A Study of Different Quality Evaluation Functions in the cAnt-Miner$_{\mathrm{PB}}$ Classification Algorithm

Matthew Medland
School of Computing
University of Kent, Canterbury
Kent, CT2 7NF
mm443@kent.ac.uk

Fernando E. B. Otero
School of Computing
University of Kent, Canterbury
Kent, CT2 7NF
F.E.B.Otero@kent.ac.uk

ABSTRACT

Ant colony optimization (ACO) algorithms for classification in general employ a sequential covering strategy to create a list of classification rules. A key component in this strategy is the selection of the rule quality function, since the algorithm aims at creating one rule at a time using an ACO-based procedure to search the best rule. Recently, an improved strategy has been proposed in the cAnt-Miner$_{\mathrm{PB}}$ algorithm, where an ACO-based procedure is used to create a complete list of rules instead of individual rules. In the cAnt-Miner$_{\mathrm{PB}}$ algorithm, the rule quality function has a smaller role and the search is guided by the quality of a list of rules. This paper sets out to determine the effect of different rule and list quality functions in terms of both predictive accuracy and size of the discovered model in cAnt-Miner$_{\mathrm{PB}}$. The comparative analysis is performed using 12 data sets from the UCI Machine Learning repository and shows that the effect of the rule quality functions in cAnt-Miner$_{\mathrm{PB}}$ is different from the results previously presented in the literature.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

Keywords

ant colony optimization, classification, sequential covering, rule quality functions, list quality functions

1. INTRODUCTION

Data mining is, in essence, the automated search for useful and usable patterns in data. These patterns are used by scientists, marketing staff, CEOs, bankers and anyone with an interest in what their data holds. There are three main approaches to finding these patterns, and subsequently three separate data mining tasks. These are *classification*, *association rule learning* and *clustering* [13]. Of these tasks, classification is the most studied.

Classification aims to produce a model which can be used to group objects (physical or metaphorical) and assign a class value (group name). Therefore, classification problems can be viewed as optimisation problems, where the goal is to find the best model that represents the predictive relationships in the data. There are many different model representations, such as 'black-box' (not easily comprehensible) models produced by support vector machines (SVM) and artificial neural networks, and 'white-box' (comprehensible) decision tree and classification rule models.

The approach on which this paper focuses is stochastic based upon Ant Colony Optimization (ACO), a meta-heuristic inspired by the foraging behaviour of ants [2]. The first application of ACO for the classification task in data mining was reported by Parpinelli et al. [11], where an ACO algorithm—called Ant-Miner—is proposed for the discovery of classification rules. Ant-Miner aims at extracting a list of *IF-THEN* classification rules of the form *IF* antecedent *THEN* consequent from a data set—the antecedent is composed by predictor attribute-value conditions, while the consequent corresponds to the class value to be predicted.

Several extensions of Ant-Miner have been proposed in the literature, as reviewed in [8]. The majority of these extensions maintain the overall structure of the algorithm—i.e., the algorithm employs an ACO-based procedure to create individual rules in order to produce a complete classification model (list of rules). This strategy to obtain a list of classification rules is referred to as sequential covering (or separate-and-conquer), where each rule is discovered individually. Recently, an improved strategy has been proposed in the cAnt-Miner$_{\mathrm{PB}}$ algorithm [10], where an ACO-based procedure is used to create a complete list of rules instead of individual rules, and it has been shown to outperform state-of-the-art rule induction algorithms. One of the main differences between the Ant-Miner and cAnt-Miner$_{\mathrm{PB}}$ algorithms is that in Ant-Miner the search is performed (and optimised) to find the best individual rules at each step of the sequential covering, whereas in cAnt-Miner$_{\mathrm{PB}}$ the search is performed (and optimised) to find the best list of rules. While in Ant-Miner the search is guided by the quality of an individual rule, in cAnt-Miner$_{\mathrm{PB}}$ the search is guided by the quality of a candidate list of rules.

