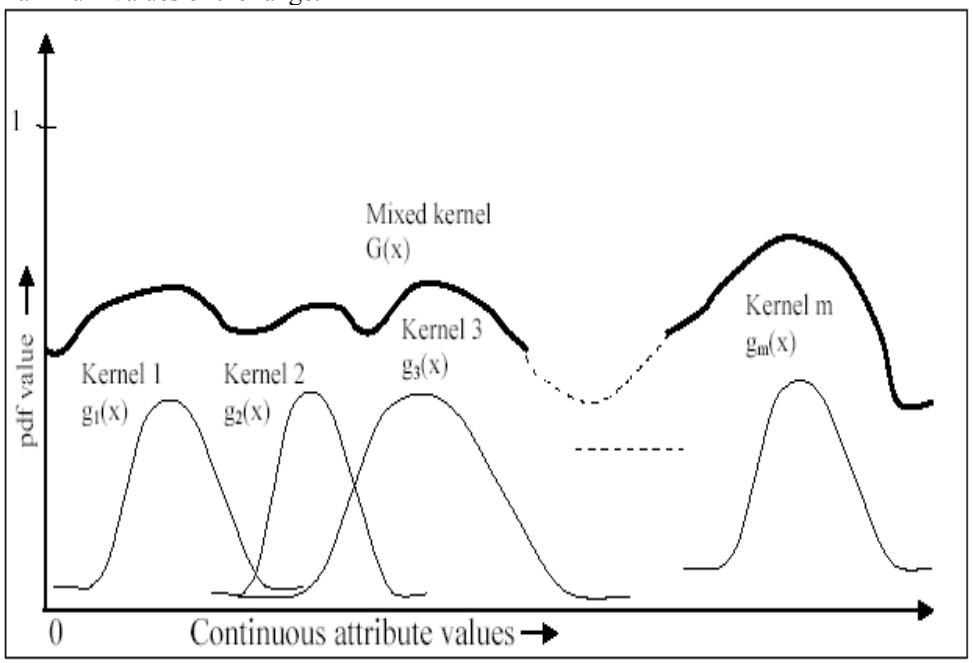

Figure 1. A mixed kernel PDF with m normal kernels

Figure 1 shows the different ranges for the continuous attribute. At the initial stage, the initial pheromone levels have to be the same for all attribute value pairs so that all the attribute value pairs have an equal probability of being added to the rule in the beginning. The continuous attributes, however, have different initial pheromone levels. Achieving a uniform distribution over the continuous domain is not possible (Monmarche et al., 2000), and hence the attributes are initialized to different values. With the use of the normalization constant 'k,' the initial pheromone values of the continuous ranges can be made equal to that of other attribute values (1/n). The normalization constant 'k' (Swaminathan, 2006) is given by

$$k_i = \frac{\sum_{x=l}^{h} G(X)}{1/n} \qquad (3)$$

where

k_i	=normalization constant for the ith range of a continuous attribute
n	= sum of attribute values for all attributes including continuous attributes
G(X)	= mixed kernel PDF for the continuous attribute
l	= the minimum value of the ith range
h	= the maximum value of the ith range

The normalization constant is calculated at the beginning of the run when the pheromone levels are initialized. The pheromone level for a range of the continuous attributes is given by

$$\tau_{ij} = \frac{\sum_{x=1}^{h} G(X)}{k_i} \quad (4)$$

where

τ_{ij} = pheromone level of the jth range of the continuous attribute at the ith position
k_j = normalization constant of the jth range
$G(X)$ = mixed kernel PDF for the continuous attribute
l = the minimum value of the ith range
h = the maximum value of the ith range

4.5 Pheromone Updating

The best rule of each iteration is stored and used to update the pheromone levels of the terms before the next iteration is initiated. In the Unordered Rule Set algorithm, the consequent (predicted class) of each rule is fixed during the construction of the rule by an ant. Because of the probabilistic nature of the algorithm, it is possible to generate rules where the number of true positives (TP) is less that the number of false positives (FP). Such rules tend to be bad rules because they have a low predictive accuracy. It is important that the pheromone of terms occurring in the rule be increased only when the just-constructed rule has an acceptable confidence value. The threshold that determines if a rule is acceptable or not (i.e., whether or not the pheromone of its terms should be increased) is expressed by the following formula.

$$\text{Rule Confidence Threshold} = \frac{\text{MAX}(0.5, |k|)}{|\text{training set}|} \quad (5)$$

where |k| is the number of training cases with the current (positive) class, and |training set| is the total number of cases in the current training set.

