A New Classification-Rule Pruning Procedure for an Ant Colony Algorithm

Allen Chan and Alex Freitas

Computing Laboratory, University of Kent,
Canterbury, CT2 7NF, UK
{ac207, A.A.Freitas}@kent.ac.uk

Abstract. This work proposes a new rule pruning procedure for Ant-Miner, an Ant Colony algorithm that discovers classification rules in the context of data mining. The performance of Ant-Miner with the new pruning procedure is evaluated and compared with the performance of the original Ant-Miner across several datasets. The results show that the new pruning procedure has a mixed effect on the performance of Ant-Miner. On one hand, overall it tends to decrease the classification accuracy more often than it improves it. On the other hand, the new pruning procedure in general leads to the discovery of classification rules that are considerably shorter, and so simpler (more easily interpretable by the users) than the rules discovered by the original Ant-Miner.

1 Introduction

Ant-Miner [3] is an Ant Colony algorithm that discovers classification rules in the context of data mining. The basic goal of data mining is to extract, from data, knowledge that is not only accurate but also comprehensible to the user [9], [5]. Knowledge comprehensibility is important because in many applications of data mining the user should validate and interpret discovered knowledge, rather than blindly trust the result provided by an algorithm. A typical example of an application where rule comprehensibility is crucial is medical diagnosis, where rules suggesting a diagnosis for a patient must be interpreted and validated by a medical doctor.

Ant-Miner has been shown to be competitive with a well-known classification algorithm in [3], in experiments across several datasets. However, those experiments did not involve datasets with a large number of attributes, where the rule pruning procedure of Ant-Miner tends to be very time consuming. In order to improve Ant-Miner's scalability to data sets with a larger number of attributes, this paper proposes a faster rule pruning procedure for Ant-Miner. The proposed procedure is essentially a hybrid pruning procedure. It combines Ant-Miner's original pruner with a faster pruning based on the information gain of individual attributes. (See [5] for a review of information gain in general.) The basic idea is that, if the candidate rule to be pruned is a long one, instead of applying Ant-Miner's original pruner the algorithm first applies the faster information gain-based pruner, as a first step to reduce the rule length. In terms of computational cost, this first step "comes for free", since the required value of the information gain is already computed by another procedure of Ant-Miner. Once the rule has been so reduced, Ant-Miner's original pruner – slower but more effective – can be applied to the rule, further reducing its length.

The proposed hybrid rule pruner is evaluated across several datasets, most of them with more than 100 attributes. The results are evaluated with respect to the classification accuracy and the comprehensibility of the discovered rules.

The remainder of this paper is organized as follows. Section 2 reviews the Ant-Miner algorithm. Section 3 describes the proposed hybrid rule pruner. Section 4 reports the results of computational experiments, and section 5 concludes the paper.

2 The Original Ant-Miner Classification Algorithm

A single ant within a colony is normally seen as a highly unintelligent individual, but collectively, as a colony, ants exhibit what is known as swarm intelligence. While ants forage for a food source they deposit on their paths a certain amount of pheromone, a chemical substance to which other ants are attracted. It turns out that over time shorter routes between two points (such as the colony's nest and some food source) will have more pheromone than longer routes, because in a fixed period of time there will be more ants completing a shorter path than a longer path. When selecting between multiple paths, ants will in general be attracted to those paths with the highest concentration of pheromone. As a result, the ants will in general prefer to follow the shortest route within a network of paths, which will further increase the concentration of pheromones in the shortest path, attracting more ants to that path. Therefore, over time ants will converge and follow the shortest route within a network of paths. This has been shown by experiments performed by Deneubourg et al. [2].

Dorigo et al, inspired by this interesting behaviour of ant colonies, first developed Ant Colony Optimization (ACO) to solve difficult combinatorial optimization problems like the classic travelling salesman problem [1], [8]. This idea was then taken from solving optimization problems and applied in the field of data mining for discovering classification rules. The Ant-Miner algorithm, developed by Parpinelli et al. [3], is an adaptation of the ACO paradigm especially for the classification task of data mining. The algorithm implements the basic idea of awarding the best attributes (used by the ants to construct the best rules) with pheromone, which increases the probability of those attributes being selected by the next ants to construct other rules. A simple high-level pseudocode of Ant-Miner is shown in Pseudocode 1, adapted from [7]. A more detailed description of Ant-Miner can be found in [3].