The importance of the rule quality function in sequential covering algorithms has been highlighted in [4, 6], and in the

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

context of ACO classification algorithm in [12]. In this paper we present a study of the effects of rule quality functions, as well as list quality functions, in the *c*Ant-Miner$_{\text{PB}}$ algorithm. Different than Ant-Miner and its extensions, *c*Ant-Miner$_{\text{PB}}$ employs two quality functions in its search procedure: a rule quality function, which is used to decide whether or not to prune an individual rule; and a list quality function, which guides the search (i.e., the pheromone update is based on the quality of a candidate list of rules). We evaluate different combinations of rule quality and list quality functions in terms of both predictive accuracy and size of the classification model, and compare the results against the default combination used in *c*Ant-Miner$_{\text{PB}}$.

The remainder of this paper is organised as follows. In Section 2 we discuss related works, focusing on the Ant-Miner family of algorithms and rule learning heuristics. In Section 3 we revisit the evaluation functions in greater depth, and define the functions used in our study. In Section 4 we describe both the steps for tuning our heuristics (Subsection 4.1) and our final results (Subsection 4.2). Section 5 then concludes this paper by assessing and aiming to explain our results.

2. BACKGROUND

Sequential covering, also called separate-and-conquer, is a classification rule learning approach with two main discrete steps [6]. In essence, the approach finds a rule with a high quality on the dataset (conquer), and then removes the examples which are covered by the rule (separate). This repeats until there are either no or few remaining training examples.

The effectiveness of a sequential covering algorithm stems from the quality of the rules that are produced [4, 6]. Some rules may only cover correctly classified examples (high consistency) but might only cover a small sample (low coverage). Conversely, a rule may cover a large portion of the dataset (high coverage) but mis-classify most of them (low consistency). Both of these results are undesirable, and rule quality functions tend to trade one off against the other. It is for this reason that the rule quality function needs to be carefully chosen to provide an optimal trade-off, and result in a list of rules with high predictive accuracy.

Ant-Miner [11] is a sequential covering rule learning metaheuristic based upon the foraging behaviour of a colony of ants. Ants, when searching for routes to food sources, drop pheromones. The ants tend to move towards areas with a higher concentration of pheromone, and as a result of the shortest route being travelled more often, it will have more pheromone and will therefore be reinforced. This behaviour has been observed, translated into optimisation algorithms and is referred to as Ant Colony Optimization (ACO) [2].

The Ant-Miner algorithm works as follows. First, a construction graph is created where each node is a value for a given variable, with every variable-value pair from the dataset represented. Each ant then moves from a start node (with an empty rule) and stochastically chooses a vertex with a probability based upon the pheromone value and a heuristic value. The visited vertex is a rule term as a variable-value pair. The ant will continue to add new terms until either all variables have been covered, or if by adding another term, it would decrease the number of covered examples bellow a predefined threshold. After a rule has been created, a pruning procedure removes irrelevant terms from the newly created rule. Once every ant in the colony (of predetermined size) has traversed the graph, the best rule (based on a rule quality function) is selected and the pheromone levels are adjusted. The pheromone on the terms included in the best rule increase and the pheromone on the others (unused terms) decrease. After a rule has been created, all of the examples covered by the rule are removed from the dataset and the next rule is created. The algorithm finishes once the training set has less than a predefined number of training examples remaining. Most of the proposed extensions of Ant-Miner follow this same strategy to create a list of rules [8].

Require: training examples
Ensure: best discovered list of rules
1. *InitialisePheromones()*;
2. $list_{gb} \leftarrow \varnothing$;
3. $m \leftarrow 0$;
4. **while** m < maximum iterations **and** not stagnation **do**
5. $list_{ib} \leftarrow \varnothing$;
6. **for** n $\leftarrow$ 1 **to** colony_size **do**
7. $examples \leftarrow$ *all training examples*;
8. $list_n \leftarrow \varnothing$;
9. **while** $|examples|$ > maximum uncovered **do**
10. *ComputeHeuristicInformation(examples)*;
11. $rule \leftarrow CreateRule(examples)$;
12. $Prune(rule)$;
13. $examples \leftarrow examples - Covered(rule, examples)$;
14. $list_n \leftarrow list_n + rule$;
15. **end while**
16. **if** $Quality(list_n) > Quality(list_{ib})$ **then**
17. $list_{ib} \leftarrow list_n$;
18. **end if**
19. **end for**
20. $UpdatePheromones(list_{ib})$;
21. **if** $Quality(list_{ib}) > Quality(list_{gb})$ **then**
22. $list_{gb} \leftarrow list_{ib}$;
23. **end if**
24. $m \leftarrow m + 1$;
25. **end while**
26. **return** $list_{gb}$;

Figure 1: High-level pseudocode of the *c*Ant-Miner$_{\text{PB}}$ algorithm [10].