Once a rule has been considered acceptable, the amount of pheromone to be applied to each of the terms in that rule increases. In the case of nominal attributes, the pheromone level is increased by using the formula

$$\tau_{ij}(t+1) = \tau_{ij}(t) + (\tau_{ij}(t) * Q) \quad (6)$$

where $\tau_{ij}(t)$ is the current (at time index t) amount of pheromone associated with $term_{ij}$, and Q is a quality function is measured based on confidence value.

For continuous attributes, a new normal kernel is added to the mixed kernel PDF. The mean and standard deviation of the added kernel are the mean and standard deviation of the range that was selected by the ant during the iteration. The addition of a new kernel changes the shape of the mixed kernel PDF curve and thus changes the values of the pheromones for the neighboring ranges. The impact of this increase in pheromones on the pheromone value of a neighboring range decreases as the distance between the means of the added range and the neighboring range increases. This increase in pheromones of the neighboring ranges preserves the relationship represented by the continuous values, which was not preserved by the original Ant-Miner.

4.6 Rule Pruning

In Ant-Miner, rules were pruned to remove irrelevant terms and to improve the predictive accuracy of rules. This involved speculatively removing each term in turn, evaluating the quality of the rule without that term, and then analyzing the change in rule quality. This process was repeated until there was only one term left in the rule antecedent or no increase in rule quality was observed during the speculative removal process. A new

consequent – namely, the class with the largest frequency among all cases covered by the rule – was assigned to the rule after each term was speculatively removed.

However, in unordered rule set Ant-Miner, the consequent must remain the same during this process, and so the rule pruning procedure is simplified. After each term is speculatively removed, there is no need to compute the quality of the new reduced rule for all possible classes in the consequent, just the quality for the current positive class is computed.

4.7 Classifying Test Data with an Unordered Rule Set

In the original Ant-Miner algorithm, classifying test data with the ordered list of rules was accomplished by finding for each case the first rule in the ordered list that covered the case and then assigning the consequent class value of that rule to the case. A default rule that assigned the majority class in the training set to a case was used to classify a test case if none of the discovered rules matched the test case.

When classifying test data with the Unordered Rule Set Ant-Miner algorithm, a different approach is required, as a case might be covered by more than one rule. One of the following scenarios will occur when classifying a given test case with rules discovered by the Unordered Rule Set Ant Miner (Smaldon & Freitas, 2006):

1. If none of the discovered rules cover the test case, that case is assigned to the default class, which is the majority class in the training data set.
2. If only one of the discovered rules covers the test case, that case is assigned to the class predicted by that rule.
3. If more than one of the discovered rules covers the test case, but all those rules predict the same class, the case is assigned that class.
4. If more than one of the discovered rules covers the test case, but the rules do not all predict the same class, a rule conflict strategy is required to determine which class should be assigned to that case.

Two rule conflict strategies were evaluated:
1. Classifying the test case with the rule that has the highest rule quality.
2. Applying a rule conflict resolution procedure based on the class distribution of the rules covering the current test case to determine that case's class value (Clark et al., 1991).

The pseudocode for this procedure is shown in the following algorithm

Algorithm – Rule Conflict Resolution Procedure Based on the Class Distribution of the Rules Covering a Case

```
FOR EACH Class c
Count(c) = 0;
END FOR
FOR EACH Rule r covering the current test case
        FOR EACH Class c
        Count(c) = Count(c) + Coverage(r, c);
        END FOR
END FOR
```

The algorithm starts by assigning the class with maximum value of Count(c) among all candidate classes to the current test case. The first For-Each-Class loop of the algorithm initializes the class counts to zero. The For-Each-Rule loop iterates over all the rules covering the current test case, i.e., the rules whose conflict must be solved. The function Coverage(r, c) returns the number of training cases having class c covered by rule r. Hence, for each of the rules covering the current test case, the algorithm adds to each class count the number of training cases covered by the current rule. Therefore, at the end of the For-Each-Rule loop, each class count will contain the frequency of the corresponding class in the total class distribution associated with all rules covering the current test case. Finally, the test case is assigned the class with the largest class count value, i.e., the most frequent class in the total class distribution associated with all conflicting rules.

