Kent Academic Repository

Full text document (pdf)

Citation for published version

Otero, Fernando E.B. and Freitas, Alex A. (2013) Improving the interpretability of classification rules discovered by an ant colony algorithm. In: 2013 Genetic and Evolutionary Computation Conference (GECCO'13), July 2013, Amsterdam, the Netherlands.

DOI

Link to record in KAR

http://kar.kent.ac.uk/34827/

Document Version

Author's Accepted Manuscript

Copyright & reuse
Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (eg Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

Versions of research
The version in the Kent Academic Repository may differ from the final published version.
Users are advised to check **http://kar.kent.ac.uk** for the status of the paper. **Users should always cite the published version of record.**

Enquiries
For any further enquiries regarding the licence status of this document, please contact:
researchsupport@kent.ac.uk

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at **http://kar.kent.ac.uk/contact.html**

KAR Kent Academic Repository

University of Kent

Improving the Interpretability of Classification Rules Discovered by an Ant Colony Algorithm

Fernando E. B. Otero
School of Computing
University of Kent, Chatham Maritime
Kent, ME4 4AG, UK
F.E.B.Otero@kent.ac.uk

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

ABSTRACT

The vast majority of Ant Colony Optimization (ACO) algorithms for inducing classification rules use an ACO-based procedure to create a rule in an one-at-a-time fashion. An improved search strategy has been proposed in the *c*Ant-Miner$_{\text{PB}}$ algorithm, where an ACO-based procedure is used to create a complete list of rules (ordered rules)—i.e., the ACO search is guided by the quality of a list of rules, instead of an individual rule. In this paper we propose an extension of the *c*Ant-Miner$_{\text{PB}}$ algorithm to discover a set of rules (unordered rules). The main motivation for discovering a set of rules is to improve the interpretation of individual rules and evaluate the impact on the predictive accuracy of the algorithm. We also propose a new measure to evaluate the interpretability of the discovered rules to mitigate the fact that the commonly-used model size measure ignores how the rules are used to make a class prediction. Comparisons with state-of-the-art rule induction algorithms and the *c*Ant-Miner$_{\text{PB}}$ producing ordered rules are also presented.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

ant colony optimization, data mining, classification, sequential covering, unordered rules

1. INTRODUCTION

Ant colony optimization (ACO) has been successfully applied to the classification task in data mining. 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 [19, 4, 25]. In essence, a classification problem consists of discovering a predictive model that represents the relationships between the predictor attribute values and the class (target) attribute values of data instances (also called examples, or cases). The discovered classification model is then used to classify—predict the class attribute value of—new examples (unseen during training) based on the values of their predictor attributes.

Since the introduction of Ant-Miner [18], the first ant colony rule induction algorithm for the discovery of a list of classification rules, many extensions have been proposed in the literature [7, 12]. The vast majority of these extensions follow the same overall design: they employ an ACO procedure to create a single classification rule in the form *IF* $<term_1$ *AND* ... *AND* $term_n>$ *THEN* $<$*class value*$>$ at each iteration of the algorithm, where the *IF* part corresponds to the antecedent of the rule and the *THEN* part corresponds to the consequent of the rule. The ACO-based construction procedure is repeated many times to produce a classification model (i.e., a list of classification rules). The strategy of creating one-rule-at-time, where the creation of each rule is an independent search problem, can lead to the problem of rule interaction—the creation of a rule affects the rules that can be created in subsequent iterations. A new strategy to mitigate the potential problem of rule interaction has been recently proposed in [17] and implemented in the *c*Ant-Miner$_{\text{PB}}$ algorithm. The main idea proposed in the new strategy is the use of an ACO procedure to create a complete list of rules and guide the search based on the quality of the whole list, therefore taking into account the interaction between the rules in the list.

This paper proposes an extension of the *c*Ant-Miner$_{\text{PB}}$ algorithm to create unordered rules. The main motivation is to improve the interpretation of individual rules. In an ordered set of rules (also referred to as list of rules), the effect (meaning) of a rule depends on all previous rules in the list, since a rule is only used if all previous rules do not cover the example. On the other hand, in an unordered set of rules, an example is shown to all rules and a single rule is used to make a prediction. We evaluate the new extension, called Unordered *c*Ant-Miner$_{\text{PB}}$, against state-of-the-art rule induction algorithms in terms of predictive accuracy using 18 publicly available datasets. We also propose a new measure to evaluate the size of the discovered model and present the results obtained by *c*Ant-Miner$_{\text{PB}}$ and the proposed Unordered *c*Ant-Miner$_{\text{PB}}$ algorithms.

The remainder of this paper is organized as follows. Section 2 presents a discussion of the new strategy implemented

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.
Copyright 2013 ACM 978-1-4503-1963-8/13/07 ...$15.00.

in the cAnt-Miner$_{\text{PB}}$ algorithm. The details of the proposed extension to create unordered rules are presented in Section 3. The computational results are presented in Section 4. Finally, Section 5 concludes this paper and presents future research directions.