*c*Ant-Miner$_{\text{PB}}$ [10] is an ACO classification algorithm that employs a different search strategy than Ant-Miner. The main difference in the search strategy of *c*Ant-Miner$_{\text{PB}}$ is that it aims to create the best list of rules rather than creating a list of best rules. This change is slight, but has a profound effect on the algorithm. The ants, rather than creating one rule each, create an entire model (list of rules) each. The best model (based on a list quality function) created in an iteration is used to update the pheromone values, and it is compared to the previous best model. If the new model is better, it survives and may end up being the final model.

The high-level pseudocode of the *c*Ant-Miner$_{\text{PB}}$ algorithm is presented in Figure 1 and works as follows. At each iteration, an ant in the colony starts with an empty list of rules and the full training set. The ant then creates a rule,

prunes the rule, and removes all of the covered examples from the training set. It is important to note that the only stage at which the rule quality function is used is in the pruning step. The ant then repeats these steps until the number of examples remaining are below a given threshold. The list of rules created by the ant is then compared to the current iteration's best list of rules based on the list quality function, and if it is better than the current best, it replaces the previous best. Once the colony's best list of rules has been found, the pheromones of the construction graph are updated. This entire process repeats until either the maximum number of iterations have been reached or until the colony converges. The best list of rules found is returned at the end.

This highlights the main differences between Ant-Miner and cAnt-Miner$_{\text{PB}}$. The Ant-Miner algorithm is guided by the quality of the individual rule. The best rule found by the colony is always used, without regard for how this will affect the quality of the list of rules being created. The cAnt-Miner$_{\text{PB}}$, on the other hand, has the goal of producing the best list of rules, with less regard to how rules perform individually. The entire list of rules is created at once, and the best is chosen to guide the search.

In evaluating the rules in either algorithm, the quality of each individual rule is calculated, and in Ant-Miner the function which calculates this quality may have a large effect on the quality of the model as a whole. It has been suggested that traditional non-parametric rule quality functions, such as Sensitivity × Specificity, accuracy or precision, implement a fixed trade off [6]. That is, they either prefer consistency or coverage, with no means of modifying this bias. Parametric functions, such as the the m-estimate or Klösgen, allow the user to modify this trade off depending on their application.

Due to the importance of rule quality functions in sequential covering classification algorithms, there is significant interest in this field and a number of papers have been published. One is a mainly theoretical study [6], which looks at how rule quality functions affect a relatively simple sequential search algorithm. In this study, it was shown that the parametrized functions performed well. The m-estimate and Klösgen functions were amongst the best performing parametric functions. A further study [12] focuses on how the rule quality function can affect ACO-based classification algorithms, which follow the same strategy as Ant-Miner. The results presented in [12] show that sensitivity × specificity, used in Ant-Miner and also in cAnt-Miner$_{\text{PB}}$, is not amongst the best performing rule quality functions that have been evaluated. These studies, however, are focused on the effect of the rule quality function in traditional sequential covering algorithms. As cAnt-Miner$_{\text{PB}}$ employs a different strategy, where the rule quality function has a smaller role and the search is guided by the quality of a list of rules, it is interesting to evaluate the effects of different rule quality functions, as well as different list quality functions.

3. EVALUATION FUNCTIONS

The rule evaluation function is the keystone of sequential covering algorithms. In Ant-Miner (and its variations), every time an ant creates a rule, its quality is calculated and the rule is only considered if it has the best quality in that iteration. In cAnt-Miner$_{\text{PB}}$, however, the rule quality function has a less prominent role. It is only used during the pruning stage, as the search is guided by the quality of a list of rules. It is with these facts in mind that we are testing different rule quality functions to judge how much effect each has on cAnt-Miner$_{\text{PB}}$, and also testing different list quality functions.

3.1 Rule Quality Functions

This section presents the rule quality functions used in our study. A series of shorthands are being used in the equations below to condense the formulae and are defined as follows.

TP The number of examples covered by the rule that belong to the class predicted by the rule (true positives).

FP The number of examples covered by the rule that do not belong to the class predicted by the rule (false positives).