The Ant-Miner algorithm uses a sequential covering approach to discover a list of classification rules which will cover all or most of all the examples in the training set. Rules discovered are in the form of: *IF <term1> AND … AND <term-n> THEN <class>*. Each term takes the form *<attribute=value>*, where *value* belongs to the domain of the *attribute*. The training set holds examples that are used for discovering a list of classification rules. The discovered rule list is empty to start with. Every iteration of the outer REPEAT-UNTIL loop of Pseudocode 1 discovers one classification rule and adds it to the list of discovered rules. Each iteration of the inner REPEAT-UNTIL loop corresponds to the trail of an ant that constructs one candidate rule. At the end of the inner REPEAT-UNTIL loop, the best rule from the set of rules constructed by all ants (i.e., in all iterations of the inner REPEAT-UNTIL loop) is added to the discovered rule list. Examples correctly covered by this rule are removed from the training set before the next iteration of the outer REPEAT-UNTIL loop begins to discover the next rule. (An example is correctly covered by a rule if the example satisfies all conditions of the rule and it has the class predicted by the rule.)

The first procedure of the inner REPEAT-UNTIL loop consists of incrementally constructing a candidate rule. This procedure starts with an empty rule and then adds one term at a time to the current rule. This incremental rule construction will terminate when one of the following two stopping criteria is met: any term added to the current rule would make the rule cover a number of examples less than a user specified threshold, or when all attributes have already been used in the current rule being generated, so that no other attribute is available. (A rule cannot have two occurrences of the same attribute, because this would lead to invalid rules such as "IF <*gender* = *male*> AND <*gender* = *female*>…".)

The inner REPEAT-UNTIL loop will terminate when one of two stopping criteria is met: the number of constructed rules is equal or greater than the maximum number of ants specified by the user, or the rule constructed by an ant is exactly the same as the rule constructed by the a certain number of previous ants. The latter criterion is checked via a convergence test. These stopping criteria are controlled by parameters, which are discussed in detail in [3]. Finally, the outer REPEAT-UNTIL loop terminates when the number of examples in the training set becomes lower than a predefined threshold.

```
TrainingSet = {all training examples};
DiscoveredRuleList = {} /* initialized with empty list */
REPEAT
   Initialize all trails with the same amount of
   pheromone;
   REPEAT
      An ant incrementally constructs a
         candidate classification rule;
      Prune the just-constructed rule;
      Update the pheromone of all trails;
   UNTIL (stopping criteria)
   Choose the best rule out of all candidate
      rules constructed by all ants;
   Add the best rule to DiscoveredRuleList;
   TrainingSet = TrainingSet - {examples correctly
      covered by best rule};
UNTIL (stopping criteria)
```

Pseudocode 1. A high-level description of the original Ant-Miner

For the purpose of this paper, the most important part of Ant-Miner is its rule pruning procedure. This procedure is computationally expensive and it can be considered the bottleneck of the algorithm with respect to processing time and scalability to large data sets, as discussed in the next section.

3 Extending Ant-Miner with a Faster Rule Pruning Procedure

3.1 The Motivation for Rule Pruning

Rule pruning is a commonplace data mining technique, used in the vast majority of rule induction algorithms [5]. Pruning can improve the quality of a rule by removing irrelevant terms from the rule antecedent. As a result, pruning can improve both the predictive accuracy and the comprehensibility of the rule.

It should be noted that Ant-Miner, like the majority of rule induction algorithms, can potentially discover rules with a long rule antecedent (with many terms), hindering the comprehensibility of the rule. Indeed, rules take the form of IF <antecedent> THEN <consequent> where the rule antecedent is a conjunction of *n* terms, where the value of *n* can potentially be close to the total number of attributes in the dataset. This means that a rule can become too long for a user to be able to interpret it. Hence, there is a preference for shorter, more comprehensible rules.

