A New Version of the Ant-Miner Algorithm Discovering Unordered Rule Sets

James Smaldon
University of Kent
Computing Laboratory
Canterbury, CT2 7NF, UK
James.Smaldon@gmail.com

Alex A. Freitas
University of Kent
Computing Laboratory
Canterbury, CT2 7NF, UK
A.A.Freitas@kent.ac.uk

ABSTRACT
The Ant-Miner algorithm, first proposed by Parpinelli and colleagues, applies an ant colony optimization heuristic to the classification task of data mining to discover an ordered list of classification rules. In this paper we present a new version of the Ant-Miner algorithm, which we call Unordered Rule Set Ant-Miner, that produces an unordered set of classification rules. The proposed version was evaluated against the original Ant-Miner algorithm in six public-domain datasets and was found to produce comparable results in terms of predictive accuracy. However, the proposed version 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. Hence, the proposed version facilitates the interpretation of discovered knowledge, an important point in data mining.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning – *concept learning*, *induction*.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Ant Colony Optimization, Data Mining, Classification Rules.

1. INTRODUCTION
Data Mining is the process of extracting useful knowledge from real-world data. Among the several data mining tasks – such as clustering and classification - this paper focuses on classification. In this task the aim is to discover, from training data (containing cases, or records, whose class is known), a classification model that can be used to predict the class of cases in the test data (containing unknown-class cases). One popular category of classification model consists of classification rules, which is the model category used in this paper. In this context, the aim of the classification algorithm is to discover a set of classification rules.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

One algorithm for solving this task is Ant-Miner, proposed by Parpinelli and colleagues [5], which employs ant colony optimization techniques [1] to discover classification rules of the form:

IF ($term_1$ AND $term_2$ AND ….. $term_m$) THEN (predicted class)

where each term is of the form <attribute = value>, and different rules can have different number of terms in their antecedent (IF part). The consequent of a rule is a predicted class, i.e., the value that the rule predicts for the class attribute when an example satisfies the conjunction of terms in the rule antecedent.

Classification rules have the advantage of representing knowledge at a high level of abstraction, so that they are intuitively comprehensible to the user [7].

Ant-Miner has produced good results when compared with more conventional data mining algorithms [5], [8] and it is still a relatively recent algorithm, which motivates further research trying to improve it. This work proposes a modification to the Ant-Miner data mining algorithm called Unordered Rule Set Ant-Miner, with the aim of improving or at least maintaining the level of predictive accuracy obtained by the original Ant-Miner, whilst at the same time facilitating the interpretation of the discovered classification rules, as follows. In the original Ant-Miner, the goal of the 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, since their conditions make sense only in the context of all the previous rules in the ordered list of rules [7]. The new version of Ant-Miner proposed in this paper discovers, from training data, an unordered set of rules that can be applied to test data in any order. This makes the discovered rules easier for the user to interpret, since now the interpretation of each rule is independent from all the other discovered rules.

Although some modifications to the Ant-Miner algorithm have already been proposed [2][3][4], to the best of our knowledge, an unordered rule set modification to the original Ant-Miner algorithm is an area of research that has not yet been explored.

This paper is organised as follows. Section 2 presents an outline of the original Ant-Miner algorithm. Section 3 explains the proposed Unordered Rule Set Ant-Miner. Section 4 discusses computational results and performance of the algorithm. Section 5 concludes the paper and suggests further areas of research.

2. A BRIEF DESCRIPTION OF THE ANT-MINER ALGORITHM
The original Ant-Miner algorithm, upon which the Unordered Rule Set Ant-Miner proposed in this paper is based, is described in the

pseudo code of Algorithm 1, taken from [5]. We provide here just a brief overview of the algorithm; for more details the reader is referred to that reference.