TN The number of examples not covered by the rule that do not belong to the class predicted by the rule (true negatives).

FN The number of examples not covered by the rule that belong to the class predicted by the rule (false negatives).

S The total number of examples (TP + FP + TN + FN).

Sensitivity × Specificity.

Sensitivity × Specificity is used in the original Ant-Miner and in the cAnt-Miner$_{\text{PB}}$. Sensitivity measures the fraction of true positives covered by the rule and the specificity measures the fraction of examples of different classes which were not covered [7]. The Sensitivity × Specificity is given by:

$$\underbrace{\frac{TP}{TP+FN}}_{\text{Sensitivity}} \cdot \underbrace{\frac{TN}{TN+FP}}_{\text{Specificity}} \tag{1}$$

Confidence + Coverage.

The Confidence + Coverage was used in AntMiner+ algorithm [9]. The confidence measures the fraction of examples covered by the rule correctly, and the coverage measures the importance of the rule by calculating the fraction of correctly covered examples against all remaining examples. The Confidence + Coverage is given by:

$$\underbrace{\frac{TP}{TP+FP}}_{\text{Confidence}} + \underbrace{\frac{TP}{S}}_{\text{Coverage}} \tag{2}$$

Jaccard Coefficient.

The Jaccard coefficient calculates the similarity between sample sets. In our case, the Jaccard coefficient is the fraction of correctly covered examples divided by the number of all but the examples not covered by the rule that do not belong to the class predicted by the rule. This rule quality function has been used in the study presented by Salama and Abdelbar [12], where it was one of the best performing functions. The Jaccard coefficient is given by:

$$\frac{TP}{TP+FP+FN} \tag{3}$$

Klösgen.

The Klösgen function, when $\omega = 0$, acts as the precision gain (the precision with respect to $\frac{(TP+FN)}{S}$). As ω increases the Klösgen acts in a manner similar to recall. The transition from one to the other is not linear, and as ω increases further it starts acting as coverage. An ω value of 0.4323 was found to be optimum by Janssen and Fürnkranz [6]. We have tested the neighbour values $\{0.3, 0.35, 0.4, 0.45, 0.5\}$ in a tuning step, as it will be discussed in Section 4. The Klösgen is given by:

$$\left(\frac{TP+FP}{S}\right)^{\omega} \cdot \left(\frac{TP}{TP+FP} - \frac{TP+FN}{S}\right) \tag{4}$$

m-estimate.

The m-estimate is a parametric function with the parameter m. The m value found to be the best by Janssen and Fürnkranz was 22.466, and this resulted in the best quality function overall in their study. We have tested the neighbour values $\{21, 22, 23, 24, 25\}$ in a tuning step, as it will be discussed in Section 4. The m-estimate is given by:

$$\frac{TP + m \cdot \frac{TP}{S}}{TP + FP + m} \tag{5}$$

3.2 List Quality Functions

This section presents the list quality functions used in our study. While we focus on lists of rules in this paper, the list quality functions presented in this section are in fact classification model evaluation functions.

Predictive Accuracy.

This is the standard list quality function used by cAnt-Miner$_{\text{PB}}$. It corresponds to the number of correct classification divided by the total number of examples. The accuracy is given by:

$$\frac{number\ of\ correct\ classification}{total\ number\ of\ examples} \tag{6}$$

Micro-Average F-Measure.

The micro-average F-Measure is a commonly used evaluation function in information retrieval and text classification systems. It involves the global precision (Pr_{mic}) and recall (Re_{mic}) values, given by:

$$Pr_{mic} = \frac{\sum_{i=0}^{|C|} TP_i}{\sum_{i=0}^{|C|} TP_i + \sum_{i=0}^{|C|} FP_i} \tag{7}$$

$$Re_{mic} = \frac{\sum_{i=0}^{|C|} TP_i}{\sum_{i=0}^{|C|} TP_i + \sum_{i=0}^{|C|} FN_i} \tag{8}$$

where $|C|$ corresponds to the total number of classes, TP_i, FP_i and FN_i correspond to the number of true positive (the number of examples of class i predicted as being in class i), false positive (the number of examples predicted as being in class i which are not of class i) and false negative (the number of examples of class i not predicted as being of class i) of the i-th class, respectively. Given the Pr_{mic} and Re_{mic}, the micro-average F-Measure is given by:

$$\frac{(1+\beta^2) \cdot Pr_{mic} \cdot Re_{mic}}{(\beta^2 \cdot Pr_{mic}) + Re_{mic}} \tag{9}$$

where β controls the importance of precision and recall: β values lower than 1 puts more emphasis on precision than recall; β values greater than 1 puts more emphasis on recall than precision; and a β equal to 1 puts the same emphasis on both precision and recall.