5 COMPUTATIONAL RESULTS AND DISCUSSION

Data sets

The performance of the algorithm was evaluated using six public domain data sets from the UCI repository (www.ics.uci.edu/~mlearn/MLRepository.html). The main characteristics of the data sets are summarized in Table 1. The algorithm has four user-defined parameters as defined in Table 2. We have evaluated the performance of the algorithm with the existing Ordered and Unordered Rule Set Ant-Miner algorithms. The comparison was carried out across two criteria, namely the predictive accuracy of the discovered rule lists and their simplicity. We considered the original Ant-Miner and Unordered Rule Set Ant-Miner values as benchmark values. We implemented Order Set Continuous Ant-Miner and Unordered Set Continuous Ant-Miner in Java. A new class called C45Disc was created to find the split points of a continuous attribute. For the purpose of comparing the performances of Continuous Ant-Miner and Ant-Miner, all the data sets on which Ant-Miner was originally tested were used.

Table 1. Dataset Characteristics

Data Set	Total No. of Examples	Nominal Attributes	Continuous Attributes	Classes
Ljubljana breast cancer	286	9	-	2
Wisconsin breast cancer	699	-	9	2
Tic-tac-toe	958	9	-	2
Hepatitis	155	13	6	2
Dermatology	366	33	1	6
Cleveland heart disease	303	8	5	5

Parameter settings

Both the original Ant-Miner algorithm and the proposed Unordered Rule Set Continuous Ant-Miner have four parameters:
1. Number of ants (No_of_ants).
2. Minimum number of cases per rule (Min_cases_per_rule).
3. Maximum number of uncovered cases in the training set (Max_uncovered_cases).
4. Number of rules used to test convergence of the ants (No_rules_converg).

Table 2. Parameter settings

Parameter	Original Ant-Miner	Unordered Rule Set Ant-Miner
No_of_ants	3000	3000
Min_cases_per_rule	5	5
Max_uncovered_cases	10	5
No_rules_converg	5	5

In Unordered Rule Set Ant-Miner, *Max_uncovered_cases* refers to the maximum number of uncovered positive cases in the training set, while in the original Ant-Miner it refers to the maximum number of cases (either positive or negative ones) in the training set. For this reason, this parameter may need to be set lower in unordered rule set Ant-Miner than in the original Ant-Miner.

Predictive accuracy

The results comparing predictive accuracy of Ant-Miner and other algorithms are reported in Table 3. We measure the predictive accuracy by using the formula:

Predictive accuracy= Number of test cases correctly classified by the rule set/ Total number of test cases

Ant-Miner and Continuous Ant-Miner use entropy value to select attributes (Parpinelli et al., 2002, Swaminathan, 2006). However, Unordered Rule Set Ant-Miner and Unordered Rule Set Continuous Ant-Miner use Laplace measure to select the attributes. Because Laplace corrected confidence measure, Clark et al., (1991) is biased towards more general rules with higher predictive accuracy than entropy and used to avoid over fitting.

The previous implementations (i.e., Ant-Miner and Continuous Ant-Miner) evaluate the quality of the rule constructed by an ant based on sensitivity and specificity, Q= sensitivity * specificity (FP + TN). However, in the unordered rule set continuous Ant-Miner, we measure the quality of the rule based on the rule confidence value. Therefore, this algorithm generates more accurate rules than Ant-Miner.

Table 3. Comparison of the predictive accuracy of rules generated using Unordered Continuous Ant-Miner with previous implementations of Ant-Miner. The numbers after the "±"symbol are the standard deviations of the corresponding average predictive accuracies.