Algorithm 1 – Original Ant-Miner

TrainingSet = {all training cases};
DiscoveredRuleList = [ ]; /* initialize rule list with empty list */
WHILE (TrainingSet > Max_uncovered_cases)
    t = 1; /* ant index, and also rule index */
    j = 1; /* convergence test index */
    Initialize all trails with the same amount of pheromone;
    REPEAT
        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; /* remove irrelevant terms from rule */
        Update the pheromone of all trails by increasing pheromone in the trail followed by Ant_t (proportional to the quality of R_t) and decreasing pheromone in the other trails (simulating pheromone evaporation);
        IF (R_t is equal to R_t – 1) /* update convergence test */
            THEN j = j + 1;
            ELSE j = 1;
        END IF
        t = t + 1;
    UNTIL (i ≥ No_of_ants) OR (j ≥ No_rules_converg)
    Choose the best rule R_{best} among all rules R_t constructed by all the ants;
    Add rule R_{best} to DiscoveredRuleList;
    TrainingSet = TrainingSet - {set of cases correctly covered by R_{best}};
END WHILE

Ant-Miner discovers an ordered list of classification rules based on a heuristic function involving information gain – a popular heuristic function in data mining [6] – and positive feedback involving artificial pheromone. For each iteration of the Repeat-Until loop, an ant attempts to discover a rule by selecting terms in a probabilistic manner, until all the attributes have been used to make the current rule, or adding any other available term would make the rule coverage less than *min_cases_per_rule* – a user-specified threshold. The discovered rule is then pruned in an attempt to reduce over-fitting to the training data and increase rule quality. Afterwards, the pheromone values for the terms in the current rule are increased, in order to increase the probability that other ants will select those terms, and then the pheromone values for all terms are normalised. The While loop iterates until the number of training examples remaining in the dataset becomes less than or equal to *Max_uncovered_cases* – another user-specified threshold. The rule discovered in the Repeat-Until loop that has the highest quality is then added to the list of discovered rules, and the training examples correctly covered by that rule are removed from the training dataset. An example is correctly covered by a rule if the example satisfies the rule antecedent and has the class predicted by the rule.

2.1 Pheromone Initialisation

Pheromone values for each term are all initialised to the same value at the beginning of each While loop iteration. The initial value of each pheromone is given by the function:

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

Where a is the total number of attributes, i is the index of an attribute, j is the index of a value in the domain of attribute i, and b_i is the number of values in the domain of attribute i.

2.2 Pheromone Updating

In Ant-Miner pheromone levels are increased for all terms in a rule just constructed by an ant, based on the quality of that rule, as measured by the rule quality formula "sensitivity * specificity", defined as follows:

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

where TP / (TP + FN) is the sensitivity, TN / (FP + TN) is the specificity, and:

TP (true positives) is the number of cases covered by the rule that have the class predicted by the rule.

FP (false positives) is the number of cases covered by the rule that have a class different from the class predicted by the rule.

FN (false negatives) is the number of cases that are not covered by the rule but that have the class predicted by the rule.

TN (true negatives) is the number of cases that are not covered by the rule and that do not have the class predicted by the rule.

2.3 Term Selection

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

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

where:

η_{ij} is the value of a problem-dependent heuristic function – more precisely information gain [6] – for $term_{ij}$ (a condition of the form $attribute_i = value_j$). The higher the value of η_{ij} the more relevant for classification the $term_{ij}$ is, and so the higher its probability of being chosen.

$\tau_{ij}(t)$ is the amount of pheromone associated with $term_{ij}$ at iteration t.

a is the total number of attributes.

x_i is set to 1 if the attribute a_i was not yet used by the current ant, 0 otherwise.

b_i is the number of values in domain of the ith attribute.

3. UNORDERED RULE SET MODIFICATIONS TO THE ANT-MINER ALGORITHM

As mentioned in the Introduction, we propose a modification to the original Ant-Miner so that the algorithm discovers a set of rules which do not need to be applied to test data in the order in which they were discovered. The pseudocode of the new algorithm is described in Algorithm 2.