3.2 Ant-Miner's Original Rule Pruner

Ant-Miner's original rule pruner takes a freshly generated rule by the current ant and tries to improve its quality (measured by the rule's predictive accuracy), by removing irrelevant terms from the rule antecedent. This is done by iteratively removing one term at a time while it improves on the rule's quality. This iterative process stops when no term removal will further increase the quality of the current rule undergoing pruning. The entire rule pruning process is described in Pseudocode 2.

```
Execute_pruning = true;
WHILE (Execute_pruning = true) AND
      (Number of terms in current rule antecedent > 1)
   FOR EACH (term $t_i$ in the current rule to be pruned)
      Temporarily remove $t_i$ and assign to
      the rule consequent the most frequent class among
      the examples covered by the rule antecedent;
      Evaluate rule quality;
      Reinstate term $t_i$ in rule antecedent;
   END FOR
   IF (rule quality was improved w.r.t. original rule's
       Quality in some iteration of the FOR loop) THEN
       Remove permanently the term whose removal improves
       current rule most;
   ELSE
      Execute_pruning = false;
   END IF-THEN-ELSE
END WHILE
```

Pseudocode 2. A high-level description of Ant-Miner's original rule pruner

Initial experiments conducted by Parpinelli et al. [3] showed that Ant-Miner produces rules that have a good predictive accuracy and are relatively short on average. However, that work also presented an analysis of computational time complexity showing that rule pruning is the most time consuming part of the algorithm, and that the time taken by Ant-Miner's rule pruner is quite sensitive to the number of attributes in the data being mined. This is due to the fact that the larger the number of attributes in the data being mined, in general the larger the number of terms in a constructed rule before pruning, and so the larger the number of iterations in the loops of Pseudocode 2. In each iteration of the FOR EACH loop a term is temporarily removed and the quality of the reduced candidate rule has to be computed by formula (1).

$$\text{Rule Quality} = \text{Sensitivity} \times \text{Specificity} = (TP / (TP+FN)) \times (TN / (TN+FP)) \quad (1)$$

where:

- TP (true positives) is the number of examples that are covered by the rule and have the same class as predicted by the rule;
- FP (false positives) is the number of examples that are covered by the rule and have a class different from the class predicted by the rule;
- TN (true negatives) is the number of examples that are not covered by rule and have a class different from the class predicted by the rule;
- FN (false negatives) is the number of examples that are not covered by the rule but have the same class as predicted by the rule.

The computation of formula (1) is computationally expensive because it requires scanning the entire current training set in order to compute the values of TP, FP, TN and FN. For rules generated with a small number of terms in its antecedent, the pruning method shown in Pseudocode 2 is relatively quick, as there are not a large number of terms to temporarily remove and evaluate rule quality. But for rule antecedents containing a large number of terms, this type of pruning is very computationally expensive. This is because the WHILE loop of the Pseudocode 2 is potentially performed a large number of times (in the worse case the number of terms in the original rule), each iteration of the while loop involves a FOR EACH loop over all current terms in the rule, and each FOR EACH iteration involves a scan of the training set.

It should be noted that the computational time taken by Ant-Miner was not a significant problem in the experiments reported by Parpinelli et al. for the following reason: those experiments involved datasets where the number of attributes was *not* very large. However, in addition to the previously-mentioned theoretical analysis of the computational time complexity of Ant-Miner identifying the rule pruner as the bottleneck of the algorithm [3], there is empirical evidence that the computational time taken by Ant-Miner becomes very long when the data being mined contains a large number of attributes. This empirical evidence consists of recent experiments trying to apply Ant-Miner to a large bioinformatics data set containing 33,079 examples and 854 attributes [10]. In that project Ant-Miner turned out to be so slow that it was not viable to use it to discover classification rules, and a much faster hybrid ACO/PSO (Particle Swarm Optimization) algorithm was developed and used instead. To quote [10]: "…the unusually large amount of attributes and classes associated with this problem mean an *extremely* large amount of computation time is required [by Ant-Miner]." Therefore, there is a clear motivation for developing a considerably faster rule pruning procedure for Ant-Miner and investigate its performance, which is the focus of the remainder of the paper.