2. BACKGROUND

The majority of ant colony classification algorithms follows a sequential covering strategy in order to create classification rules. The sequential covering strategy (also known as separate-and-conquer) is a commonly used strategy in machine learning to create a list/set of rules and it consists of two main steps: the algorithm creates a rule that classifies part of the available training examples (conquer step) and then removes the classified examples (separate step). This iterative process is repeated until (almost) all examples have been classified—i.e., there is a rule that classifies each of the available training examples. The use of the sequential covering strategy reduces the problem of creating a list/set of classification rules into a sequence of simpler problems, each requiring the creation of a single rule. In the case of ant colony classification algorithms, a single rule is created by an ACO procedure, which aims at searching for the best rule given a rule quality function. This is the strategy found in Ant-Miner [18], the first ACO-based rule induction algorithm, and its many extensions [7, 12]. One of the few exceptions is the Grammar Based Ant Programming (GBAP) algorithm [15, 16], which does not follow the sequential covering. In GBAP, each ant in the colony creates a rule using a context-free grammar and a list of rules is obtained using a niching approach—different ants compete to cover all training examples and the most accurate ones are used to compose a list of rules.

Recently, [17] proposed a new strategy to create classification rules using an ACO algorithm, implemented in the cAnt-Miner$_{\text{PB}}$ algorithm. The main motivation for the new strategy is to avoid the potential problem of rule interaction arising from the greedy nature of the sequential covering. Since rules are discovered in an one-at-a-time fashion in the sequential covering, the outcome of a rule (the examples removed by the rule) affects the remaining rules that can be created—the removal of the examples effectively changes the search space for the later iterations. As a result, Ant-Miner (and its variations) perform a greedy search for the list of best rules, using an ACO procedure to search for the best rule give a set of examples, and it is highly dependant in the order that rules are created. The strategy implemented in cAnt-Miner$_{\text{PB}}$ mitigates the problem of rule interaction by using an ACO procedure to search for the best list of rules. Therefore, an ant in cAnt-Miner$_{\text{PB}}$ creates a complete list of rules, while an ant in Ant-Miner creates a single rule.

In this paper we propose an extension to cAnt-Miner$_{\text{PB}}$ to discover unordered rules (set of rules) instead of ordered rules (list of rules), with the aim of improving the interpretability of the discovered rules. The discovery of unordered rule sets has been previously explored as extensions to the Ant-Miner algorithm only in [23, 14] to the best of our knowledge, although the search strategy of Ant-Miner and cAnt-Miner$_{\text{PB}}$ are very different—both of the Ant-Miner extensions in [23, 14] use an ACO procedure to create an individual rule. The motivation for extending the cAnt-Miner$_{\text{PB}}$ algorithm is to use an ACO procedure to search for the best set of rules, taking advantage of the improved strategy implemented in cAnt-Miner$_{\text{PB}}$.

3. CREATING UNORDERED RULE SETS

cAnt-Miner$_{\text{PB}}$ creates a list of rules (also referred to as ordered rules), where the order of rules plays an important role in the interpretation of individuals rules. When using a list of rules to classify a new example, each rule is tested sequentially—i.e., the example is shown to the first rule, then the second, and so forth—until a rule that covers[1] the example is found. Therefore, the effect (meaning) of a rule depends on all previous rules in the list, since a rule is only used if all previous rules do not cover the example. A simple example of this effect was given by [1]:

```
      If  feathers = yes  then  class = bird
else  if      legs = two  then  class = human
else  ...
```

The rule '`if legs=two then class=human`' cannot be correctly interpreted alone, since birds also have two legs. If we analyse the rule in the context of the list, an example is going to be tested against it only if the first rule is not used. In the case of birds, they will satisfy the first rule and be classified correctly as '`birds`'. This problem becomes more complex when we consider larger lists of rules.

An alternative to improve the interpretation of individual rules is to create a set of rules (also referred to as unordered rules), where the order of rules is not important. The use of a set of rules to classify an example consists of finding all rules that cover the example. If only one rule covers the example, the rule classifies the example; if multiple rules cover the example, a conflict resolution criteria is used to decide the final classification of the example. Rule conflict resolution criteria will be discussed in Subsection 3.4.

3.1 Unordered cAnt-Miner$_{\text{PB}}$

The main modification in order to create unordered rules is in the way ants create the set of rules. Instead of creating a rule and then determine its consequent based on the majority class value of the covered training examples, the Unordered cAnt-Miner$_{\text{PB}}$ introduces an extra loop to iterate over each class value. Therefore, an ant creates rules for each class value in turn, using as negative examples all the examples associated with different class values. Figure 1 presents the high-level pseudocode of the Unordered cAnt-Miner$_{\text{PB}}$ algorithm.

