[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

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

Fernando E. B. Otero F.E.B.Otero@kent.ac.uk
University of Kent, Chatham Maritime, United Kingdom

Alex A. Freitas A.A.Freitas@kent.ac.uk
University of Kent, Canterbury, United Kingdom

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 cAnt-Miner$_{\mathrm{PB}}$ 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 cAnt-Miner$_{\mathrm{PB}}$ algorithm to discover a set of rules (unordered rules). The main motivations for this work are to improve the interpretation of individual rules by discovering a set of rules and to 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, support vector machines and the cAnt-Miner$_{\mathrm{PB}}$ producing ordered rules are also presented.

Keywords

Ant colony optimization, data mining, classification, sequential covering, unordered rules, comprehensibility.

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 (Piatetsky-Shapiro and Frawley, 1991; Fayyad et al., 1996; Witten and Frank, 2011). 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 (Parpinelli et al., 2002), the first ant colony rule induction algorithm for the discovery of a list of classification rules, many extensions have been proposed in the literature (Freitas et al., 2008; Martens et al., 2011). 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

[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

to the antecedent of the rule and the *THEN* part corresponds to the consequent of the rule. The ACO-based rule 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-a-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 (Otero et al., 2013) and implemented in the cAnt-Miner$_{\mathrm{PB}}$ algorithm. The main idea proposed in the new strategy is the use of an ACO procedure to create a complete list of rules, guiding the search based on the quality of the whole list, and therefore, taking into account the interaction between the rules in the list.

This paper proposes an extension of the cAnt-Miner$_{\mathrm{PB}}$ 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 or a subset of rules that covers the example is used to make a prediction. The proposed unordered extension, called Unordered cAnt-Miner$_{\mathrm{PB}}$, is evaluated against state-of-the-art rule induction algorithms and support vector machines in terms of predictive accuracy. We also propose a new measure to evaluate the size of the discovered model and present the results comparing both cAnt-Miner$_{\mathrm{PB}}$ and Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithms against state-of-the-art rule induction algorithms.

This paper is an extended version of a previous conference paper (Otero and Freitas, 2013), providing a more detailed description of the proposed approach and also extending the computational results in four ways: we extend the number of datasets from 18 in our previous conference paper to 32 in the current paper, report predictive accuracy for two more rule induction algorithms (PSO/ACO2 and BioHEL), report model size results based on the proposed *prediction-explanation size* measure for all the 9 rule induction algorithms evaluated in the current paper (whilst model size results were reported for only 3 algorithms in our previous conference paper), and report predictive accuracy for a SVM classifier, including statistical significance analysis of the results.

The remainder of this paper is organized as follows. Section 2 presents a discussion of the new strategy implemented in the cAnt-Miner$_{\mathrm{PB}}$ 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 (one-rule-at-a-time) 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 well-performing 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 (Parpinelli et al., 2002), the first ACO-based rule induction algorithm, and its many extensions (Freitas et al., 2008; Martens et al., 2011). One of the few exceptions is the Grammar Based Ant Programming (GBAP) algorithm (Olmo et al., 2011, 2012), 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.

As aforementioned in the Introduction (Section 1), the cAnt-Miner$_{\mathrm{PB}}$ (Otero et al., 2013) algorithm implements a new strategy to create classification rules, where the main motivation is to avoid the potential problem of rule interaction arising from the greedy nature of the sequential covering. The rule interaction problem is the result of the use of a strategy where rules are discovered in an one-at-a-time fashion: the outcome of a rule (the examples removed by the rule) has an impact on the rules that can be created in subsequent iterations, since 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 given a set of examples, and it is highly dependant on the order that rules are created. The strategy implemented in cAnt-Miner$_{\mathrm{PB}}$ mitigates the problem of rule interaction by using an ACO procedure to search for the best list of rules. In cAnt-Miner$_{\mathrm{PB}}$, an ant creates an entire list of rules, while in Ant-Miner an ant creates a single rule. This emphasises the differences in their ACO search strategies: in Ant-Miner (and its extensions), the ACO search is guided by the quality of the individual rules, as in (traditional) sequential covering algorithms; the ACO search in cAnt-Miner$_{\mathrm{PB}}$ algorithm, however, is not concerned by the quality of the individual rules as long as the quality of the entire list of rules is improving, since the entire list is created at once and the best list is chosen to guide the search.

This paper presents an extension to cAnt-Miner$_{\mathrm{PB}}$ 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 (Smaldon and Freitas, 2006; Nalini and Balasubramanie, 2008) to the best of our knowledge, although the search strategy of Ant-Miner and cAnt-Miner$_{\mathrm{PB}}$ are very different—both of the Ant-Miner extensions in (Smaldon and Freitas, 2006; Nalini and Balasubramanie, 2008) use an ACO procedure to create an individual rule. The motivation for extending the cAnt-Miner$_{\mathrm{PB}}$ 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$_{\mathrm{PB}}$.