3.3 Proposed Hybrid Rule Pruner for Ant-Miner

After an analysis of Ant Miner's original rule pruner, the following is a proposal to a new hybrid rule pruner, combining the original Ant-Miner's rule pruner with a rule pruner based on information gain – the latter somewhat inspired by the rule pruner proposed in [4]. (For a review of information gain in general, see [5].)

```
INPUT:
a) information gain of all terms individually,
calculated using the entire current training set;
/* previously done by another procedure of Ant Miner */
b) value of $r$ /* user-defined parameter: number of terms
in the current rule which will be given to Ant-Miner's
original rule pruner */
Reduced_rule = {};
Num_terms_selected = 0;
IF (number of terms in current rule's antecedent > $r$)
THEN
  WHILE (Num_terms_selected < $r$)
      FOR EACH (term $t_i$ in current rule's antecedent)
         Calculate probability of selecting a term $t_i$ as:
                $\text{prob}(t_i) = \frac{\text{InfoGain}(t_i)}{\sum_{i=1}^{T}(\text{InfoGain}(t_i))}$
       /*$T$ = number of terms in the rule antecedent */
      END FOR
      Create roulette wheel for selection and select one
      Term, called selected_term, by spinning the wheel;
      Reduced_rule = Reduced_rule $\cup$ selected_term;
      Remove selected_term from current rule's antecedent
      to avoid reselection;
      Num_terms_selected = Num_terms_selected + 1;
  END WHILE
  Assign to the consequent of the Reduced_rule the most
  frequent class among all examples covered by the rule;
  Run Ant-Miner's original rule pruner on Reduced_rule;
  ELSE
   Run Ant-Miner's original rule pruner on current rule;
END IF-THEN-ELSE
```

Pseudocode 3. A high-level description of the proposed hybrid rule pruner

First of all, the motivation for this new hybrid rule pruner is to significantly reduce the computational time taken by Ant-Miner, and hopefully do it without unduly reducing the accuracy of the discovered rules, by comparison with the original Ant-Miner. In other words, the basic idea is to combine the effectiveness of the original Ant-Miner pruner (in terms of maximizing predictive accuracy) with the speed of a rule pruner based on information gain. This latter is very fast, because it does not require any scan of the training set, as explained below.

The way this hybrid rule pruner functions is described in Pseudocode 3. The information gain of each term has already been computed by Ant-Miner – in order to compute the values of the heuristic function [3], and is re-used in this hybrid procedure. The parameter r represents the number of terms in the rule antecedent that will be subject to the original Ant-Miner's rule pruner.

As shown in Pseudocode 3, the hybrid pruner selects *r* terms, out of all the terms in the current rule, before applying Ant-Miner's original rule pruner. If the number of terms in the rule antecedent of a freshly generated rule exceeds the value of *r*, the rule first undergoes reduction of the number of terms to the value of parameter *r*. This reduction is obtained as follows. For each term within the rule antecedent, the rule pruner computes a measure of the probability of selecting that term. This probability measure is based on the pre-computed value of that term's information gain with respect to the class attribute. Then the rule pruner selects *r* number of terms using the roulette wheel selection technique (commonplace in genetic algorithms), with the probability of selecting each term proportional to the information gain of that term. Once *r* terms have been selected by spinning the roulette wheel *r* times, the resulting reduced rule is placed back into Ant-Miner's original rule pruner.

If the original rule does not contain a number of terms in its rule antecedent exceeding the value of *r* parameter, then it gets placed straight into Ant-Miner's original rule pruner, with no need to apply the information gain-based pruning.

Intuitively it is difficult to specify an ideal value for parameter *r*, since the best value of this parameter tends to be dataset-dependent. Therefore we have conducted experiments to investigate the influence of different values of this parameter in the performance of the proposed hybrid rule pruner, as discussed in the next section.

4 Computational Results

4.1 Experimental Setup and Datasets Used in the Experiments

As discussed earlier, the proposed hybrid rule pruner has a parameter, *r*, which can have a significant influence in the performance of Ant-Miner. To investigate this issue, we conducted experiments with different values of *r*, varying from 3 to 10. These experiments have two main goals. First, evaluating how sensitive the performance of Ant-Miner with the hybrid pruner is with respect to different values of the parameter *r*. Second, comparing the performance of the hybrid rule pruner with the performance of the original Ant-Miner's rule pruner.