In summary, the unordered algorithm works as follows. An ant starts with an empty set of rules (outer *for loop*). Then, it creates rules for each of the class values (inner *for-all loop*). An ant creates rules for a specific class value until all examples of the class have been covered or the number of examples remaining for the class is below a given maximum threshold (inner *while loop*). After a rule is created and pruned, it is added to current set of rules and the training examples correctly covered by the rule are removed—i.e., the examples covered by the rule that are associated with the rule's class value (positive examples). The heuristic information for the current class value is recalculated at each iteration to reflect the changes in the predictive power of

[1]A rule covers an example when the example satisfies all the conditions in the antecedent of the rule.

```
Require: training examples
Ensure: best discovered rules
1. InitialisePheromones();
2. rules_gb ← ∅;
3. m ← 0;
4. while m < maximum iterations and not stagnation do
5.   rules_ib ← ∅;
6.   for n ← 1 to colony_size do
7.     rules_n ← ∅;
8.     for all class in classes do
9.       examples ← all training examples;
10.      while Count(examples, class) > maximum uncovered do
11.        ComputeHeuristicInformation(examples, class);
12.        rule ← CreateRule(examples, class);
13.        Prune(rule);
14.        examples ← examples − Covered(rule, class, examples);
15.        rules_n ← rules_n + rule;
16.      end while
17.    end for
18.    if Quality(rules_n) > Quality(rules_ib) then
19.      rules_ib ← rules_n;
20.    end if
21.  end for
22.  UpdatePheromones(rules_ib);
23.  if Quality(rules_ib) > Quality(rules_gb) then
24.    rules_gb ← rules_ib;
25.  end if
26.  m ← m + 1;
27. end while
28. return rules_gb;
```

Figure 1: High-level pseudocode of the Unordered cAnt-Miner$_{\text{PB}}$ algorithm.

the candidate terms due to the removal of the positive examples. The examples associated with different class values (negative examples) remain, even the ones that are covered by a rule—unlike the original (ordered) cAnt-Miner$_{\text{PB}}$ algorithm, where all covered examples are removed. After creating rules for all class values, the iteration-best set of rules is updated if the quality of the newly created set of rules is greater than the quality of the current iteration-best set. Once all ants have created a set of rules and the iteration-best set is determined, the pheromone values are updated using the iteration-best set and the global-best set of rules is updated, if the quality of the iteration-best set is greater than the quality of the global-best set (i.e., the best set of rules produced so far since the start of the search). The entire procedure (outer *while loop*) is repeated until either a maximum number of iterations has been reached or the search has converged. At the end, the best set of rules found is returned as the discovered set of rules.

Note that when an ant is creating a rule, the consequent of the rule (the class value predicted by the rule) is fixed. Therefore, the heuristic information and the dynamic discretization of continuous values take advantage of the class information and use a more accurate class-specific measure.

3.2 Class-Specific Heuristic Information

The heuristic information of each vertex v_i of the construction graph for the class value c is given by

$$\eta_{v_i}(c) = \frac{|Examples(v_i, c)|}{|Examples(v_i)|}, \tag{1}$$

where $|Examples(v_i, c)|$ is the number of training examples that satisfy the term (attribute-condition) represented by vertex v_i and that are associated with class value c, and $|Examples(v_i)|$ is the number of training examples that satisfy the term (attribute-condition) represented by vertex v_i. In other words, the heuristic information $\eta_{v_i}(c)$ corresponds to the fraction of training cases that are correctly covered by the term v_i with respect to the class value c.

3.3 Class-Specific Dynamic Discretisation

Continuous attributes represent a special case of vertices in the construction graph since they do not have a set of fixed intervals to define a complete term (attribute condition). When a vertice representing a continuous attribute is used, either for computing heuristic information or during the rule construction process, a dynamic discretisation procedure is employed to select a discrete interval in order to create a term. cAnt-Miner$_{\text{PB}}$ uses an entropy-based procedure, which does not require the class information a priori. Since in the unordered extension the class value is available to the discretization procedure, we use the Laplace accuracy as a criteria to select a threshold value to discretize a continuous attribute as follows.

A threshold value t in the domain of the continuous at-

tribute x dynamically generates two intervals: $x \leq t$ and $x > t$. The best threshold value is the value t that maximises the interval accuracy in the set of examples S regarding the class value c, given by

$$max(Lap(c, S_{x \leq t}), Lap(c, S_{x>t})) \;\; \forall\, t \in D_x\,, \qquad (2)$$

where $S_{x \leq t}$ is the set of examples that satisfy the interval $x \leq t$, $S_{x>t}$ is the set of examples that satisfy the interval $x > t$ and D_x are the values in the domain of attribute x. The Laplace accuracy of an interval is given by

$$Lap(c, E) = \frac{|E_c| + 1}{|E| + k}\,, \qquad (3)$$

where E is the set of examples in the interval, $|E_c|$ is the number of examples in E that are associated with the class value c, $|E|$ is the number of examples in E and k is the number of different values in the domain of the class attribute.