Macro-Average F-Measure.

The macro-average F-Measure is also a commonly used evaluation function in information retrieval and text classification systems. The main difference between the micro-average and macro-average is that in the former, global precision and recall values are used in the calculation of the F-Measure, while in the latter, individual precision and recall values for each class are used in the calculation. The precision (Pr_i) and recall (Re_i) values for the i-th class is given by:

$$Pr_i = \frac{TP_i}{TP_i + FP_i} \tag{10}$$

$$Re_i = \frac{TP_i}{TP_i + FN_i} \tag{11}$$

The macro-average F-Measure is given by:

$$\sum_{i=0}^{|C|} \left(\frac{1}{|C|} \cdot \frac{(1+\beta^2) \cdot Pr_i \cdot Re_i}{(\beta^2 \cdot Pr_i) + Re_i}\right) \tag{12}$$

Weighted Macro-Average F-Measure.

The macro-average F-Measure presented in Equation (12) gives an equal importance for every class. In order to take the number of examples of each class in the quality function, we have used a weighted macro-average F-Measure (where the weight corresponds to the fraction of examples belonging to the class) given by:

$$\sum_{i=0}^{|C|} \left(\frac{TP_i + FN_i}{|D|} \cdot \frac{(1+\beta^2) \cdot Pr_i \cdot Re_i}{(\beta^2 \cdot Pr_i) + Re_i}\right) \tag{13}$$

where $|D|$ corresponds to the total number of examples.

Inverse Weighted Macro-Average F-Measure.

Using the same idea of having different weights for classes, the inverse weighted macro-average puts more emphasis on classes with a smaller number of examples (the smaller the number of examples, the more important the class) and it is given by:

$$\sum_{i=0}^{|C|} \left(\left[1 - \frac{TP_i + FN_i}{|D|}\right] \cdot \frac{(1+\beta^2) \cdot Pr_i \cdot Re_i}{(\beta^2 \cdot Pr_i) + Re_i}\right) \tag{14}$$

4. COMPUTATIONAL RESULTS

In order to evaluate the different rule and list quality functions, we first determined optimal parameter values for the parametric functions in a tuning step, described in Subsection 4.1. These parameter values were then used in our final experiments, described in Subsection 4.2.

4.1 Experimental Setup

We split the process of finding the optimal combinations of rule and list quality functions into three distinct steps. Throughout these three experiments we used the automobile, blood transfusion, ecoli, heart-c, heart-h, hepatitis, horse-colic, voting records and zoo datasets from the UCI Machine Learning repository [3]. These data sets are only used for parameter tuning and comprise our training data sets.

1. **Rule Quality Functions:** With the aim of finding the optimal parameter values for the parametric rule quality functions—ω parameter for Klösgen (Equation 4) and m parameter for m-estimate (Equation 5), we ran the cAnt-Miner$_{\text{PB}}$ algorithm on our training data sets for each different rule evaluation function and parameter value (described in Section 3.1), repeating each experiment 10 times, each time using 10-fold cross validation. The list quality function was fixed to predictive accuracy, the default list quality function used in cAnt-Miner$_{\text{PB}}$.

2. **List Quality Functions:** Using a similar setup of the previous step, we have determined optimal values for the β parameter used in the list quality functions based on the F-Measure—micro-average F-Measure (Equation 9), macro-average F-Measure (Equation 12), weighted macro-average F-Measure (Equation 13) and inverse weighted macro-average F-Measure (Equation 14). In this step, the rule quality function was fixed to the Sensitivity × Specificity, the default rule quality function used in cAnt-Miner$_{\text{PB}}$.

3. **Determining the best combination of rule and list quality functions:** Finally, we then found the best combination of rule and list quality functions by testing each of the 5 different rule quality functions (using the optimised parameters found in the first step) with each of the 4 different list quality functions (using the optimised parameters found in the second step).