Dataset	Ant-Miner Ordered Rule Set	Continuous Ant-Miner Ordered Rule Set	Ant-Miner Unordered Rule Set	Continuous Ant-Miner Unordered Rule Set
Ljubljana breast cancer	75.28+/-2.24	75.91 +/-8.38	78.42 +/- 1.70	79.87 +/-1.85
Wisconsin breast cancer	96.04+/-0.93	96.84 +/-2.13	92.38 +/- 0.84	96.96 +/- 0.89
Tic-tac-toe	73.04+/-2.53	73.52 +/-2.81	72.45 +/- 0.87	73.78 +/- 1.20
Dermatology	94.29+/-1.20	90.94 +/-4.75	80.50 +/- 1.56	96.84 +/- 1.56
Hepatitis	90.00+/-3.11	93.16 +/-10.33	95.42 +/- 2.50	96.15+/-2.90
Cleveland heart disease	59.67+/-2.50	73.93 +/-8.75	64.84 +/- 2.60	84.76 +/- 2.67

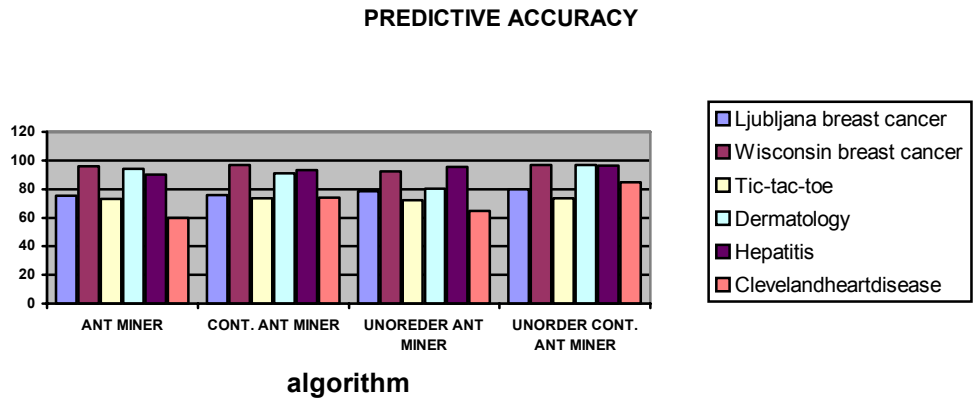

Figure 2. Predictive accuracy

The accuracy of the Unordered Rule Set Continuous Ant-Miner is highly comparable to the accuracy of the Ordered Continuous Ant-Miner algorithm in most cases. As shown in these tables, the Unordered Rule Set Continuous Ant-Miner discovers rules with a better predictive accuracy than the previous algorithm in four data sets, namely Ljubljana breast cancer, Wisconsin breast cancer, Hepatitis, and Cleveland heart disease. Also those data sets contained mixed attributes. The only dataset in which there was no improvement associated with the Unordered Rule Set versions of Ant-Miner was Tic-tac-toe. We conclude that the Unordered Rule Set Continuous Ant-Miner is greater than the original Ant-Miner, and the heuristic and quality functions play an important role.

Simplicity

We now turn to the results concerning simplicity of the discovered rule list, which is measured by the number of discovered rules and the average number of terms per rule. The results comparing the simplicity of the rule lists discovered by the Unordered Rule Set Continuous Ant-Miner are reported in Tables 4 and 5 and Figures 3 and 4.

Table 4. Comparison of the number of rules generated using Unordered Continuous Ant-Miner with previous implementations of Ant-Miner.

Dataset	Ant-Miner Ordered Rule Set	Continuous Ant-Miner Ordered Rule Set	Ant-Miner Unordered Rule Set	Continuous Ant-Miner Unordered Rule Set
Ljubljana breast cancer	7.10+/-0.31	8.10 +/- 0.99	6.10+/-0.11	7.75 +/-0.42
Wisconsin breast cancer	6.20+/-0.25	7.80 +/- 0.63	6.50 +/- 0.19	7.38 +/-0.44
Tic-tac-toe	8.50+/-0.62	7.60 +/- 1.89	6.30 +/- 0.17	6.25 +/-0.79
Dermatology	7.30+/-0.15	7.40 +/- 0.69	6.00 +/- 0.0	6.45 +/- 0.4
Hepatitis	3.40+/-0.16	5.40 +/- 0.52	3.00 +/- 0.0	4.90 +/-0.3
Cleveland heart disease	9.50+/-0.92	7.20 +/- 0.63	11.00 +/- 0.0	8.25 +/-0.6