After selecting the best threshold value t, a term for the continuous attribute x is created based on the Laplace accuracy of the two intervals generated, given by

$$term_x = \begin{cases} x \leq t \text{ if } Lap(c, S_{x \leq t}) > Lap(c, S_{x>t}) \\ x > t \text{ if } Lap(c, S_{x \leq t}) < Lap(c, S_{x>t}) \end{cases}\,. \qquad (4)$$

3.4 Using a Set of Rules to Classify Examples

As aforementioned, in order to classify an example using a set of rules, all rules that cover the example are identified. The prediction of the class value of an example leads to one of the following scenarios:

1. *None of the rules covers the example*: the example is assigned the default class value, which corresponds to the majority class value of the training set;

2. *Only one rule covers the example*: the example is assigned the class value predicted by the rule;

3. *Multiple rules predicting the same class value cover the example*: the example is assigned the class value predicted by the rules;

4. *Multiple rules predicting different class values cover the example*: a conflict resolution strategy is used to determine the predicted class value. There are mainly two strategies: (i) use the rule with the highest quality (rule selection strategy); (ii) combine (sum up) the class distribution of covered examples amongst the class values of each rule and predict the majority class value in the sum (rule aggregation strategy), as proposed in [1].

Let us consider an example in a 2-class problem `{Y,N}` that is covered by rules `R1`⇒`Y [7,0]`, `R2`⇒`Y [4,0]` and `R3`⇒`N [1,5]` (the values between squared brackets correspond to the class values distribution of the covered examples). Since `R3` predicts a class value different from the one predicted by `R1` and `R2`, we have a conflict. If we use a class rule aggregation strategy to resolve the conflict, we first sum up the class distribution of the rules (which is `[12,5]`) and then predict the most common value in the summed distribution (`Y`). In this example the use of a rule selection strategy would produce the same prediction (`Y`), based on the assumption that rule `R1` is the rule with the highest quality.

Note that each of the conflict resolution strategies has a different impact in the interpretability of the discovered rules. In the case of the rule selection strategy, a single rule is responsible for the classification of an example—the rule with the highest quality—regardless if multiple rules cover the example or not; in the case of the rule aggregation strategy, (potentially) multiple rules are responsible for the classification of an example. While in the former case the user has to analyse a single rule in order to interpret a particular prediction, several rules should be analysed in order to interpret a particular prediction in the latter case. Hence, the rule selection strategy usually leads to simpler interpretations.

4. COMPUTATIONAL RESULTS

We divided the computational results in three sets of experiments. In the first set of experiments, we evaluated different configurations of the proposed Unordered *c*Ant-Miner$_{\text{PB}}$. The aim is to determine the effects of the different conflict resolution strategies, and also the effects of both the dynamic rule quality function selection and the error-based list quality function [13], in the performance of the algorithm. In the second set of experiments, we evaluated the Unordered *c*Ant-Miner$_{\text{PB}}$ configuration against state-of-the-art rule induction classification algorithms in terms of predictive accuracy. In the third set of experiments, we compared the interpretability of the rules discovered by the original *c*Ant-Miner$_{\text{PB}}$ against the rules discovereds by the proposed Unordered *c*Ant-Miner$_{\text{PB}}$ algorithm.

In all the experiments, the performance of a classification algorithm is measured using a tenfold cross-validation procedure, which consists of dividing a dataset into ten stratified partitions (i.e., each partition contains a similar number of examples and class distribution). For each partition, the classification algorithm is run using the remaining nine partitions as training data and the predictive accuracy of the discovered model is evaluated in the unseen (hold-out) partition. The final value of the predictive accuracy for a particular dataset is the average value obtained across the ten partitions.

4.1 Evaluating different configurations

We evaluated 8 different configurations of the proposed Unordered *c*Ant-Miner$_{\text{PB}}$ combining both conflict resolution strategies with both the dynamic rule quality function selection and the error-based list quality function extensions proposed in [13]: 2 different conflict resolution strategies (rule selection and rule aggregation), 2 rule quality function selection approaches (static and dynamic), 2 list quality functions (predictive accuracy and error-based); a total of 8 configurations, varying those 3 general 'parameters'.

In this first set of experiments, which can be considered as a parameter tuning step, we selected 8 dataset from the UCI Machine Learning repository [5], namely *automobile*, *blood-transfusion*, *ecoli*, *statlog heart*, *hepatitis*, *horse-colic*, *voting records* and *zoo*. For each of the datasets we carried out a tenfold cross-validation procedure using the default parameters of *c*Ant-Miner$_{\text{PB}}$ [17]: *colony size* of 5, *maximum*

Table 1: Summary of the datasets used in the second set of experiments.

dataset	# attributes nom.	cont.	# classes	# examples
annealing	29	9	6	898
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
credit-g	13	7	2	1000
cylinder-bands	16	19	2	540
dermatology	33	1	6	366
glass	0	9	7	214
heart-c	6	7	5	303
heart-h	6	7	5	294
ionosphere	0	34	2	351
iris	0	4	3	150
liver-disorders	0	6	2	345
parkinsons	0	22	2	195
pima	0	8	2	768
wine	0	13	3	178

number of iterations of 500, *evaporation factor* of 0.9 (evaporation rate equal to $1-$*factor*). Given the stochastic nature of the algorithm, each of the Unordered cAnt-Miner$_{\text{PB}}$ configurations was run 10 times for every dataset.

Using a separate set of datasets just for parameter tuning, as in this Section, has the advantage that, after finding good parameter settings in this set of 8 datasets, we can evaluate the generalisation ability of those settings in a different set of datasets (Section 4.2). Such generalisatin ability is important in the classification task of data mining.