The experiments used mainly 5 datasets – as summarised in Table 1, detailing key statistics (the number of examples, attributes and classes) for each dataset. The *Chess* and the *House-votes* datasets have been taken from the well-known UCI Machine Learning dataset repository – see [6] for details about these data sets. They have a small number of attributes, and were included in the experiments as control datasets. By contrast, the three *Web-mining* datasets are more challenging, because they have a considerably larger number of attributes, varying from 159 to 339. In addition, note that these data sets are very "sparse", in the sense that the number of examples is even smaller than the number of attributes. (Such challenging datasets are commonplace in text/web mining and bioinformatics applications, and therefore it is important to investigate the performance of Ant-Miner in this kind of very sparse dataset.)

In the *Web-mining* datasets, each example is a web page, and the goal is to classify each example into one of three classes: 'Technology', 'Sport' and 'Education'. These classes represent the general subject of the web page. These datasets were harvested from a small selection of BBC and Yahoo web pages relating to the above named subjects. All attributes within these datasets are binary, where each attribute denotes whether or not a given word occurs in a given web page (example). These datasets have been collected by and previously been experimented with Ant-Miner by Holden & Freitas [7].

In addition to the above datasets, we also did experiments with 2 bioinformatics datasets using a single value of *r*. Both datasets have 1872 examples (proteins) and the same values of 102 binary predictor attributes. Each attribute indicates whether or not a protein has a given Prosite pattern. The datasets differ in the class to be predicted: whether or not a protein is involved in DNA repair (first dataset) or in DNA damage (second dataset). The creation of these datasets is explained in [11].

Table 1. Summarized details of the 5 main datasets used in the experiments

Dataset	No. of examples	No. of attributes	No. of classes
Chess	3196	36	2
House-votes-84	434	16	2
Web-mining 1	124	159	3
Web-mining 2	124	293	3
Web-mining 3	124	339	3

Ant-Miner takes on several parameters besides the one we have discussed for the hybrid pruning procedure. The values of all those other parameters were maintained at their default values, specified in [3].

All the experiments were conducted using a stratified 5-fold cross validation procedure [5]. In essence, the dataset was partitioned into five folds with each fold retaining as closely as possible the class distribution of the whole dataset being mined. Each version of Ant-Miner (with original pruner and with the new hybrid pruner) is then run 5 times. In these runs, each fold was used four times as the training set and once as a test set. All results reported in this paper were averaged over the five iterations of the cross validation procedure.

4.2 Results on Classification Accuracy

Reported in Table 2 are the average classification accuracies on the test set and the corresponding standard deviations for each value of *r* in the range of 3 to 10. The classification accuracy is the number of correctly classified test examples divided by the total number of test examples. The first line with results in the table shows the classification accuracy of Ant-Miner using only its original rule pruner. The results for this original version of Ant-Miner are provided as a comparison to see how well it performs against the new hybrid rule pruner. For each dataset, the best result (out of the trials using the hybrid pruner) is highlighted in bold, in order to indicate which value of *r* yielded the highest accuracy.

From Table 2, in general there was a considerable variation in the classification accuracy across different values of the parameter *r*. The only exception was the *House-votes* dataset, where accuracies varied only in the range 93.1–95.4%. In the *Chess* dataset, most values of *r* led to an accuracy higher than the accuracy obtained with the original Ant-Miner's rule pruner, although in general the differences are not statistically significant (considering the standard deviations). On the other hand, in the three *Web-mining* datasets the accuracies obtained with the hybrid rule pruner were lower than the accuracies obtained with Ant-Miner's original rule pruner. This drop in accuracy associated with the use of the hybrid rule pruner in the *Web-mining* datasets can be explained as follows.