In the original Ant-Miner, ants chose 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 was then assigned afterwards by determining the class value that would produce the highest quality rule. In Unordered Rule Set Ant-Miner, by contrast, an extra For-Each loop is added as the outer loop of the algorithm, iterating over the values in the class attribute domain, as indicated in Algorithm 2. As a result of this loop, the consequent for the rule is known by the ant during rule construction and does not change. The current ant tries to choose terms that will produce the rule predicting the class value in the current iteration of the For-Each loop with an optimum level of accuracy. In theory, such an approach should lead to faster convergence on good rules, by comparison with the original Ant-Miner. The reason 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.

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.

Rule construction occurs as follows. 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. However, unlike the original Ant-Miner, in the proposed algorithm an ant starts with a rule containing the known consequent (the current class value of the FOR loop) and an empty antecedent. The rule is constructed incrementally by selecting terms with a probabilistic method that favours terms with a large amount of pheromone and a high Laplace-corrected confidence value (see below). The ant stops constructing a rule if all the attributes have been used in the rule antecedent 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, and 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.

Algorithm 2 – Proposed Unordered Rule Set Ant-Miner

```
Discovered Rule Set = {} /* initialize rule set with empty set */
FOR EACH Class
    TrainingSet = {all training cases}
    PositiveSet = {training cases of current class}
    NegativeSet = TrainingSet – PositiveSet
    WHILE (|PositiveSet| > max_uncovered_cases)
        t = 1;
        j = 1;
        initialise all trails to the same amount of pheromone;
        REPEAT
            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
            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 (i ≥ No_of_ants) OR (j ≥ No_rules_converg)
        Choose the best rule R_best among all rules R_t constructed
        by all ants;
        Add rule R_best to DiscoveredRuleSet;
        TrainingSet = TrainingSet – {set of positive cases
                                     covered by R_best };
        PositiveSet = PositiveSet – {set of positive cases
                                     covered by R_best};
    END WHILE
END FOR
```

3.1 Problem dependent heuristic function

Since the consequent of the rule is already known when the ants discover rules, the heuristic function of the original Ant-Miner must be altered to favour the selection of terms that increase the probability that the rule will predict the current class of the For-Each

loop. The problem dependent heuristic function chosen is the Laplace-corrected confidence for each term, given by:

$$\eta_{ij} = \frac{|term_{ij}, k| + 1}{|term_{ij}| + \text{No\_of\_classes}}$$

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. The Laplace correction is also used in other rule induction algorithms such as CN2 [9], and it has the advantage of penalizing rules that are too specific (covering too few cases), helping to reduce overfitting. For instance, suppose a terms occurs in just one case, and that case has the current positive class, so that $|term_{ij}| = |term_{ij}, k| = 1$. Without the Laplace correction, the confidence of that term would be 1 / 1 = 100%, a too optimistic value for such an extremely specific rule, which is unlikely to generalize well for test data unseen during training. With the Laplace correction, and supposing *No_of_classes* = 2, the confidence of the rule is corrected to (1 + 1) / (1 + 2) = 67%, a more realistic value. Note that the Laplace correction will have little effect when $|term_{ij}|$ is large, which is consistent with the fact that there is a lot of statistical support for such a generic rule.

3.2 Pheromone Updating

As mentioned earlier, in the unordered rule set algorithm the consequent (predicted class) of each rule is fixed during the construction of the rule by an ant. Due to 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.

$$RuleConfidenceThreshold = \text{MAX}(0.5, \frac{|k|}{|\text{training set}|})$$

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.

The rule confidence threshold is therefore the maximum of the relative frequency of the predicted class and 0.5. The motivation for the use of the max operator in this formula is as follows. In the case of a predicted class whose relative frequency is "large" (greater than 0.5), the rule is considered acceptable only if its confidence is at least as great as the relative frequency of that class. For instance, if the relative frequency of a class is 70%, a rule predicting that class with a confidence of 60% is clearly a bad rule. This requirement is not enough, however, when the predicted class has a "small" relative frequency (lower than 50%). For instance, suppose a class has a relatively frequency of 10%. A rule with a confidence of 15% satisfies the criterion of having a confidence greater than the relative frequency of the predicted class, but it is still a weak rule. Hence, the use of the max operator guarantees that, when the predicted class has a low relative frequency, the confidence threshold is raised to 50%. We make no claim that the threshold value 0.5 is an optimal value, but this value worked well in our preliminary experiments. Optimizing this parameter is a topic left for future research.