As a result of the experiments outlined above we have determined a group of optimal rule and list quality functions pairings, presented in Table 1. In that table, a row corresponds to a pair of rule and list quality functions. Pairing 1 is the default (baseline) used in cAnt-Miner$_{\text{PB}}$. Pairing 2 corresponds to the Klösgen function (with ω equal to 0.5) paired with the predictive accuracy list quality function. Pairing 3 corresponds to the m-estimate function (with m equal to 21) paired with the predictive accuracy list quality function. Pairings 3 and 4 are using the Jaccard and Confidence + Coverage functions, respectively, and are both paired with the weighted macro F-Measure list quality function (with β equal to 0.5).

4.2 Evaluating the Optimal Combinations

Once the optimal parameters and pairing (Table 1) had been determined, we aimed to find which combination is the most effective. To do this we used a separate set of data sets from the UCI Machine Learning repository [3]. The summary of the data sets used is presented in Table 2. During this experiment, we used 10-fold cross-validation (each fold having the same class distribution, i.e., stratified folds), and the default parameters of cAnt-Miner$_{\text{PB}}$ [10]: colony size of 5, 500 iterations and evaporation factor 0.90.[1] The only parameter that has been changed is the minimum number of examples, which was set to 2 (instead of 10) to test if any of the rule quality functions are prone to overfitting. Given the stochastic nature of cAnt-Miner$_{\text{PB}}$, we ran the algorithm 10 times for each fold of the cross-validation.

Table 3 presents the results for each of the function pairings concerning the predictive accuracy and Table 4 presents the results concerning the size of the discovered list of rules (measured as the total number of rule conditions). Each value in those tables represents the average value calculated over the cross-validation procedure. Table 5 shows the results of the Friedman statistical test [1, 5] for predictive accuracy and size of the discovered list of rules. The information presented in Table 5 corresponds to the average rank (first column), where the lower the rank the better the pairing's performance, the p-value of the statistical test (second column), and Holm's post-hoc critical value (third column).

Table 1: Optimal combinations of rule and list quality functions.

Pair	Rule Function	List Function
1	Sensitivity × Specificity	Accuracy
2	Klösgen ($\omega = 0.5$)	Accuracy
3	m-estimate ($m = 21$)	Accuracy
4	Jaccard	Weighted Macro F-Measure ($\beta = 0.5$)
5	Confidence + Coverage	Weighted Macro F-Measure ($\beta = 0.5$)

Table 2: Summary of the data sets used in the experiments.

data set	attributes		classes	size
	nominal	continuous		
balance-scale	4	0	3	625
breast-l	9	0	2	286
breast-tissue	0	9	6	106
breast-w	0	30	2	569
credit-a	8	6	2	690
dermatology	33	1	6	366
glass	0	9	7	214
ionosphere	0	34	2	351
iris	0	4	3	150
liver-disorders	0	6	2	345
parkinsons	0	22	2	195
wine	0	13	3	178

[1]The binaries and source-code of the cAnt-Miner$_{\text{PB}}$ algorithm implementation used in this paper can be found at http://sourceforge.net/projects/myra. The datasets partitions can be found at http://cs.kent.ac.uk/∼febo.

Table 3: Average predictive accuracy (*average* [*standard error*]) in %, measured by 10-fold cross-validation. The value of the most accurate configuration for a given data set is shown in bold.

data set	Sen × Spe	Klösgen	m-estimate	Jaccard	Conf + Cov
balance-scale	76.99 [0.19]	78.54 [0.32]	**80.22 [0.37]**	77.05 [0.30]	73.96 [0.23]
breast-l	**69.77 [0.60]**	62.51 [0.66]	64.26 [0.62]	69.04 [0.57]	63.00 [0.86]
breast-tissue	**66.74 [0.91]**	62.76 [0.81]	64.30 [0.75]	63.65 [0.72]	62.97 [1.24]
breast-w	93.44 [0.28]	93.14 [0.28]	93.49 [0.25]	**93.72 [0.48]**	93.60 [0.32]
credit-a	84.49 [0.18]	83.10 [0.30]	82.97 [0.38]	**85.22 [0.23]**	79.16 [0.27]
dermatology	92.81 [0.44]	92.94 [0.19]	93.23 [0.35]	**93.38 [0.42]**	91.73 [0.48]
glass	**70.66 [0.70]**	69.16 [0.60]	68.78 [0.56]	69.74 [0.64]	67.11 [0.68]
ionosphere	**90.49 [0.21]**	88.45 [0.39]	89.04 [0.43]	90.09 [0.55]	88.70 [0.40]
iris	94.13 [0.29]	93.00 [0.40]	93.27 [0.48]	**94.33 [0.11]**	93.40 [0.32]
liver-disorders	**65.82 [0.38]**	64.91 [0.45]	63.57 [0.77]	63.25 [0.28]	64.50 [0.57]
parkinsons	**85.49 [0.64]**	83.51 [0.28]	83.07 [0.77]	84.43 [0.88]	84.53 [0.78]
wine	91.63 [0.52]	90.63 [0.79]	92.14 [0.56]	**92.59 [0.44]**	90.90 [0.56]