The results of these experiments showed that the use of the error-based list quality function in the Unordered cAnt-Miner$_{\text{PB}}$ algorithm had a negative impact in the predictive accuracy of the discovered rules. Interesting, this is the opposite effect observed in the original cAnt-Miner$_{\text{PB}}$, where an improvement in predictive accuracy is observed when the error-based list quality function is used [13]. The use of the dynamic rule quality function selection led to an improvement in predictive accuracy, independently of the conflict resolution strategy. As a result of these experiments, we determined that the dynamic rule quality function selection and the predictive accuracy as the list quality function are more suitable for the Unordered cAnt-Miner$_{\text{PB}}$ algorithm. While the rule selection conflict resolution strategy led to an improvement in the predictive accuracy compared to the configuration using the rule aggregation strategy, we did not select a specific strategy at this stage, since they have a different impact in the interpretability of the discovered rules. Therefore, we carried out the remaining of the experiments using both rule selection and rule aggregation conflict resolution strategies.

Table 2: The 4 rule induction algorithms used in the second set of experiments in addition to the cAnt-Miner$_{\text{PB}}$ and Unordered cAnt-Miner$_{\text{PB}}$ algorithms.

name	ref.	description
Unordered CN2	[1]	A version of the well-known CN2 rule induction algorithm that creates unordered rules using a beam search procedure to create a classification rule
C4.5rules	[21]	A rule induction algorithm that extracts a set of classification rules from an unpruned decision tree created by the well-known C4.5 algorithm [21, 20]
PART	[6, 25]	A rule induction algorithm that combines a sequential covering strategy with a decision tree induction procedure to create a rule
JRip	[25]	Weka's implementation of the RIPPER [2] algorithm, a rule induction algorithm that employs a global optimisation step in order to produce a list of rules, which takes into account both the quality and length of the rules

4.2 Comparisons with state-of-the-art classification algorithms

The computational experiments comparing the proposed Unordered cAnt-Miner$_{\text{PB}}$[2] algorithm against state-of-the-art rule induction algorithms were carried out using a set of 18 publicly available datasets from the UCI Machine Learning repository [5]—a summary of the datasets used in the experiments is presented in Table 1. The second and third columns give the number of nominal and continuous attributes, respectively, for each dataset. The other column names are self-explained.

We have selected 4 rule induction algorithms, in addition to the cAnt-Miner$_{\text{PB}}$[3] and Unordered cAnt-Miner$_{\text{PB}}$ algorithms. The details of the selected algorithms are given in Table 2. All algorithms were used with the default parameter values proposed by their corresponding authors—both cAnt-Miner$_{\text{PB}}$ and Unordered cAnt-Miner$_{\text{PB}}$ were used with the same parameter values from the first set of experiments (see Subsection 4.1). We used 2 configurations for the Unordered cAnt-Miner$_{\text{PB}}$: one using the rule selection conflict resolution strategy (denoted as U-cAM$_{\text{PB}}$ [S]) and one using

[2]The source-code and binaries of the new Unordered cAnt-Miner$_{\text{PB}}$ algorithm are available for download at http://sourceforge.net/projects/myra.

[3]We used the cAnt-Miner$_{\text{PB}}$ algorithm with the error-based list quality function, as suggested by [13].

Table 3: Average predictive accuracy (*average* [*standard error*]) in %, measured by tenfold cross-validation. The value of the most accurate algorithm for a given dataset is shown in boldface.