2.1 An overview of the cAnt-Miner$_{\mathrm{PB}}$ algorithm

As we mentioned, the cAnt-Miner$_{\mathrm{PB}}$ is an ACO classification algorithm that employs a different search strategy than Ant-Miner and the vast majority of ACO classification algorithms. Rather than creating a list of rules by searching for the best individual rule at each iteration as Ant-Miner does, cAnt-Miner$_{\mathrm{PB}}$ instead searches for the best list of rules. This difference in the search strategy is reflected in the list creation process: in cAnt-Miner$_{\mathrm{PB}}$, each ant creates an entire list of rules, whereas in Ant-Miner each ant creates an individual rule. When an iteration of the ACO procedure is completed (i.e., after each ant of the colony creates a candidate list of rules), the best list of rules is used to update the pheromone values. The pheromone update will guide the search to more

```
Input: training examples
Output: best discovered list of rules
InitialisePheromones();
$list_{gb} \leftarrow \{\}$;
$m \leftarrow 0$;
while $m$ < maximum iterations and not stagnation do
    $list_{ib} \leftarrow \{\}$;
    for n ← 1 to colony_size do
        examples ← all training examples;
        $list_n \leftarrow \{\}$;
        while |examples| > maximum uncovered do
            ComputeHeuristicInformation(examples);
            rule ← CreateRule(examples);
            $Prune$(rule);
            examples ← examples − Covered(rule, examples);
            $list_n \leftarrow list_n$ + rule;
        end while
        if $Quality(list_n) > Quality(list_{ib})$ then
            $list_{ib} \leftarrow list_n$;
        end if
    end for
    $UpdatePheromones(list_{ib})$;
    if $Quality(list_{ib}) > Quality(list_{gb})$ then
        $list_{gb} \leftarrow list_{ib}$;
    end if
    $m \leftarrow m + 1$;
end while
return $list_{gb}$;
```

Figure 1: High-level pseudocode of the cAnt-Miner$_{\mathrm{PB}}$ algorithm (Otero et al., 2013).

prominent regions of the search space, affecting the candidate lists created in the future iterations. The final (discovered) list of rules is the best candidate list of rules created throughout the execution of the algorithm, based on a list quality function.

The high-level pseudocode of cAnt-Miner$_{\mathrm{PB}}$ is presented in Figure 1. At each iteration (outer *while loop*), an ant in the colony starts with an empty list and the full training set. Each ant then creates a rule probabilistically using the pheromone values and heuristic information, prunes the rule, and removes all of the covered examples from the training set. An ant repeats these steps to create a rule until the number of remaining examples reaches a predefined minimum threshold. The list creation process (inner *while loop*) can be seen as a sequential covering but without an optimisation step: at each iteration a rule is created probabilistically and added to the list of rules regardless of its quality. At no point are rules compared to each other and the rule quality is not calculated during the rule creation process—the only time the rule quality function is used is during the pruning step. The number of rules that an ant creates depends on the available training examples at each iteration of the list creation process, which in turn varies according to the number of examples covered by the previous rules created

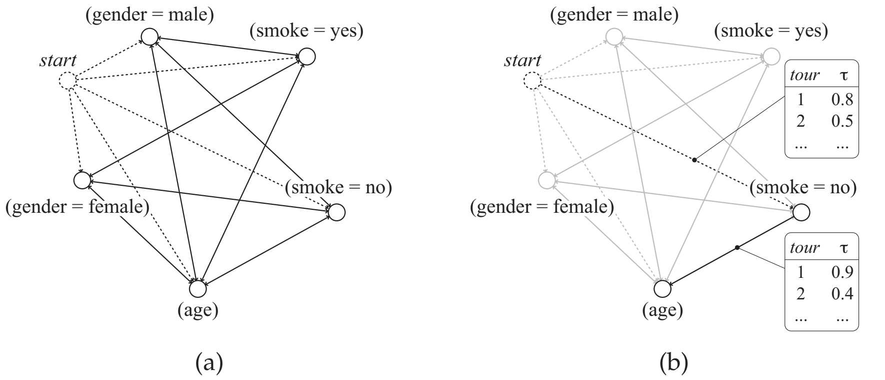

Figure 2: The construction graph of cAnt-Miner$_{\mathrm{PB}}$ when pheromone values are associated with edges. In (a) the construction graph with a dummy '*start*' vertex, corresponding to the starting point for creating the rule antecedent and its purpose is to associate pheromone values with the edge leading to the first vertex of the antecedent of a rule; (b) the multiple pheromone values in the edges connecting vertices ('*start*' → '$smoke = no$' → '*age*') are highlighted.

by the ant. As a consequence, the list creation process is flexible with respect to the size of the lists: there is no pre-defined number of rules that an ant has to create and the lists created by the ants might have different sizes.

In order to mitigate the problem of rule interaction, the order (sequence) that ants create the rules is indirectly encoded in the pheromone values. During the list creation process, an ant uses different pheromone values depending of the position of the current rule being created. This is achieved by extending the pheromone matrix to include a *tour* identification to represent the position of the rule (e.g., 1 for the first rule, 2 for the second and so forth). Each *edge*$_{ij}$ of the construction graph, connecting vertices v_i and v_j, has multiple pheromone values associated, one value for each rule position. This is illustrated in Figure 2. When an ant is creating the i-th rule, it uses the pheromone values associated with the i-th entry of every *edge*$_{ij}$. A similar process occurs during the pheromone update, where an ant updates the pheromone values of the edges used to create the rules according to the rule's position in the candidate list of rules.

After an iteration is completed, the iteration-best list of rules is used to update the pheromones and the global-best list is updated, if the quality of the iteration-best list is greater than the quality of the current global-best list. This entire process is repeated until either the maximum number of iterations is reached or the ACO search stagnates.