Table 2. Classification accuracy rate (%) on the test set (5-fold cross validation)

	Dataset				
Value of *r*	**Chess**	**House-votes**	**Web-mining 1**	**Web-mining 2**	**Web-mining 3**
Original	72.18±9.61	94.23±1.75	68.53±4.29	57.26±7.25	55.90±4.79
3	78.79±7.84	**95.38±1.33**	50.76±3.57 (-)	44.89±5.95	48.83±1.00 (-)
4	83.77±7.77	94.23±1.75	40.26±4.07 (-)	43.32±6.56 (-)	48.95±5.02
5	74.00±9.80	93.07±1.95	50.86±3.24 (-)	38.55±5.48 (-)	50.4±1.91
6	67.40±8.82	94.23±1.75	49.16±2.81 (-)	43.20±6.58 (-)	**52.24±8.17**
7	78.75±9.42	94.00±1.61	51.86±4.62 (-)	49.56±6.37	51.87±5.23
8	79.85±7.99	94.23±1.75	51.10±2.37 (-)	44.92±1.78 (-)	50.27±5.67
9	76.91±8.33	94.00±1.69	**62.83±3.86**	40.06±4.11 (-)	45.84±5.90
10	**84.13±8.88**	93.53±2.19	53.96±2.39 (-)	**55.26±5.92**	49.50±6.00

As mentioned earlier, the *Web-mining* datasets are particularly challenging because they are very "sparse". Each of those datasets contains a number of attributes greater than the number of examples. Recall that the hybrid rule pruner selects *r* number of terms, and terms are selected with probability based on their information gain. In very sparse datasets such as the *Web-mining* datasets, the values of the information gain of the attributes are not very "reliable", since they are prone to overfitting issues. As a result, the hybrid rule pruner has difficulty in selecting *r* relevant terms based on the computed information gain values. Of course, the issue of overfitting also occurs with the other component of the hybrid rule pruner, i.e., the original Ant-Miner's rule pruner. However, the latter is a more direct and more reliable measure of the relevance (predictive power) of the terms, since it is based on evaluating a candidate pruned rule as a whole, taking into account term interactions. By contrast, the heuristic of selecting terms based on the information gain of individual attributes seems more sensitive to overfitting issues, since the quality of each term is estimated by ignoring term interactions, i.e., ignoring the actual effect of the term in the current candidate rule. As a result, in the *Web-mining* datasets the accuracy obtained with the hybrid rule pruner is consistently lower than the accuracy obtained with the Ant-Miner's original rule pruner; a phenomenon that is not observed in the much less sparse *Chess* and *House-votes* datasets.

In any case, the difference of accuracy between Ant-Miner's original rule pruner and the new hybrid rule pruner is not significant in the majority of the cases in Table 2, taking into account the standard deviations. More precisely, in Table 2 the cells where the accuracy of the hybrid pruner is significantly lower than the accuracy of Ant-Miner's original rule pruner – in the sense that the corresponding standard deviation intervals do not overlap – are marked with the symbol "(-)". The drop in accuracy associated with the hybrid pruner was significant in 13 out of the 40 cells with hybrid pruner results in Table 2. In the other 27 cells the difference in accuracy is not significant, and as mentioned earlier the hybrid rule pruner even obtains a somewhat higher accuracy in the majority of the cases for the Chess data set.

We have also applied the hybrid rule pruner with a single value of *r*, viz. *r* = 5, to a couple of bioinformatics datasets with 102 attributes and 1872 examples, as mentioned in section 4.1, to evaluate the performance of the method in less sparse datasets. In the DNA repair dataset the hybrid rule pruner obtained a predictive accuracy of 97.69% ± 0.81%, against the original Ant-Miner's accuracy of 98.50% ± 0.58%. In the DNA damage dataset the hybrid rule pruner obtained an accuracy of 95.30% ± 4.03%, against the original Ant-Miner's accuracy of 93.25% ± 3.49%. Hence, in these datasets the hybrid pruner did not significantly reduce the accuracy.