Once a rule has been considered acceptable, the amount of pheromone increase to be applied to each of the terms in that rule is determined by the following formula.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + (\tau_{ij}(t) \cdot \delta)$$

where $\tau_{ij}(t)$ is the current (at time index t) amount of pheromone associated with $term_{ij}$, and δ is set to the rule quality Q (the formula in section 2.2, which is also the formula used in the original Ant-Miner) if the Laplace-corrected confidence for the rule was above *RuleConfidenceThreshold* or set to 0 otherwise. The basic idea of the Laplace-correction was already explained in section 3.1, in the context of the problem-dependent heuristic function. A conceptually similar idea is used in the context of the confidence of a rule. A rule's confidence (before the use of Laplace correction) is defined as follows. Let IF A THEN C be a rule, where A is the antecedent (conjunction of terms) and C be the predicted class. The confidence of a rule is given by:

$$|\text{A and C}| / |\text{A}|$$

i.e., the number of training cases satisfying both A and C divided by the total number of cases satisfying A. The Laplace-corrected confidence is then given by:

$$(|\text{A and C}| + 1) / (|\text{A}| + \text{No\_of\_classes}) .$$

The effect of the Laplace correction in the confidence of a rule is conceptually similar to the effect of this correction in the value of the problem-dependent function, as explained in section 3.1.

3.3 Rule Pruning

It is necessary to make some alterations to the rule pruning in the Ant-Miner algorithm to support unordered rule sets. In the original 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 and evaluating the quality of the rule without that term; and then definitely removing the term whose removal provided the largest increase in rule quality. This process was then 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. For 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.

3.4 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 (i.e. the case's attribute values matched the rule antecedent), 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 Unordered Rule Set Ant-Miner, 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.

1. If none of the discovered rules cover the test case, that case is assigned 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 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 in this work:

1. Classify the test case with the rule that has the highest rule quality.
2. Apply 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 [9]. The pseudocode for this procedure is shown in Algorithm 3.