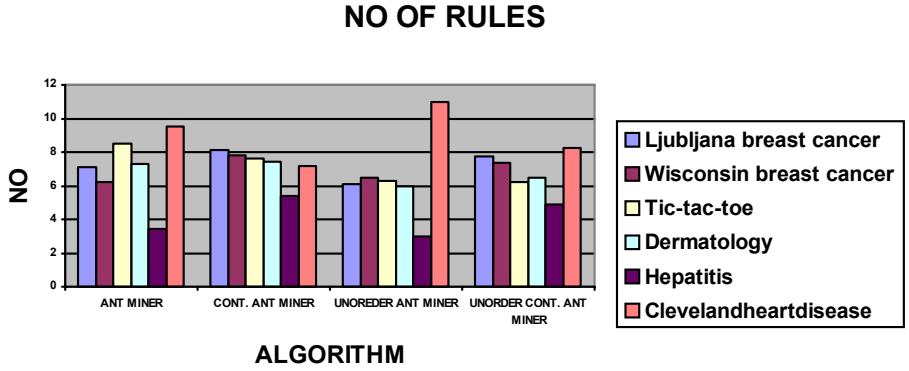

Figure 3. Number of rules generated

The following rule set was generated as one of the 10 fold cross validation procedure using the above parameter setting for the Hepatitis data set

Attribute information:

1. Class: DIE, LIVE
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
3. SEX: male, female
4. STEROID: no, yes
5. ANTIVIRALS: no, yes
6. FATIGUE: no, yes
7. MALAISE: no, yes
8. ANOREXIA: no, yes
9. LIVER BIG: no, yes
10. LIVER FIRM: no, yes
11. SPLEEN PALPABLE: no, yes
12. SPIDERS: no, yes
13. ASCITES: no, yes
14. VARICES: no, yes
15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
17. SGOT: 13, 100, 200, 300, 400, 500,
18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
20. HISTOLOGY: no, yes

Class Distribution:

1.DIE: 32

2.LIVE: 123
NUMBER OF Nominal attributes :13
Number of continuous attributes:6

Rule set:
IF< VARICES =2>AND < SPIDERS=2> AND < ASCITES=2> THEN class=2
IF< MALAISE=1>AND < HISTOLOGY=2>THEN class=1
IF<AGE <50> AND <LIVER FIRM =2> THEN class= 2

Table 5. Comparison of the number of terms per rule generated by Unordered Continuous Ant-Miner with previous implementations of Ant-Miner

Dataset	Ant-Miner Ordered Rule Set	Continuous Ant-Miner Ordered Rule Set	Ant-Miner Unordered Rule Set	Continuous Ant-Miner Unordered Rule Set
Ljubljana breast cancer	1.28	1.06	1.85	1.35
Wisconsin breast cancer	1.97	1.09	2.55	1.35
Tic-tac-toe	1.18	1.17	1.10	1.15
Dermatology	3.16	2.93	3.10	2.91
Hepatitis	2.41	1.96	3.33	2.25
Cleveland heart disease	1.71	1.33	2.54	2.32