3 Creating Unordered Rule Sets

cAnt-Miner$_{\mathrm{PB}}$ 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

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

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 Clark and Boswell (1991):

```
      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`'. The problem of analysing rules 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 and overlaps might occur (i.e., different rules covering the same 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 (the predicted class value) of the example. Rule conflict resolution criteria will be discussed in Subsection 3.4.

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

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$_{\mathrm{PB}}$ 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 3 presents the high-level pseudocode of the Unordered cAnt-Miner$_{\mathrm{PB}}$ 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*). In order to create a rule, ants probabilistically select terms to be added to their current partial rule based on the pheromone values (τ) and heuristic information (η); pheromone values are associated with edges connecting adjacent vertices, while heuristic information is associated with vertices (candidate terms). In the first iteration of the ACO search, all vertices have the same pheromone values—they are all initialised to the same value (line 1); therefore, the heuristic information will have a higher influence at the early stages of the search. 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*). Once a rule is created, it undergoes a pruning procedure to remove irrelevant terms (vertices) from its antecedent. This is necessary due to the stochastic nature of the rule creation process—terms are added to the antecedent as long as there are available vertices for an ant to visit.

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 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-

```
Input: training examples
Output: best discovered rules
InitialisePheromones();
$rules_{gb} \leftarrow \{\}$;
$m \leftarrow 0$;
while $m$ < maximum iterations and not stagnation do
    $rules_{ib} \leftarrow \{\}$;
    for n ← 1 to colony_size do
        $rules_n \leftarrow \{\}$;
        for all class in classes do
            examples ← all training examples;
            while Count(examples, class) > maximum uncovered do
                ComputeHeuristicInformation(examples, class);
                rule ← CreateRule(examples, class);
                $Prune$(rule);
                examples ← examples − Covered(rule, class, examples);
                $rules_n \leftarrow rules_n +$ rule;
            end while
        end for
        if $Quality(rules_n) > Quality(rules_{ib})$ then
            $rules_{ib} \leftarrow rules_n$;
        end if
    end for
    $UpdatePheromones(rules_{ib})$;
    if $Quality(rules_{ib}) > Quality(rules_{gb})$ then
        $rules_{gb} \leftarrow rules_{ib}$;
    end if
    $m \leftarrow m + 1$;
end while
return $rules_{gb}$;
```

Figure 3: High-level pseudocode of the Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithm.

Miner$_{\mathrm{PB}}$ 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 discretisation of continuous values take advantage of the class information and use a more accurate class-specific measure—as it will be discussed in Subsections 3.2 and 3.3.

It is also important to emphasise that while the order of the rules is not important when the rules are used during classification, it is important during the ACO search. Similar to the cAnt-Miner$_{\mathrm{PB}}$, the edges of the construction graph in the unordered algorithm also have multiple values, one value for each rule position. Therefore, the current position of the rule (*tour* identification) is used during the rule creation and pheromone update processes. This allows the unordered algorithm to encode the order that the rules were created and update the pheromones accordingly, which in turn guides the search to more prominent sets of rules.

3.2 Class-Specific Heuristic Information

The rule creation in the Unordered cAnt-Miner$_{\mathrm{PB}}$ is a probabilistically process based on the pheromone values and heuristic. The probability of an ant to visit a vertex v_j when creating the rule t and located at vertex v_i is given by

$$P_{v_j} = \frac{\tau_{(t,v_i,v_j)} \cdot \eta_{v_j}}{\sum\limits_{k=1}^{\mathcal{F}_{v_i}} \tau_{(t,v_i,v_k)} \cdot \eta_{v_k}}, \tag{1}$$

where $\tau_{(t,v_i,v_j)}$ is the amount of pheromone associated with the entry (t, v_i, v_j)—the entry for *edge*$_{ij}$—in the pheromone matrix, η_{v_j} is the heuristic information associated with vertex v_j and $\mathcal{F}_{v_i}$ is the set of neighbouring vertices of vertex v_i. Note that the exponents α and β commonly used to control the influence of the pheromone and heuristic information, respectively, are set to 1 as in the original Ant-Miner algorithm and therefore omitted from Eq. 1.

Given that in the unordered algorithm the consequent of the rules is fixed before a rule is created, there is an opportunity to use a class-specific heuristic information in the rule creation process. In this case, the heuristic information of each vertex v_i of the construction graph for the class value c is given by

$$\eta_{v_i}(c) = \frac{|Examples(v_i, c)|}{|Examples(v_i)|}, \tag{2}$$