Table 4: Average number of terms (rule conditions) in the discovered list (*average* [*standard error*]) measured by 10-fold cross-validation. The value of the configuration with the lowest average for a given data set is shown in bold.

data set	Sen × Spe	Klösgen	m-estimate	Jaccard	Conf + Cov
balance-scale	**20.26 [0.30]**	105.21 [1.16]	77.42 [1.50]	20.56 [0.46]	332.07 [6.09]
breast-l	**62.96 [1.90]**	229.71 [2.51]	199.92 [2.53]	70.03 [1.55]	258.77 [1.52]
breast-tissue	**15.93 [0.31]**	26.88 [0.33]	21.21 [0.31]	17.72 [0.18]	32.80 [0.22]
breast-w	13.87 [0.20]	22.90 [0.46]	25.29 [0.42]	**12.11 [0.20]**	18.47 [0.33]
credit-a	**29.04 [0.51]**	111.82 [1.32]	104.19 [1.96]	29.14 [0.68]	228.96 [2.36]
dermatology	**52.13 [0.74]**	58.14 [0.98]	54.62 [0.99]	53.40 [0.92]	69.32 [1.70]
glass	**28.41 [0.23]**	58.86 [0.51]	48.03 [0.41]	29.70 [0.34]	75.68 [1.05]
ionosphere	20.06 [0.24]	30.09 [0.35]	29.15 [0.44]	**14.31 [0.18]**	24.30 [0.75]
iris	8.78 [0.20]	12.70 [0.22]	12.54 [0.31]	**7.32 [0.15]**	11.11 [0.26]
liver-disorders	24.31 [0.37]	93.12 [1.34]	79.13 [0.54]	**21.30 [0.26]**	152.21 [1.24]
parkinsons	12.05 [0.16]	22.29 [0.48]	21.25 [0.36]	**11.42 [0.28]**	18.50 [0.67]
wine	9.68 [0.31]	13.43 [0.37]	12.81 [0.42]	**9.25 [0.15]**	13.40 [0.43]

A row is shown in bold when there is a statistically significant difference between the average rank of a pairing and the control pairing (the pairing with the best average rank)—when the p-value is lower than the critical value—and shows that the control pairing is significantly better than the pairing in that row.

The Sensitivity × Specificity (Pair 1) and Jaccard (Pair 4) were the most successful, each achieving an average rank of 2.08 with regards to predictive accuracy. They were also the highest performing pairings with regards to the size of the list of rules, with Jaccard achieving an average rank of 1.49 and Sensitivity × Specificity achieving a rank of 1.5. Our results show that the parametric functions did not perform well in the *c*Ant-Miner$_{\text{PB}}$ algorithm. The m-estimate function (Pair 3) achieved the third ranking, while the Klösgen (Pair 2) achieved the lowest ranking and it was statistically significantly worse than Sensitivity × Specificity, in terms of both predictive accuracy and model size. This is different than the results presented by Janssen and Fürnkranz [6], where these parametric functions were amongst the best performing ones in a traditional sequential covering algorithm. Considering the results presented by Salama and Abdelbar [12], our results agree in respect to the use of Jaccard function, given that Jaccard is amongst the best performing ones. However, in Salama and Abdelbar study, Sensitivity × Specificity was amongst the worse performing ones, which is not the case in the *c*Ant-Miner$_{\text{PB}}$ algorithm.