4.3 Results on Rule Comprehensibility

We now turn to another criterion of performance often used in data mining, namely the comprehensibility of the discovered rules. We emphasize that rule comprehensibility is an important performance criterion in the context of data mining [5], [9] where the goal usually is to discover knowledge that can be interpreted and validated by human beings, to support intelligent decision making. As usual in the literature, we measure rule comprehensibility by the average number of terms in the discovered rules. The basic idea is that in general the shorter a rule is (i.e., the fewer terms it has in its antecedent), the simpler and more easily interpretable the rule is to the user. In this spirit, Table 3 reports the average number of terms per discovered rule when using the original Ant-Miner's rule pruner and when using the hybrid rule pruner – again, with values of *r* varying from 3 to 10. Similarly to Table 2, the numbers after the symbol "±" are standard deviations. For each dataset, the best result (i.e., the smallest number of terms per rule) is shown in bold.

Table 3. Average number of terms per discovered rule

	Dataset				
Value of *r*	**Chess**	**House-votes**	**Web-mining 1**	**Web-mining 2**	**Web-mining 3**
Original	3.35±0.49	0.95±0.05	10.07±0.65	10.02±1.76	7.33±1.52
3	**1.50±0.14**	0.96±0.07	**1.77±0.23**	**1.32±0.15**	**1.30±0.24**
4	1.85±0.15	**0.86±0.06**	2.42±0.28	2.00±0.15	2.00±0.15
5	1.97±0.24	1.12±0.12	3.13±0.08	2.30±0.13	2.60±0.24
6	2.07±0.14	0.95±0.05	3.62±0.19	3.20±0.36	3.13±0.27
7	2.62±0.24	1.27±0.17	3.97±0.42	3.38±0.20	3.65±0.49
8	2.22±0.65	0.95±0.05	4.67±0.47	3.77±0.34	4.67±0.27
9	2.48±0.15	1.00±0.08	5.67±0.38	3.80±0.34	3.87±0.54
10	2.51±0.30	0.95±0.05	5.53±0.49	4.33±0.46	4.37±0.38

The only dataset in which the hybrid rule pruner did not significantly lower rule length, by comparison with Ant-Miner's original rule pruner, was the *House-votes* dataset. In this dataset the original Ant-Miner already obtained a very short average rule length, close to 1, and so it is perfectly acceptable that this result cannot be significantly improved. In the other four datasets, the hybrid rule pruner has significantly lowered the rule length, and so significantly improved rule comprehensibility, taking into account the standard deviations, for all tested values of *r*. The results were particularly good in the *Web-mining* datasets, as can be seen in Table 3, where rule length is reduced to less than half the rule length associated with the original Ant-Miner in most cases.

As expected, the shortest rule lengths were in general obtained with the smallest tried value of r, i.e., $r = 3$. With this value of r, in the *Web-mining* datasets rule length is reduced from about 10 or 7 to less than 1.5, a very significant improvement in rule comprehensibility. In addition, in all datasets but the *House-votes* one, there is a clear correlation between the value of r and the average length of the discovered rules. That is, in four out of the five datasets, in general the larger the value of r the larger the average rule length, and so the less comprehensible the discovered rules are. This result can be explained by the fact that the hybrid rule pruner's component based on information gain is more "aggressive" than the other component – Ant-Miner's original pruner. The latter is more "conservative" in the sense that it will only remove a term from a candidate rule if that removal improves the rule quality. By contrast, the pruner based on information gain always reduces the rule to r terms as the first step of the hybrid pruner, and so the hybrid pruner as a whole tends to produce shorter rules as the value of r is reduced.

5 Conclusions and Future Work

This work has proposed a new hybrid rule pruner for the Ant-Miner algorithm. The hybrid pruner combines Ant-Miner's original pruner with a faster pruning based on information gain. The basic idea is that, if the candidate rule to be pruned is a long one, instead of applying Ant-Miner's original pruner the algorithm first applies the faster information gain-based pruner, as a first step to reduce the rule length. In terms of computational cost, this first step "comes for free", since the required value of the information gain is already computed by another procedure of Ant-Miner. Once the rule has been so reduced, Ant-Miner's original pruner – slower but more effective – can be applied to the rule, further reducing its length.