dataset	U-cAM$_{\text{PB}}$ [S]	U-cAM$_{\text{PB}}$ [A]	Unordered CN2	C4.5rules	PART	JRip	cAnt-Miner$_{\text{PB}}$
annealing	**97.70 [0.13]**	95.53 [0.23]	88.10 [0.84]	94.22 [0.62]	94.88 [0.98]	94.43 [0.81]	97.34 [0.11]
balance-scale	77.82 [0.20]	**88.85 [0.21]**	79.34 [1.39]	74.87 [1.16]	77.12 [1.40]	72.95 [1.92]	76.26 [0.29]
breast-l	74.02 [0.19]	73.59 [0.35]	73.44 [1.81]	68.56 [1.93]	68.94 [1.80]	69.26 [2.04]	**75.27 [0.35]**
breast-tissue	65.91 [0.67]	**67.58 [0.72]**	63.35 [4.51]	66.16 [2.97]	64.36 [3.63]	60.18 [3.35]	64.16 [0.98]
breast-w	**95.20 [0.12]**	95.02 [0.19]	93.15 [1.51]	94.18 [1.14]	94.19 [1.12]	93.66 [1.42]	94.34 [0.16]
credit-a	84.93 [0.28]	85.68 [0.16]	82.93 [1.49]	85.53 [1.53]	83.33 [1.04]	**86.52 [1.10]**	86.10 [0.23]
credit-g	72.84 [0.32]	71.08 [0.22]	**74.60 [0.98]**	71.60 [0.92]	70.60 [1.49]	72.20 [1.07]	73.67 [0.28]
cylinder-bands	72.06 [0.42]	71.83 [0.41]	**76.85 [2.05]**	76.48 [2.56]	72.41 [2.23]	68.70 [2.33]	72.36 [0.35]
dermatology	90.54 [0.32]	90.08 [0.44]	87.43 [1.60]	93.45 [1.22]	**94.26 [1.17]**	88.01 [2.25]	92.40 [0.40]
glass	69.46 [0.81]	68.24 [0.81]	65.73 [3.97]	68.63 [1.70]	72.81 [3.42]	65.71 [3.74]	**73.11 [0.61]**
heart-c	56.66 [0.50]	**57.13 [0.52]**	55.81 [2.27]	53.12 [1.92]	53.83 [1.33]	53.50 [1.52]	55.21 [0.41]
heart-h	64.64 [0.45]	64.02 [0.33]	60.90 [1.20]	63.31 [1.40]	63.64 [1.58]	63.93 [1.29]	**65.92 [0.35]**
ionosphere	89.44 [0.33]	89.85 [0.35]	**91.73 [2.77]**	90.85 [2.59]	90.59 [2.00]	87.45 [2.64]	89.95 [0.23]
iris	94.33 [0.20]	93.87 [0.19]	92.66 [1.55]	95.32 [1.42]	93.33 [1.99]	**96.00 [1.09]**	93.13 [0.26]
liver-disorders	**69.78 [0.64]**	69.49 [0.46]	66.37 [1.52]	64.90 [3.21]	62.70 [3.40]	66.34 [2.80]	66.71 [0.41]
parkinsons	**87.75 [0.45]**	85.28 [0.62]	86.66 [1.38]	83.49 [2.23]	86.05 [2.47]	84.53 [2.55]	87.42 [0.50]
pima	74.55 [0.30]	74.32 [0.15]	73.42 [1.13]	74.32 [1.73]	71.73 [1.71]	73.55 [1.63]	**74.67 [0.20]**
wine	95.46 [0.30]	**95.48 [0.32]**	93.80 [1.54]	91.03 [2.05]	91.54 [1.52]	92.68 [2.09]	94.51 [0.31]

Table 4: Statistical test results of the algorithms' average predictive accuracies according to the non-parametric Friedman test with the Hommel's post-hoc test. Statistically significant differences at the $\alpha = 0.5$ level are shown in boldface.

Algorithm	Avg. Rank	p-value	Hommel
U-cAM$_{\text{PB}}$ [S] (control)	2.72	–	–
cAnt-Miner$_{\text{PB}}$	2.89	0.8169	0.05
U-cAM$_{\text{PB}}$ [A]	3.28	0.4404	0.025
PART	**4.55**	**0.0108**	**0.0166**
Unordered CN2	**4.55**	**0.0108**	**0.0125**
C4.5rules	**4.83**	**0.0034**	**0.0100**
JRip	**5.17**	**6.8E-4**	**0.0083**

the rule aggregation conflict resolution strategy (denoted as U-cAM$_{\text{PB}}$ [A]).

Table 3 presents the results concerning the predictive accuracy, where the higher the value the better the algorithm performance in terms of accuracy, measured as the average value obtained by an algorithm at the end of the tenfold cross-validation procedure. In the case of stochastic algorithms cAnt-Miner$_{\text{PB}}$ and Unordered cAnt-Miner$_{\text{PB}}$, the average value is computed over 10 executions of the tenfold cross-validation procedure (i.e., each algorithm is run 10×10 times for each dataset); the remaining algorithms are deterministic and the average is computed over a single run of the tenfold cross-validation (i.e., each algorithm is run 1×10 times for each dataset). Table 4 presents the statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test [3, 8]: the first column corresponds to the algorithm's name, the second column correspond to the average rank—where the lower the rank the better the algorithm's performance—obtained in the Friedman test, the third column shows the p-value of the statistical test when the average rank is compared to the average rank of the algorithm with the best rank (the 'control' algorithm), and the fourth column corresponds to the Hommel's critical value. A row is shown in boldface when there is a statistically significant difference at the 5% significance level between the average ranks of an algorithm and the control algorithm, determined by the fact that the p-value is lower than Hommel's critical value—i.e., it corresponds to the case where the control algorithm is significantly better than the algorithm in that row.

The Unordered cAnt-Miner$_{\text{PB}}$ using the rule selection conflict resolution strategy (denoted as U-cAM$_{\text{PB}}$ [S]) achieved the best average rank, outperforming state-of-the-art rule induction algorithms with statistically significant differences, namely PART, Unordered CN2, C4.5rules and JRip. The use of the rule selection conflict resolution strategy led to an improvement in predictive accuracy, as in our initial experiments (Subsection 4.1), although the differences are not statistically significant when compared to the use of the rule aggregation conflict resolution strategy. While there are no statistically significant differences between cAnt-Miner$_{\text{PB}}$ and Unordered cAnt-Miner$_{\text{PB}}$ algorithms, the results obtained by the proposed Unordered cAnt-Miner$_{\text{PB}}$ are pos-

itive, overall: the discovery of unordered rules explicitly improves their interpretability (i.e., a particular rule has a modular meaning independent of the others rules), without sacrificing the predictive accuracy.