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 (satisfy the condition represented by 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), as illustrated in Figure 2. When a vertex 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$_{\mathrm{PB}}$ uses an entropy-based procedure, which does not require the class information a priori. Since in the unordered algorithm the class value is available to the discretisation procedure, we use the Laplace accuracy as a criterion to select a threshold value to discretise a continuous attribute as follows.

A threshold value w in the domain of the continuous attribute x dynamically generates two intervals: $x \leq w$ and $x > w$. The aim of the discretisation procedure is to find the best threshold value given the current set of training cases available. The best threshold value is the value w that maximises the interval accuracy in the set of examples S regarding the class value c, given by

$$max(\mathit{Lap}(c, S_{x \leq w}), \mathit{Lap}(c, S_{x>w})) \quad \forall\, w \in D_x \,, \tag{3}$$

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

$$\mathit{Lap}(c, E) = \frac{|E_c| + 1}{|E| + k} \,, \tag{4}$$

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 total 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 w, a term for the continuous attribute x is created based on the Laplace accuracy of the two intervals generated, given by

$$term_x = \begin{cases} x \leq t \text{ if } \mathit{Lap}(c, S_{x \leq w}) > \mathit{Lap}(c, S_{x>w}) \\ x > t \text{ if } \mathit{Lap}(c, S_{x \leq w}) < \mathit{Lap}(c, S_{x>w}) \end{cases} \,. \tag{5}$$

In order words, the dynamic discretisation procedure selects the term (attribute-condition) representing the interval that has the highest Laplace accuracy. There is no need to store the threshold value w as long as the same previous vertices are selected, since the dynamic discretisation procedure is deterministic and it will select the same threshold value given the same set of training examples (Otero et al., 2008, 2009).

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 (Clark and Boswell, 1991).

Let us consider an example in a 2-class problem $\{$`Y,N`$\}$ that is covered by rules `R1`$\Rightarrow$`Y [7,0]`, `R2`$\Rightarrow$`Y [4,0]` and `R3`$\Rightarrow$`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 four sets of experiments. In the first set of experiments, we evaluated different configurations of the proposed Unordered cAnt-Miner$_{\mathrm{PB}}$. 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 (Medland et al., 2012), in the performance of the algorithm. In the second set of experiments, we evaluated the Unordered cAnt-Miner$_{\mathrm{PB}}$ configurations 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 all algorithms used in the second set of experiments. Finally, the fourth set of experiments compare the Unordered cAnt-Miner$_{\mathrm{PB}}$ against a SVM classification algorithm in terms of predictive accuracy.

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 cAnt-Miner$_{\mathrm{PB}}$ combining both conflict resolution strategies with both the dynamic rule quality function selection and the error-based list quality function extensions proposed in (Medland et al., 2012): 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 datasets from the UCI Machine Learning repository (Lichman, 2013), 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 for each of the 8 aforementioned parameter configurations; we used the default parameters of cAnt-Miner$_{\mathrm{PB}}$ (Otero et al., 2013) for the remaining parameters: *colony size* of 5, *maximum number of iterations* of 500, *evaporation factor* of 0.9 (evaporation rate equal to $1 - \mathit{factor}$). Given the stochastic nature of the algorithm, each of the Unordered cAnt-Miner$_{\mathrm{PB}}$ 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 generalisation 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$_{\mathrm{PB}}$ 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$_{\mathrm{PB}}$, where an improvement in predictive accuracy is observed when the error-based list quality function is used (Medland et al., 2012). 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$_{\mathrm{PB}}$ 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.

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

The computational experiments comparing the proposed Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithm against state-of-the-art rule induction algorithms were carried out using a set of 32 publicly available datasets from the UCI Machine Learning repository (Lichman, 2013)—a summary of the datasets used in the experiments is presented in Table 1. In this table, the second and third columns give the number of nominal and continuous attributes, respectively; the fourth column gives the number of classes and the fifth column gives the number of examples of the dataset.

We have selected 6 rule induction algorithms, in addition to the cAnt-Miner$_{\mathrm{PB}}$ and Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithms.[2] 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$_{\mathrm{PB}}$ and Unordered cAnt-Miner$_{\mathrm{PB}}$ 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$_{\mathrm{PB}}$: one using the rule selection conflict resolution strategy (denoted as U-cAM$_{\mathrm{PB}}$ [S]) and one using the rule aggregation conflict resolution strategy (denoted as U-cAM$_{\mathrm{PB}}$ [A]). None of the algorithms had their parameter values optimised to individual datasets.