Experiments were performed with several data sets, comparing the performance of the proposed hybrid rule pruner with the performance of Ant-Miner's original rule pruner. In general the hybrid pruner significantly reduced the computational time of Ant-Miner, by comparison with the computational time taken with the original rule pruner. In the datasets with the largest numbers of attributes (the *Web-mining* datasets), in most cases the computational time was significantly reduced, by comparison with the original Ant-Miner's computational time. In particular, in the *Web-mining-2* data set, the use of the hybrid rule pruner reduced Ant-Miner's computational time to a fraction of the original Ant-Miner's time in all cases, and this fraction varied from 11.6% in the best case to 65.0% in the worst case. A larger computational time reduction is expected in a dataset with a much larger number of attributes and examples. Concerning the quality of the classification rules discovered by Ant-Miner with the new hybrid rule pruner, there are three main conclusions.

First, the predictive accuracy of Ant-Miner is quite sensitive to values of a parameter of the hybrid rule pruner that determines how aggressive the information gain-based rule pruner is. Hence, when using the hybrid rule pruner in important real-world problems, it is recommended to carry out experiments optimizing the value of this parameter for the target dataset. Such parameter optimization is, of course, normally recommended in the context of data mining in general, where the performance of the algorithm is typically considerably dependent on the dataset being mined. Second, with respect to the comprehensibility of the discovered rules, the hybrid rule pruner in general led to the discovery of rules considerably shorter (and so more easily interpretable by users) than the rules discovered with the original Ant-Miner's rule pruner. Hence, the hybrid rule pruner is particularly recommended in applications where rule comprehensibility is very important, such as medical applications – where discovered rules should be carefully interpreted by experts before they are actually used to diagnose a patient or suggest a medical treatment. Third, the results suggest that, as long as the main parameter of the hybrid rule pruner is suitably adjusted for the target data set, it is possible to obtain a good trade-off between accuracy and comprehensibility. In each of the three web mining data sets – where accuracy was overall most reduced by using the hybrid rule pruner – the hybrid rule pruner with its best parameter value obtained a rule set with no significant drop in accuracy and with a significant gain in comprehensibility. However, to be on the safe side it is recommended to use Ant-Miner's original rule pruner whenever possible, in order to avoid the potential loss of accuracy associated with the hybrid rule pruner.

A future research direction is to develop a more adaptive version of the proposed hybrid rule pruner, where the value of *r* is automatically adapted by the algorithm on-the-fly, rather than being statically determined by the user.

References

[1] M. Dorigo, and L. M. Gambardella, "Ant colonies for the travelling salesman problem", *BioSystems, 43:* 73 – 81, 1997.

[2] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant.", *Journal of Insect Behaviour, 3*:159 – 168, 1990.

[3] R.S. Parpinelli, H.S. Lopes and A.A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm", *IEEE Trans. on Evolutionary Comput., 6(4)*, Aug 2002, 321-332

[4] Deborah R. Carvalho and A.A. Freitas, "A hybrid decision tree/genetic algorithm method for data mining" *Information Sciences 163(1-3),* pp. 13-35. June 2004.

[5] I. H. Witten and E. Frank, *Data Mining – Pratical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann 2000.

[6] UCI Machine Learning Repository (University of California at Irvine) – http://www.ics.uci.edu/~mlearn/MLSummary.html (visited 14/10/2004)

[7] N.Holden and A.A.Freitas, "Web Page Classification with an Ant Colony Algorithm" *Proc. 2004 Parallel Problem Solving from Nature, LNCS 3242,* 1092-1102. Springer, 2004.

[8] M. Dorigo and T. Stuetzle. *Ant Colony Optimization*. MIT Press, 2004.

[9] U.M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From data mining to knowledge discovery: an overview. In: U.M. Fayyad et al (Eds.) *Advances in Knowledge Discovery and Data Mining*, 1-34. AAAI/MIT, 1996.

[10] N. Holden and A.A. Freitas. "A Hybrid Particle Swarm/Ant Colony Algorithm for the Classification of Hierarchical Biological Data". *Proc. 2005 IEEE Swarm Intelligence Symposium*, 100-107. IEEE, 2005.

[11] A. Chen. *Ant Colony Optimisation for High-Dimensional and Multi-Label Classification in Data Mining*. Master Thesis (in preparation). University of Kent, UK. Sep. 2005.