Algorithm 3 –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
```

Assign to the current test case the class with maximum value of Count(*c*), among all candidate classes.

The first For-Each-Class loop of Algorithm 3 initializes the class counts to zero. The For-Each-Rule loop up 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 value of class count, i.e., the most frequent class in the total class distribution associated with all conflicting rules.

4. COMPUTATIONAL RESULTS

4.1 Datasets Used in the Experiments

The performance of the proposed Unordered Rule Set Ant-Miner was evaluated using six public-domain data sets from the UCI (University of California at Irvine) data set repository – available from: http://www.ics.uci.edu/~mlearn/MLRepository.html. Table 1 shows the main characteristics of the datasets, which were the same datasets used to evaluate the original Ant-Miner in [5].

Note that Ant-Miner (both the original one and the unordered rule set version proposed in this paper) cannot cope directly with continuous attributes, i.e., continuous attributes have to be discretized in a preprocessing step. For the datasets having continuous attributes in Table 1, we used the same discretized version of the data used in the experiments with the original Ant-Miner reported in [5]. Those discretized datasets were kindly provided by Parpinelli.

Table 1 – Dataset Characteristics

DataSet	Number of examples	Number of categorical attributes	Number of continuous attributes	Number of classes
Ljubljana breast cancer	282	9	0	2
Wisconsin breast cancer	683	0	9	2
Cleveland heart disease	303	8	5	5
Dermatology	366	33	1	6
Hepatitis	155	13	6	2
Tic-tac-toe	958	9	0	2

4.2 Comparison of Results

Both the original Ant-Miner algorithm and the proposed Unordered Rule Set 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).

As explained earlier, in Unordered Rule Set Ant-Miner, Max_uncovered_cases refers to the maximum number of uncovered *positive* cases in the training set, whilst 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. The value of 5 (half the value used for the original Ant-miner) was used in this work as a reasonable value. The other parameters of Unordered Rule Set Ant-Miner were set to the same values as in the original Ant-Miner, since those other parameters have the same meaning in both versions of the algorithm. We make no claim that these are optimal parameter values, and finding the

optimum values for Unordered Rule Set Ant-Miner's parameters is an area requiring further research. (In any case, the optimum value for a parameter tends to be strongly problem-dependent, as usual in bio-inspired algorithms.) Table 2 shows the parameter settings used when testing both versions of Ant-Miner.

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

Ten-fold cross validation [7] was performed on each of the datasets with the following versions of the Ant-Miner algorithm:

1. Original, Ordered Rule List Ant-Miner.
2. Unordered Rule Set Ant-Miner using highest-quality rule conflict resolution strategy.
3. Unordered Rule Set Ant-Miner using class-distribution rule conflict resolution strategy.

Table 3 shows a comparison of the mean classification accuracy (%) on the test set and corresponding standard deviation of the rule sets discovered by each version of Ant-Miner during the cross validation procedure. In the last two columns of that table, the presence of the symbol (+) or (-) in a cell indicates that the predictive accuracy of the corresponding version of Unordered Rule Set Ant-Miner is significantly better or worse than the predictive accuracy of the original Ordered List Ant-Miner in the corresponding dataset. A difference in predictive accuracy is considered significant if the standard deviation intervals (containing plus or minus one standard deviation around the mean) of the two accuracies do not overlap.

Table 3 – Mean accuracy (%) of discovered rules

DataSet	Ordered List	Unordered Rules Class Distribution	Unordered Rules Highest Quality
Ljubljana breast cancer	72.98 +/- 1.97	78.42 +/- 1.70 (+)	60.31 +/- 3.52 (-)
Wisconsin breast cancer	95.91 +/- 0.48	92.38 +/- 0.84 (-)	95.61 +/- 0.92
Cleveland heart disease	57.20 +/- 1.77	64.84 +/- 2.60 (+)	56.38 +/- 1.49
Dermatology	92.74 +/- 1.38	80.50 +/- 1.56 (-)	96.05 +/- 1.46 (+)
Hepatitis	88.81 +/- 2.94	95.42 +/- 2.50 (+)	96.35 +/- 1.88 (+)
Tic-tac-toe	72.24 +/- 1.24	72.45 +/- 0.87	64.39 +/- 1.57 (-)

The results in Table 3 show that the accuracy of both versions of Unordered Rule Set Ant-Miner is comparable to the accuracy of the original Ant-Miner algorithm in most cases. The largest gain in accuracy occurred in the Hepatitis dataset, where both versions of Unordered Rule Set Ant-Miner obtained a significantly higher accuracy than the original Ordered Rule List Ant-Miner. The only datasets in which there was no improvement associated with the Unordered Rule Set versions of Ant-Miner were the Tic-tac-toe and Wisconsin breast cancer sets.