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

[2] The source-code and binaries of the new Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithm are available for download at `http://www.cs.kent.ac.uk/people/staff/febo`. We used the cAnt-Miner$_{\mathrm{PB}}$ algorithm (from the same software package) with the error-based list quality function, as suggested by Medland et al. (2012).

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

dataset	# attributes		# classes	# examples
	nominal	continuous		
annealing	29	9	6	898
balance-scale	4	0	3	625
banknote	0	4	2	1372
breast-l	9	0	2	286
breast-p	0	32	2	198
breast-tissue	0	9	6	106
breast-w	0	30	2	569
cardiotocography	0	21	3	2126
climate	0	18	2	540
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
indian-liver	1	9	2	579
ionosphere	0	34	2	351
iris	0	4	3	150
liver-disorders	0	6	2	345
lymphography	15	3	4	148
mammographic	4	1	2	944
parkinsons	0	22	2	195
pima	0	8	2	768
thoracic	13	3	2	470
thyroid	0	5	2	215
tic-tae-toe	9	0	2	958
vertebral-column-2c	0	6	2	310
vertebral-column-3c	0	6	3	310
waveform	0	21	3	5000
wine	0	13	3	178
yeast	0	8	10	1484

average value obtained by an algorithm at the end of the tenfold cross-validation procedure. In the case of stochastic algorithms—cAnt-Miner$_{\mathrm{PB}}$, Unordered cAnt-Miner$_{\mathrm{PB}}$, PSO/ACO2 and BioHEL—the average value is computed over 15 executions of the tenfold cross-validation procedure (i.e., each algorithm is run 15×10 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). In this table, the value of the algorithm with the best accuracy is indicated by $\triangle$ symbol.

Table 4 presents the statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test (Demšar, 2006; García and Herrera, 2008): the first column corresponds to the algorithm's name, the second column corresponds 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

Table 2: The 6 rule induction algorithms used in the second set of experiments, in addition to the cAnt-Miner$_{\mathrm{PB}}$ and Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithms.

name	reference	description
Unordered CN2	(Clark and Boswell, 1991)	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	(Quinlan, 1993)	A rule induction algorithm that extracts a set of classification rules from an unpruned decision tree created by the well-known C4.5 algorithm (Quinlan, 1993, 1996)
PART	(Frank and Witten, 1998; Witten and Frank, 2011)	A rule induction algorithm that combines a sequential covering strategy with a decision tree induction procedure to create a rule
JRip	(Witten and Frank, 2011)	Weka's implementation of the RIPPER (Cohen, 1995) 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
PSO/ACO2	(Holden and Freitas, 2008)	A hybrid particle swarm optimisation/ant colony optimisation (PSO/ACO) algorithm for the discovery of classification rules. The PSO/ACO2 algorithm follows a sequential covering strategy and directly deals with both continuous and nominal attribute values
BioHEL	(Bacardit et al., 2009)	A genetic algorithm (GA) that discovers a list of classification rules using a sequential covering strategy. Each rule is created using a GA search

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 italic-boldface when there is a statistically significant difference at the 0.05 (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$_{\mathrm{PB}}$ using the rule selection conflict resolution strategy (denoted as U-cAM$_{\mathrm{PB}}$ [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—and also outperformed both PSO/ACO2 and BioHEL algorithms with statistically significant differences. The predictive accuracy results did not show statistically significant differences between the rule selection and rule aggregation conflict resolution strategies, although there is a clear difference in terms of the interpretability of the discovered rules (as discussed in Subsection 4.3). While there are no statistically significant differences between cAnt-Miner$_{\mathrm{PB}}$ and Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithms, the results obtained by the proposed Unordered cAnt-Miner$_{\mathrm{PB}}$ are positive, overall: the discovery of unordered rules explicitly improves their interpretability (i.e., a particular rule has a modular meaning independent

[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

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

dataset	U-cAM$_{\mathrm{PB}}$ [S]	U-cAM$_{\mathrm{PB}}$ [A]	Unordered CN2	C4.5rules	PART	JRip	cAnt-Miner$_{\mathrm{PB}}$	PSO/ACO2	BioHEL
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]	97.25 [0.08]	99.76 [0.00] △
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]	77.37 [0.03]	69.47 [0.08]
banknote	98.80 [0.00] △	98.64 [0.01]	96.07 [0.84]	98.69 [0.22]	98.32 [0.39]	98.03 [0.51]	98.55 [0.00]	98.39 [0.09]	97.35 [0.01]
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] △	72.06 [0.07]	57.01 [0.14]
breast-p	75.19 [0.44]	76.07 [0.32] △	73.68 [1.56]	74.68 [1.99]	72.74 [3.95]	75.79 [0.91]	69.47 [0.32]	68.09 [0.45]	57.47 [1.47]
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]	63.07 [0.85]	51.11 [0.31]
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]	94.61 [0.04]	92.10 [0.06]