While Klösgen (Pair 2) and Confidence + Coverage (Pair 5) were statistically significantly worse than Sensitivity × Specificity in terms of both predictive accuracy and model size, they achieved the highest predictive accuracy during training overall. Therefore, their poor performance in the test set is likely due to overfitting—the case where the list of rules created is too tailored to the training set and does not generalise well, i.e., it has a lower predictive accuracy in unseen test cases.

Table 5: Statistical test results according to the non-parametric Friedman test with the Holm's post-hoc test for $\alpha = 0.05$.

configuration	average rank	p	*Holm*
(i) Accuracy			
Sen × Spe (control)	2.08	–	–
Jaccard	2.08	0.999	0.0500
m-estimate	3.08	0.121	0.0250
Con + Cov	**3.83**	**0.007**	**0.0167**
Klösgen	**3.92**	**0.004**	**0.0125**
(ii) Model Size			
Jaccard (control)	1.49	–	–
Sen × Spe	1.50	0.999	0.0500
m-estimate	**3.42**	**0.003**	**0.0250**
Con + Cov	**4.25**	**2.04E-5**	**0.0167**
Klösgen	**4.33**	**1.14E-5**	**0.0125**

Although our results did not improve the predictive accuracy or the size of the discovered list of rules, they show that the rule quality functions which perform well in traditional sequential covering algorithms do not have the same performance in *c*Ant-Miner$_{\text{PB}}$. This suggests that other aspects of the algorithm (e.g., solution construction, pruning procedure, pheromone update) are more interesting to investigate in order to improve the overall performance of *c*Ant-Miner$_{\text{PB}}$.

5. CONCLUSION

Our findings show that, in contrast to the results found by Salama and Abdelbar [12], the Sensitivity × Specificity rule quality function is a good choice for *c*Ant-Miner$_{\text{PB}}$, and that it performs well alongside the accuracy list quality function. This is a useful discovery as it means that the current default selection for *c*Ant-Miner$_{\text{PB}}$ is adequate. In agreement with Salama and Abdelbar, we have also shown that the Jaccard coefficient performs well and is comparable to the rule quality function currently in use. The use of parametric rule quality functions, used in a similar study by Janssen and Fürnkranz [6], did not improve the predictive accuracy of the algorithm and in some cases were statistically significantly worse. The difference in results is likely to be an effect of the new strategy employed by *c*Ant-Miner$_{\text{PB}}$ to search for the best list of rules instead of the list of best rules.

In our study the optimal rule quality functions and their parameter values were optimised statically (prior to the run of the algorithm). It will be interesting to see if a dynamic selection technique will provide better results, and this is an interesting future research direction.

6. REFERENCES

[1] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Machine Learning Research*, 7:1–30, 2006.

[2] M. Dorigo and T. Stüzle. *Ant Colony Optimization*. The MIT Press, 2004.

[3] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[4] J. Fürnkranz and P. Flach. ROC 'n' Rule Learning—Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58:39–77, 2005.

[5] S. García and F. Herrera. An Extension on 'Statistical Comparisons of Classifiers over Multiple Data Sets' for all Pairwise Comparisons. *Machine Learning Research*, 9:2677–2694, 2008.

[6] F. Janssen and J. Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78:343–379, 2010.

[7] H. S. Lopes, M. S. Coutinho, and W. C. Lima. An evolutionary approach to simulate cognitive feedback learning in medical domain. In E. Sanchez, T. Shibata, and L. A. Zadeh, editors, *Genetic Algorithms and Fuzzy Logic Systems*, volume 7, pages 193–207. 1997.

[8] D. Martens, B. Baesens, and T. Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82:1–42, Jan 2011.

[9] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens. Classification with ant colony optimization. *Evolutionary Computation, IEEE Transactions on*, 11(5):651 –665, Oct 2007.

[10] F. Otero, A. Freitas, and C. Johnson. A new sequential covering strategy for inducing classification rules with ant colony algorithms. To appear in IEEE Transactions on Evolutionary Computation, 2012.

[11] R. Parpinelli, H. Lopes, and A. Freitas. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6(4):321 – 332, Aug 2002.

[12] K. Salama and A. Abdelbar. Exploring different rule quality evaluation functions in aco-based classification algorithms. In *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, pages 1 –8, Apr 2011.

[13] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd edition, 2011.