Table 3 also shows that the rule conflict resolution strategy used by Unordered Rule Set Ant-Miner when classifying test data is very important. The difference in accuracy between the Highest Rule Quality-based and Class Distribution-based strategies is almost 20% for the Ljubljana Cancer dataset, and the Dermatology and Cleveland HD datasets also show large discrepancies between the results for the two strategies. Overall, Class Distribution-based rule conflict resolution strategy slightly outperformed Highest Rule Quality-based rule conflict resolution. In particular, the Class Distribution-based strategy obtained a predictive accuracy significantly better (worse) than the original Ant-Miner in 3 (2) datasets; whilst the Highest Quality-based strategy obtained a predictive accuracy significantly better (worse) than the original Ant-Miner in 2 (2) datasets.

One could argue that, although the Class Distribution-based rule conflict resolution strategy slightly outperformed the Highest Rule Quality-based rule conflict resolution strategy with respect to predictive accuracy, the former hinders the interpretability of the discovered rules, because it involves combining the predictions of several rules, rather than just using the prediction of an individual rule. This is to some extent a concern, but there is a reply to this argument. The reply is that the interpretation of the rules by the user is, conceptually, independent of whether or not the rules are combined for making a prediction about the class of a specific example in the test. The rules were discovered by following a sequential covering strategy, consisting of discovering one rule at a time. Due to the details of the procedure used for discovering the rules (in the *Unordered* Rule Set version of Ant-Miner), each rule does have a modular meaning independent of the others – regardless of how the rule is used to classify test examples. In addition, note that it is not practical to ask the user to interpret at the same time the set of all rules used to classify a test example because this set varies from example to example. I.e., a given rule can sometimes be the only rule used to classify a certain test example (if the rule is the only to cover that example), whilst in other occasions the same rule might be combined with other rules to classify another test example (if there are two or more rules of different classes covering the same test example). The main motivation for showing the rules to the user is to try to give the user more insight about the data and the application domain. Users can potentially get such insight by interpreting the rules individually, one rule at a time, even when using the Class Distribution-based rule conflict resolution strategy.

We now consider the simplicity of the rules discovered by the Unordered Rule Set Ant-Miner. As usual in the data mining literature, simplicity is measured by the number of rules and the number of terms per rule – the smaller these values, the simpler the rule set is. The original Ant-Miner was competitive with CN2 and C4.5 with respect to accuracy, while producing much more simple rules [5], [8]. This is desirable as discovered rule sets that are more simple are easier to interpret and understand, and are potentially less likely to over fit the training data. A comparison of the mean number of rules discovered by the different versions of Ant-Miner is shown in *Table 4* and the mean number of terms per rule is shown in *Table 5*. The values after the mean are standard deviations. In *Table 4* in the last two columns, the presence of the symbol (+) or (-) in a cell indicates the mean number of rules discovered by the corresponding version of Unordered Rule Set Ant-Miner is significantly greater or smaller than the mean number of rules discovered by the Ordered Rule List Ant-Miner in the corresponding dataset. In *Table 5* in the last two columns, the

presence of the symbols (+) or (-) in a cell indicates the number of terms in the rules discovered by the corresponding version of Unordered Rule Set Ant-Miner is significantly greater or smaller that the number of terms in the rules discovered by the Ordered Rule List Ant-Miner in the corresponding dataset. As in *Table 3*, a difference between two results is considered significant if the standard deviation intervals do not overlap.

Table 4 – Mean number of rules discovered without counting the default rule

DataSet	Ordered List	Unordered Rules Class Distribution	Unordered Rules Highest Quality
Ljubljana breast cancer	6.70 +/- 0.37	6.10 +/- 0.11 (-)	6.00 +/- 0.00 (-)
Wisconsin breast cancer	5.60 +/- 0.30	6.50 +/- 0.19 (+)	6.20 +/- 0.15 (+)
Cleveland heart disease	14.20 +/- 0.43	11.00 +/- 0.00 (-)	11.00 +/- 0.00 (-)
Dermatology	6.00 +/- 0.00	6.00 +/- 0.00	6.00 +/- 0.00