cardiotocography	92.46 [0.01]	91.53 [0.01]	88.99 [0.59]	92.85 [0.53]	92.85 [0.52]	92.66 [0.62]	92.71 [0.01]	93.84 [0.09] △	91.36 [0.02]
climate	93.11 [0.26] △	92.91 [0.02]	91.29 [0.48]	92.96 [1.13]	92.96 [0.82]	91.11 [0.91]	93.07 [0.08]	92.85 [0.28]	83.80 [0.08]
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]	83.83 [0.04]	77.33 [0.14]
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]	70.35 [0.09]	64.38 [0.12]
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]	69.68 [0.53]	72.02 [0.10]
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]	92.48 [0.02]	92.24 [0.14]
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] △	70.57 [0.28]	59.23 [0.59]
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]	53.23 [0.23]	45.39 [0.38]
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]	63.18 [0.45]	82.06 [0.41] △
indian-liver	69.26 [0.27]	70.52 [0.06]	71.67 [0.78] △	68.23 [1.53]	70.64 [1.70]	67.88 [1.57]	68.29 [0.16]	68.23 [0.36]	58.04 [0.16]
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]	86.06 [0.20]	87.16 [0.21]
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]	95.38 [0.01]	95.20 [0.09]
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]	68.13 [0.33]	58.48 [0.26]
lymphography	79.96 [0.47]	80.82 [0.68] △	78.91 [4.08]	79.14 [2.66]	71.67 [3.92]	76.39 [3.83]	76.39 [0.62]	54.89 [0.06]	75.12 [0.44]
mammographic	83.12 [0.02]	83.26 [0.01]	77.76 [1.02]	83.58 [0.85] △	81.77 [1.08]	82.73 [1.24]	83.11 [0.01]	82.60 [0.12]	76.04 [0.04]
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]	87.36 [0.17]	80.93 [0.79]
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] △	73.09 [0.05]	65.62 [0.12]
thoracic	82.87 [0.02]	84.83 [0.02]	85.10 [0.31] △	81.70 [1.39]	78.08 [1.10]	84.68 [0.43]	82.89 [0.04]	80.70 [0.30]	71.40 [0.43]
thyroid	92.83 [0.30]	92.52 [0.28]	94.42 [1.83] △	92.08 [2.12]	93.01 [2.25]	92.53 [1.62]	91.57 [0.02]	91.76 [0.03]	92.12 [0.10]
tic-tae-toe	91.60 [0.39]	79.97 [0.35]	98.74 [0.51]	98.75 [0.64]	94.78 [0.69]	97.60 [0.60]	81.36 [0.21]	100.00 [0.00] △	100.00 [0.00] △
vertebral-column-2c	81.40 [0.44]	82.00 [0.35]	80.00 [2.14]	79.35 [1.93]	80.32 [0.75]	82.58 [2.32] △	79.61 [0.10]	81.78 [0.07]	68.81 [0.16]
vertebral-column-3c	79.40 [0.41]	80.47 [0.31]	76.11 [2.11]	79.35 [1.61]	80.00 [1.43]	80.32 [2.33]	81.33 [0.20]	82.90 [0.08] △	67.97 [0.18]
waveform	80.19 [0.02]	80.64 [0.01]	69.50 [0.75]	77.80 [0.45]	78.48 [0.71]	79.82 [0.50]	75.47 [0.02]	81.73 [0.10] △	76.05 [0.02]
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]	89.17 [0.16]	85.51 [0.30]
yeast	57.45 [0.03]	57.39 [0.03]	49.72 [1.12]	58.22 [0.85] △	53.43 [0.88]	56.87 [1.15]	56.08 [0.08]	52.98 [0.27]	55.04 [0.07]

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 $\alpha = 0.05$ significance level are shown in italic-boldface.

Algorithm	Avg. Rank	p-value	Hommel
U-cAM$_{\text{PB}}$ [S] (control)	3.37	–	–
U-cAM$_{\text{PB}}$ [A]	3.56	0.7841	0.05
cAnt-Miner$_{\text{PB}}$	4.15	0.2538	0.025
C4.5rules	***5.01***	***0.0165***	***0.0166***
PSO/ACO2	***5.25***	***0.0061***	***0.0125***
PART	***5.25***	***0.0061***	***0.01***
JRip	***5.40***	***0.0030***	***0.0083***
Unordered CN2	***5.53***	***0.0016***	***0.0071***
BioHEL	***7.45***	***2.5E-9***	***0.0062***

of the others rules), without sacrificing the predictive accuracy. More interestingly, the Unordered cAnt-Miner$_{\text{PB}}$ significantly outperformed the Unordered CN2 algorithm, which is the only other algorithm that discovers unordered rules. Both algorithms share a similar strategy to create the rules, consisting of creating rules for each class value separately—i.e., the consequent of the rule is fixed during the creation process. Therefore, the difference in performance can be attributed mainly to the ACO search strategy, which allows the Unordered cAnt-Miner$_{\text{PB}}$ to effectively explore the search space for the best set of rules, instead of using a greedy strategy as the Unordered CN2.

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 (Freitas, 2013). 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 (Lavrac, 1999; Lavesson, 2011). An empirical study tested the assumption that smaller models are more comprehensible to users (Huysmans et al., 2011). 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 (Schwabacher and Langley, 2001) it is reported that users found that a model with 41 rules was considered too large to be analysed by a user, while in (Tsumoto, 2000) 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—may not be 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

Table 5: Average prediction-explanation size, measured by tenfold cross-validation. The value of the best algorithm (lowest value) for a given datasets is marked with a △ symbol.

dataset	U-cAM$_{\mathrm{PB}}$ [S]	U-cAM$_{\mathrm{PB}}$ [A]	Unordered CN2	C4.5rules	PART	JRip	cAnt-Miner$_{\mathrm{PB}}$	PSO/ACO2	BioHEL
annealing	2.14 [0.02] △	5.64 [0.08]	4.49 [0.21]	19.92 [0.99]	39.53 [3.02]	21.97 [1.55]	7.39 [0.14]	26.97 [0.36]	8.35 [0.05]
balance-scale	1.25 [0.01] △	4.86 [0.02]	3.58 [0.09]	42.78 [0.84]	24.18 [1.27]	10.99 [0.62]	4.40 [0.01]	23.66 [0.33]	98.19 [0.46]
banknote	1.75 [0.01] △	5.71 [0.04]	4.09 [0.08]	11.94 [0.44]	6.50 [0.20]	11.45 [0.44]	4.56 [0.04]	6.48 [0.03]	24.91 [0.30]
breast-l	2.64 [0.04] △	8.36 [0.14]	2.88 [0.14]	13.32 [1.66]	16.98 [2.42]	3.42 [0.50]	4.00 [0.12]	13.15 [0.29]	75.13 [0.57]
breast-p	1.48 [0.03]	4.77 [0.05]	3.40 [0.15]	4.31 [0.56]	5.40 [0.85]	0.80 [0.44] △	4.91 [0.07]	9.04 [0.49]	35.85 [0.42]