4.3 Interpretability of the discovered rules

In order to quantify the interpretability of the discovered rules, we propose a new measure, called *prediction-explanation size*. Before presenting the details of how this measure is calculated, it is worth discussing the problems with the commonly-used model size as a measure of comprehensibility (or interpretability). The model size measure, usually determined by the number of rules and the size of rules present in the list/set of rules, ignores how the rules are used to make class predictions—e.g., if a single or multiple rules are needed to classify an example. In addition, there is evidence showing that, in some applications, larger models were considered by users as more comprehensible than smaller ones, since they contained more informative attributes to the user [11, 10]. An empirical study tested the assumption that smaller models are more comprehensible to users [9]. The findings of this study indicate that the comprehensibility of the model from a user perspective tends to increase in line with the size of the model. Additionally, the concept of 'small' or 'large' is subjective—e.g., in [22] it is reported that users found that a model with 41 rules was considered too large to be analysed by a user, while in [24] a user analysed 29,050 rules and identified a subset of 220 interesting rules. Therefore, the model size—either measured as the number of rules or the total number of attribute-conditions of the rules—is not an adequate indicator of the comprehensibility of a classification model.

We define the *prediction-explanation size* as the average number of attributes-conditions (terms) that are evaluated in the model in order to predict the class value of an example, where the average is computed over all examples being classified in the test set. The rationale behind the *prediction-explanation size* measure is that it provides an estimate of the number of attributes-conditions that a user has to analyse in order to interpret a model's prediction and those attribute-conditions can be regarded as an explanation for the class prediction. In the case of the original *c*Ant-Miner$_{PB}$ algorithm, which produces an ordered list of rules, the prediction-explanation size is calculated taking into account all rules that are evaluated in order to make a prediction. E.g., if there are 3 rules in the list, each composed by 3 attributes-conditions, and the second rule is used to make a prediction, the prediction-explanation size is the sum of the attributes-conditions of the first and second rules. While the first rule is not directly involved in the prediction, it is involved indirectly in the prediction, since the second rule is only evaluated if the first rule is not used (i.e., its attribute-conditions evaluate to false).

In the case of the Unordered *c*Ant-Miner$_{PB}$ algorithm, which produces an (unordered) set of rules, the prediction-explanation size depends on the conflict resolution strategy used. When the rule selection strategy is used, only one rule is responsible for the prediction and therefore, the prediction-explanation size is the number of attribute-conditions of the rule. When the rule aggregation strategy is used, the prediction-explanation size is the sum of the attribute-conditions of the rules that cover the example, since all rules that cover the example contribute to the final prediction. It should be noted that in the Unordered *c*Ant-Miner$_{PB}$ algorithm, each rule does have a modular meaning independent of the others, regardless of the conflict resolution strategy used, since the order of the rules is not important.

Table 5: Average number of attribute-conditions (terms) involved in the classification of an example. The value of the algorithm with the lowest average for a given dataset is shown in boldface.

dataset	U-*c*AM$_{PB}$ [S]	U-*c*AM$_{PB}$ [A]	*c*Ant-Miner$_{PB}$
annealing	**2.28 [0.01]**	5.65 [0.12]	8.00 [0.19]
balance-scale	**1.40 [0.01]**	5.09 [0.06]	4.94 [0.06]
breast-l	**2.74 [0.05]**	8.94 [0.12]	5.30 [0.34]
breast-tissue	**1.47 [0.01]**	4.35 [0.06]	3.89 [0.06]
breast-w	**1.40 [0.01]**	7.11 [0.08]	3.91 [0.14]
credit-a	**2.76 [0.06]**	8.73 [0.17]	6.61 [0.39]
credit-g	**2.83 [0.05]**	13.11 [0.15]	15.77 [0.36]
cylinder-bands	**2.78 [0.04]**	12.38 [0.15]	27.71 [0.77]
dermatology	**3.46 [0.06]**	14.11 [0.29]	14.91 [0.33]
glass	**2.39 [0.04]**	5.72 [0.06]	6.93 [0.15]
heart-c	**3.12 [0.03]**	14.50 [0.26]	11.89 [0.38]
heart-h	**2.88 [0.05]**	8.12 [0.04]	5.81 [0.21]
ionosphere	**3.05 [0.03]**	7.07 [0.10]	6.51 [0.14]
iris	**1.38 [0.01]**	3.09 [0.09]	2.80 [0.04]
liver-disorders	**2.38 [0.02]**	7.48 [0.06]	12.44 [0.27]
parkinsons	**1.53 [0.03]**	4.78 [0.06]	3.64 [0.07]
pima	**2.19 [0.03]**	6.38 [0.03]	5.24 [0.07]
wine	**1.13 [0.01]**	3.27 [0.05]	3.04 [0.03]

Table 5 presents the results (*average* [*standard error*]) concerning the *prediction-explanation size* of the models discovered by the *c*Ant-Miner$_{PB}$ and Unordered *c*Ant-Miner$_{PB}$ algorithms, the lower the value the better the algorithm performance. The advantage of unordered rules combined with the rule selection strategy (denoted as U-*c*AM$_{PB}$ [S]) are clear: in all the datasets, it has the lowest number of attribute-conditions involved in the classification of an example. Overall, these results are positive: we were able to improve the interpretablity of the rules (measured as the *prediction-explanation size*) by discovering unordered rules, with no negative impact on the predictive accuracy. The Unordered *c*Ant-Miner$_{PB}$ with the rule selection strategy was the algorithm that achieved the best rank in terms of predictive accuracy and the best value for the prediction-explanation size measure.