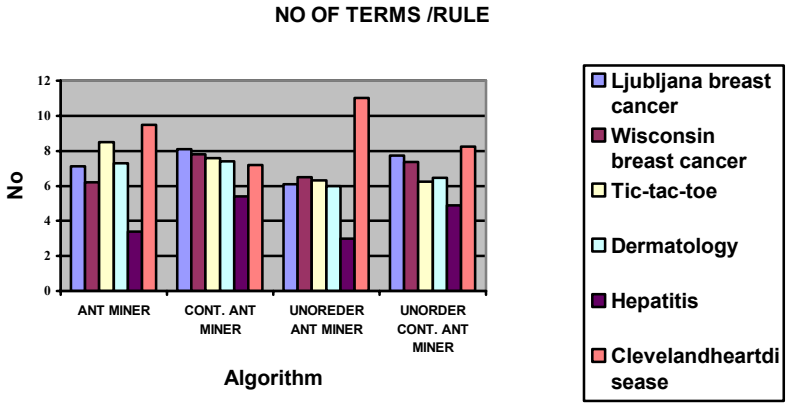

Figure 4. Number of terms/rule generated

The Unordered Rule Set Continuous Ant-Miner algorithm discovers more modular rules, i.e., rules that can be interpreted independently from other rules – unlike the rules in an ordered list, where the interpretation of a rule requires knowledge of the previous rules in the list, than previous algorithms. The experiments show the Unordered Rule Set Continuous Ant-Miner accuracy is also superior to that obtained by other Ant-Miner versions. In the unordered rule list, we fix the consequent value at the initial stage, so rule pruning is more accurate and simple than in the Ordered Rule Set Ant-Miner versions. The main obstacle faced during this research was to normalize the pheromone values of the continuous attributes and nominal attributes, as they are in two different domains.

6 CONCLUSIONS AND FUTURE RESEARCH

This work has proposed an algorithm to discover unordered rule sets. The main goal of this algorithm is to discover classification rules in data sets. We have compared the performance of the proposed algorithm with existing algorithms. In existing algorithms, the numerical values are discretized into nominal attributes before the algorithm is executed. However, in the proposed method, there is no discretization process. We use multimodal functions (PDF) to handle continuous attributes. Compared to previous implementations of Ant-Miner, the implementations in this paper reduce the potential information loss resulting from the discretization

process of continuous attributes. Continuous Ant-Miner is designed and implemented to induce classification rules from a training dataset with continuous attributes and nominal attributes.

Our experiments have shown that the proposed Unordered Rule Set Continuous Ant-Miner algorithm is capable of discovering rules that are comparable to those discovered by the original Ant-Miner algorithm, in terms of both predictive accuracy and rule set simplicity. The rules discovered by the Unordered Rule Set Ant-Miner are more modular than the rules discovered by the original Ordered List Ant-Miner because in the Unordered Rule Set Anti-Miner algorithm, each rule can be interpreted independently from the others. In contrast, in the rule list discovered by the original Ant-Miner, a given rule can be interpreted only in the context of all the previous rules in the list. This modularity facilitates the interpretation of the rules by the user, an important point in data mining. The main obstacle faced during this research was to normalize the pheromone values of the continuous attributes and nominal attributes because they are in two different domains.

Two important directions for future research are as follows: first, a study is needed on how to optimize the user defined parameter values. Second, Particle Swarm Intelligence (PSI) techniques should be examined to find the appropriate ranges for continuous attributes in datasets.

7 REFERENCES

Bilchev, G. & Parmee, I. C. (1995) The Ant Colony Metaphor for Searching Continuous Design Spaces. In Fogarty, T. C. (ed.), *Proceedings of the AISB Workshop on Evolutionary Computation, volume 993 of LNCS* (pp 25-39), Berlin, Germany.

Bonabeau,.E., Dorigo, M., & Theraulaz, G. (1999) *Swarm Intelligence from Natural to Artificial Systems*. New York: Oxford University Press.

Clark, P. & Boswell, R. (1991) Rule induction with CN2: some recent improvements. *Proc. European Working Session on Learning (EWSL-91), LNAI 482,* pp. 151-163.

Dreo, J. & Siarry, P. (2002) A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. *Proceedings of ANTS 2002*.

Dorigo.M, Colorni, A., & Maniezzo, V. (1996), The Ant System: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol.26, no. 1*, pp. 29-41.

Kohavi, R. (1995) A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 1137-1143), California.

Monmarche, N., Venturini, G., & Slimane, M. (2000) On how pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems*, vol. 16, pp. 937–946.

Parpinelli, R., Lopes, H., & Freitas, A. (2002) A Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computing 6(4)*, pp. 321–332.

Quinlan, J. (1993) *C4.5: Programs for Machine Learning.* Los Altos, California: Morgan Kauffman.

Smaldon, J. & Freitas , A. (2006) A New Version of the Ant-Miner Algorithm Discovering Unordered Rule Sets., *GECCO'06*, Seattle, Washington, USA.

Socha, K. (2004) ACO for Continuous and Mixed-Variable Optimization. *Proceedings of the Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence,* Brussels, Belgium.

Swaminathan, S. (2006) *Rule Induction Using Ant Colony Optimization for Mixed Variable Attributes,* M.S Thesis, Texas Tech University, Lubbock, TX.