breast-tissue	1.42 [0.01] △	4.43 [0.06]	2.10 [0.25]	12.50 [0.82]	9.67 [0.72]	6.35 [0.46]	3.92 [0.03]	6.85 [0.13]	19.71 [0.40]
breast-w	1.30 [0.01] △	7.00 [0.09]	5.51 [0.31]	9.15 [0.39]	4.90 [0.26]	6.67 [0.54]	3.13 [0.09]	5.35 [0.07]	22.89 [0.23]
cardiotocography	2.71 [0.03] △	8.86 [0.17]	5.44 [0.37]	74.82 [4.56]	29.80 [1.14]	34.01 [2.51]	5.66 [0.06]	21.95 [0.21]	122.83 [0.46]
climate	1.75 [0.04] △	4.73 [0.05]	5.77 [0.22]	14.88 [0.94]	2.58 [0.15]	7.31 [0.55]	2.93 [0.07]	3.40 [0.12]	36.71 [0.43]
credit-a	2.29 [0.03] △	8.52 [0.15]	5.46 [0.21]	17.39 [0.79]	25.75 [4.62]	5.72 [1.13]	3.34 [0.11]	24.14 [0.37]	105.55 [0.53]
credit-g	2.72 [0.03] △	13.25 [0.13]	4.69 [0.16]	41.82 [3.61]	73.21 [3.40]	8.25 [1.18]	13.70 [0.35]	133.15 [2.19]	235.32 [1.03]
cylinder-bands	3.21 [0.05] △	12.70 [0.14]	3.40 [0.21]	54.82 [1.17]	43.28 [2.52]	13.88 [1.78]	25.99 [0.74]	47.62 [0.57]	99.46 [0.79]
dermatology	3.66 [0.06]	15.18 [0.20]	2.76 [0.14] △	22.77 [0.76]	13.87 [0.72]	17.89 [1.00]	14.91 [0.24]	13.97 [0.09]	22.48 [0.24]
glass	2.21 [0.02] △	6.01 [0.08]	3.46 [0.30]	29.84 [1.10]	21.21 [1.83]	12.78 [0.77]	5.26 [0.08]	26.53 [0.33]	44.14 [0.41]
heart-c	2.76 [0.02] △	12.04 [0.11]	3.80 [0.35]	43.66 [1.78]	52.76 [2.04]	6.34 [1.28]	8.01 [0.23]	29.05 [0.44]	107.23 [0.47]
heart-h	2.62 [0.05] △	11.21 [0.11]	4.64 [0.17]	45.48 [0.78]	24.77 [1.77]	4.29 [0.69]	5.71 [0.20]	16.54 [0.19]	51.99 [0.47]
indian-liver	2.91 [0.03] △	8.49 [0.07]	3.37 [0.22]	19.63 [4.01]	14.24 [1.81]	3.94 [0.61]	5.05 [0.09]	45.10 [0.71]	115.35 [1.05]
ionosphere	2.57 [0.04] △	6.35 [0.09]	4.69 [0.30]	13.12 [0.42]	8.99 [0.46]	4.89 [0.68]	5.11 [0.11]	8.05 [0.29]	18.54 [0.16]
iris	1.37 [0.01] △	2.93 [0.06]	2.03 [0.13]	3.47 [0.05]	2.53 [0.10]	2.46 [0.14]	2.89 [0.05]	1.67 [0.00]	6.71 [0.07]
livers-disorder	2.20 [0.02] △	7.01 [0.08]	3.55 [0.26]	26.34 [1.49]	13.47 [1.29]	5.81 [0.71]	5.24 [0.07]	33.27 [0.20]	62.40 [0.50]
lymphography	2.57 [0.08] △	9.13 [0.10]	4.74 [0.29]	13.32 [0.52]	10.80 [0.73]	7.72 [0.48]	6.81 [0.51]	2.08 [0.03]	28.96 [0.29]
mammographic	1.77 [0.03] △	8.16 [0.09]	3.88 [0.11]	6.59 [0.17]	8.41 [2.14]	2.47 [0.17]	3.42 [0.07]	7.22 [0.27]	121.34 [0.64]
parkinsons	1.45 [0.03] △	4.73 [0.05]	3.89 [0.27]	11.79 [0.47]	5.45 [0.48]	6.65 [0.65]	3.00 [0.06]	4.69 [0.05]	18.33 [0.26]
pima	2.06 [0.01] △	8.54 [0.12]	4.20 [0.18]	14.94 [0.76]	9.06 [0.62]	6.01 [0.72]	4.95 [0.06]	45.65 [0.36]	120.05 [0.67]
thoracic	4.28 [0.08]	11.19 [0.56]	4.56 [0.24]	25.56 [1.51]	31.30 [2.43]	0.40 [0.31] △	4.59 [0.30]	31.07 [1.77]	96.43 [0.46]
thyroid	2.17 [0.05] △	4.17 [0.09]	2.35 [0.21]	8.62 [0.16]	5.23 [0.15]	4.74 [0.35]	3.06 [0.06]	5.32 [0.03]	11.22 [0.09]
tic-tac-toe	2.37 [0.02] △	5.24 [0.09]	3.65 [0.11]	33.34 [0.87]	48.96 [1.91]	26.91 [2.01]	8.86 [0.58]	17.59 [0.00]	71.48 [0.16]
vertebral-column-2c	1.49 [0.01] △	5.48 [0.08]	2.52 [0.18]	7.25 [0.67]	3.12 [0.22]	5.78 [0.80]	3.12 [0.06]	8.35 [0.10]	38.16 [0.31]
vertebral-column-3c	1.62 [0.01] △	5.72 [0.05]	3.68 [0.19]	8.68 [0.52]	5.37 [0.26]	8.99 [1.04]	3.72 [0.06]	8.86 [0.10]	39.28 [0.23]
waveform	2.95 [0.01] △	9.00 [0.09]	6.16 [0.14]	229.06 [5.69]	151.41 [3.23]	82.35 [5.01]	5.58 [0.06]	279.83 [1.39]	357.86 [1.27]
wine	1.12 [0.01] △	3.50 [0.06]	3.46 [0.17]	5.24 [0.11]	3.62 [0.13]	4.51 [0.31]	2.34 [0.04]	5.28 [0.09]	11.49 [0.13]
yeast	3.69 [0.03] △	14.18 [0.09]	5.47 [0.08]	113.97 [4.43]	321.17 [17.77]	34.44 [2.23]	11.31 [0.26]	357.65 [2.62]	245.87 [1.26]

Table 6: Statistical test results of the algorithms' average prediction-explanation size according to the non-parametric Friedman test with the Hommel's post-hoc test. Statistically significant differences at the $\alpha = 0.05$ significance level are shown in italic-boldface.

Algorithm	Avg. Rank	p-value	Hommel
U-cAM$_{\mathrm{PB}}$ [S] (control)	1.12	–	–
Unordered CN2	***2.53***	***0.0399***	***0.05***
cAnt-Miner$_{\mathrm{PB}}$	***3.50***	***5.2E-4***	***0.025***
JRip	***4.68***	***1.9E-7***	***0.0166***
U-cAM$_{\mathrm{PB}}$ [A]	***4.75***	***1.1E-7***	***0.0125***
PSO/ACO2	***6.21***	***1.0E-13***	***0.01***
PART	***6.25***	***7.1E-14***	***0.0083***
C4.5rules	***7.15***	***1.2E-18***	***0.0071***
BioHEL	***8.78***	***4.9E-29***	***0.0062***

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 cAnt-Miner$_{\mathrm{PB}}$, PART, C4.5rules, JRip, PSO/ACO2 and BioHEL algorithms, which produce 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 indirectly involved, 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 cAnt-Miner$_{\mathrm{PB}}$ algorithm, which produces a set of rules (unordered 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. The Unordered CN2 uses a similar rule aggregation strategy to resolve conflicts, and therefore the same definition for the prediction-explanation size. It should be noted that in the Unordered cAnt-Miner$_{\mathrm{PB}}$ 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 presents the results (*average* [*standard error*]) concerning the prediction-explanation size of the models discovered by all the algorithms used in the second set of experiments (Subsection 4.2), the lower the value the better the algorithm performance. In this table, the value of the algorithm with the best prediction-explanation size is indicated by $\triangle$ symbol. Table 6 presents the statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test—where the lower the rank the better the algorithm's performance. A row is shown in italic-boldface when