Hepatitis	2.70 +/- 0.17	3.00 +/- 0.00 (+)	3.00 +/- 0.00 (+)
Tic-tac-toe	4.60 +/- 0.48	6.30 +/- 0.17 (+)	6.30 +/- 0.17 (+)

Table 5 – Average No. of terms per rule

DataSet	Ordered List	Unordered Rules Class Distribution	Unordered Rules Highest Quality
Ljubljana breast cancer	1.79 +/- 0.08	1.85 +/- 0.02	1.83 +/- 0.00
Wisconsin breast cancer	2.27 +/- 0.09	2.55 +/- 0.08 (+)	2.37 +/- 0.08
Cleveland heart disease	2.69 +/- 0.10	2.54 +/- 0.01 (-)	2.53 +/- 0.02 (-)
Dermatology	13.25 +/- 0.14	13.10 +/- 0.07	13.28 +/- 0.11
Hepatitis	3.81 +/- 0.15	3.33 +/- 0.05 (-)	3.33 +/- 0.00 (-)
Tic-tac-toe	1.26 +/- 0.10	1.10 +/- 0.04 (-)	1.10 +/- 0.04 (-)

The two versions of Unordered Rule Set Ant-Miner produced significantly fewer rules for the Ljubljana breast cancer and Cleveland heart disease datasets. The largest difference was for the Cleveland HD dataset, where the original, Ordered Rule List Ant-Miner produced 14.20 rules on average, whereas the two versions of Unordered Rule Set Ant-Miner produced 11, with the number of terms per rule being significantly smaller for the rules discovered by Unordered Set Ant Miner than for the rules discovered by Ordered List Ant-Miner in that dataset.

Although the two versions of Unordered Rule Set Ant-Miner produced a significantly increased number of rules for three out of the six data sets (Wisconsin breast cancer, Hepatitis and Tic-tac-toe), the unordered rule sets discovered for two of those three datasets (Hepatitis and Tic-tac-toe) had a significantly smaller number of terms per rule than the corresponding rule lists discovered by Ordered Rule List Ant-Miner. In any case, overall the two versions of Ant-Miner (with Ordered and Unordered Rules) obtained rule sets with similar levels of simplicity.

Interestingly for the Cleveland HD, Dermatology and Hepatitis datasets there was no deviation from the mean number of rules during the cross validation, for the two versions of Unordered Rule Set Ant-Miner.

5. CONCLUSIONS AND FUTURE RESEARCH

Our experimentation has shown that, overall, the proposed Unordered Rule Set Ant-Miner 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 (size of the classification model). In any case, it should be recalled that the rules discovered by Unordered Rule Set Ant-Miner are more modular than the rules discovered by the original Ordered List Ant-Miner. This is the case because in the former kind of algorithm each rule can be interpreted independently from the others, whereas in the rule list discovered by the original Ant-Miner a given rule should 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 [7], and therefore an advantage of the Unordered Rule Set Ant-Miner proposed in this paper.

The results also highlight the importance of the rule conflict resolution strategy in the application of discovered unordered rules to test data. Further research in this area could be focused on developing rule conflict resolution strategies that are more robust across a number of datasets.

In this work the algorithm parameters were not optimised for any particular data set, since the focus was on comparing the different versions of Ant-Miner, rather than optimizing parameters for each data set. Hence, another area of further research might be to attempt to determine the optimum parameter settings that would maximise the accuracy of the discovered classification rules for each dataset.

6. ACKNOWLEDGMENTS

Discretized versions of the data sets Wisconsin breast cancer, Cleveland heart disease, Dermatology and Hepatitis – which originally contained one or more continuous attributes – were kindly provided by Rafael Parpinelli.

The Ljubljana breast cancer data set was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

7. REFERENCES

[1] M. Dorigo, A. Colorni and V. Maniezzo, "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, 1996.

[2] B. Liu, H.A. Abbass, B. Mckay. Classification rule discovery with ant colony optimization. Proceeding of the IEEE/WIC International Conference on Intelligent Agent Technology, Beijing, China (2003), pp. 83–88.

[3] M. P. Oakes, "Ant Colony Optimisation for Stylometry: The Fedaralist Papers." International Conference on Recent Advances in Soft Computing, November 2004.

[4] Ziqiang Wang, Boqin Feng, Classification Rule Mining with an Improved Ant Colony Algorithm, Lecture Notes in Computer Science, Volume 3339, Jan 2004, pp. 357 – 367.

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

[6] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[7] I.H. Witten and E. Frank. Data Mining: practical machine learning tools and techniques. 2nd Edition. Morgan Kaufmann, 2005.

[8] R.S. Parpinelli, H.S. Lopes and A.A. Freitas. An Ant Colony Algorithm for Classification Rule Discovery. In: H.A. Abbass, R.A. Sarker, C.S. Newton. (Eds.) Data Mining: a Heuristic Approach, pp. 191-208. London: Idea Group Publishing, 2002.

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