5. CONCLUSION

In this paper we have proposed an extension to the *c*Ant-Miner$_{PB}$ in order to discover unordered rules, called Unordered *c*Ant-Miner$_{PB}$. The main motivation is to improve the interpretation of individual rules. In an ordered list of rules, the effect (meaning) of a rule depends on all previous rules in the list, since a rule is only used if all previous rules do not cover the example. On the other hand, in an unordered set of rules, an example is shown to all rules and, depending on the conflict resolution strategy, a single rule is used to make a prediction.

We compared the proposed Unordered *c*Ant-Miner$_{PB}$ algorithm against state-of-the-art rule induction algorithms

in 18 publicly available datasets. The Unordered *c*Ant-Miner$_{\text{PB}}$ algorithm achieved the best results in terms of predictive accuracy, outperforming state-of-the-art rule induction algorithms with statistically significant differences. We also proposed a new measure to characterise the interpretability of the discovered rules. Our results show that the predictions made by an unordered set of rules are potentially easier to be interpreted by a user, due to the nature of unordered rules (i.e., each rule has a modular meaning independent of the others) and there are less attribute-conditions involved in the predictions.

While in this work we have evaluated different rule quality and list quality functions, the Unordered *c*Ant-Miner$_{\text{PB}}$ algorithm's more specific parameters were not optimised, since we focused in comparing the differences between ordered and unordered rules. It might be possible to further improve the algorithm evaluating different parameter settings. The evaluation of different conflict resolution strategies can also lead to improvements to the predictive accuracy of the discovered rules.

6. REFERENCES

[1] P. Clark and R. Boswell. Rule Induction with CN2: Some Recent Improvements. In *Machine Learning – Proceedings of the Fifth European Conference (EWSL-91)*, pages 151–163, 1991.

[2] W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115âĂŞ–123. Morgan Kaufmann, 1995.

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

[4] U. Fayyad, G. Piatetsky-Shapiro, and P. Smith. From data mining to knowledge discovery: an overview. In U. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery & Data Mining*, pages 1–34. MIT Press, 1996.

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

[6] E. Frank and I. Witten. Generating Accurate Rule Sets Without Global Optimization. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[7] A. Freitas, R. Parpinelli, and H. Lopes. Ant colony algorithms for data classification. In *Encyclopedia of Information Science and Technology*, volume 1, pages 154–159. Information Science Reference, 2nd edition, 2008.

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

[9] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51:141–154, 2011.

[10] H. A. N. Lavesson. User-oriented assessment of classification model understandability. In *Proceedings of the 11th Scandinavian Conference on Artificial Intelligence (SCAI)*, pages 11–19. IOS Press, 2011.

[11] N. Lavrac. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16:3–23, 1999.

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

[13] M. Medland, F. Otero, and A. Freitas. Improving the *c*Ant-Miner$_{\text{PB}}$ Classification Algorithm. In M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Groß, and T. Stützle, editors, *Swarm Intelligence*, volume 7461 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 2012.

[14] C. Nalini and P. Balasubramanie. Discovering Unordered Rule Sets for Mixed Variables Using an Ant-Miner Algorithm. *Data Science Journal*, 7:76–87, 2008.

[15] J. Olmo, J. Romero, and S. Ventura. Using Ant Programming Guided by Grammar for Building Rule-Based Classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41:1585–1599, 2011.

[16] J. Olmo, J. Romero, and S. Ventura. Classification rule mining using ant programming guided by grammar with multiple pareto fronts. *Soft Computing*, 16:2143–2163, 2012.

[17] F. Otero, A. Freitas, and C. Johnson. A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):64–76, 2013.

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

[19] G. Piatetsky-Shapiro and W. Frawley. *Knowledge Discovery in Databases*. AAAI Press, 1991.

[20] J. Quinlan. Improved Use of Continuous Attributes in C4.5. *Artificial Intelligence Research*, 7:77–90, 1996.

[21] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[22] M. Schwabacher and P. Langley. Discovering communicable scientific knowledge from spatio-temporal data. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, pages 489–496. Morgan Kaufmann, 2001.

[23] J. Smaldon and A. Freitas. A new version of the ant-miner algorithm discovering unordered rule sets. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 43–50, 2006.

[24] S. Tsumoto. Clinical knowledge discovery in hospital information systems: two case studies. In *Proceedings of European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD-2000)*, pages 652–656. Springer, 2000.

[25] H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.