```
     IF     suprapapillary-epidermis = 3
     THEN   class = 1

ELSE IF     focal-hypergranulosis = 2
     THEN   class = 3

ELSE IF     oral-mucosal = 2
     THEN   class = 3

ELSE IF     suprapapillary-epidermis = 2
     THEN   class = 1

ELSE IF     rete-ridges = 2
        AND spongiosis = 0
     THEN   class = 1

ELSE IF     papillary-dermis = 3
     THEN   class = 5

ELSE IF     scaling = 3
        AND knee-elbow = 0
        AND koebner-phenomenon = 0
     THEN   class = 2
```

(a) Ordered rules used by cAnt-Miner$_{\mathrm{PB}}$.

```
     IF     eosinophils-infiltrate = 1
        AND polygonal-papules = 0
        AND disappearance-granular-layer = 0
        AND suprapapillary-epidermis = 0
        AND koebner-phenomenon = 0
     THEN   class = 2 [0,14,0,0,4,0]

     IF     scaling = 3
        AND polygonal-papules = 0
        AND hyperkeratosis = 0
        AND koebner-phenomenon = 0
        AND family-history = 0
        AND munro-microabcess = 0
        AND disappearance-granular-layer = 0
     THEN   class = 2 [3,10,0,0,0,0]
```

(b) Rules used by Unordered cAnt-Miner$_{\mathrm{PB}}$ [A] (the class frequencies are shown between brackets).

```
     IF     spongiosis = 3
        AND perifollicular-parakeratosis = 0
        AND vacuolisation-basal-layer = 0
     THEN   class = 2
```

(c) Rule used by Unordered cAnt-Miner$_{\mathrm{PB}}$ [S] (the one with the highest quality).

Figure 4: Rules used to classify a particular test example from the dermatology dataset: (a) in cAnt-Miner$_{\mathrm{PB}}$, rules need to be evaluated following their order—rule 7 (bottom rule from the second column) is used only if all previous rules do not classify the example (total of 10 attribute-conditions); (b) Unordered cAnt-Miner$_{\mathrm{PB}}$ [A] can make the same prediction using 2 rules (total of 12 attribute-conditions); (c) Unordered cAnt-Miner$_{\mathrm{PB}}$ [S] uses only 1 rule (total of 3 attribute-conditions) to make the same prediction.

there is a statistically significant difference at the 0.05 (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.

On one hand, the advantage of unordered rules combined with the rule selection strategy (denoted as U-cAM$_{\mathrm{PB}}$ [S]) is clear: in 29 out of the 32 datasets, it has the lowest number of attribute-conditions involved in the classification of an example. It outperformed all other algorithms with statistically significant differences. On the other hand, the results using the rule aggregation strategy (denoted as U-cAM$_{\mathrm{PB}}$ [A]) were somewhat unexpected: U-cAM$_{\mathrm{PB}}$ [A] was statistically significantly worse that the U-cAM$_{\mathrm{PB}}$ [S], and even achieved a worse average rank than the cAnt-Miner$_{\mathrm{PB}}$. This is due to the fact that in the set of rules discovered by U-cAM$_{\mathrm{PB}}$ [A], rules are longer (with more attribute-conditions) on average and they may overlap when classifying an example. Since all rules covering an example will be taken into consideration in the prediction-explanation size calculation, it leads to an increased number of attribute-conditions. This is illustrated in Figures 4 and 5. Figure 4 shows the rules involved to classify a particular test example of the *dermatology* dataset: cAnt-Miner$_{\mathrm{PB}}$ uses a total of 10

```
     IF     ink_type = cover
        AND blade_mfg = benton
        AND grain_screened = yes
     THEN   class = band

ELSE IF     paper_type = super
     THEN   band

ELSE IF     press_type = motter70
        AND plating_tank = 1910
        AND press_speed > 1565.0
        AND type_cylinder = yes
     THEN   class = noband (1,12)

ELSE IF     press_speed > 2184.5
     THEN   class = noband (10,75)

ELSE IF     humifity <= 69.5
        AND ink_type = coated
        AND grain_screened = no
     THEN   class = noband
```

```
ELSE IF     press_speed <= 1325.0
     THEN   class = band (16,5)

ELSE IF     press_speed <= 1495.0
        AND press = 802
        AND press_type = albert70
     THEN   class = noband

ELSE IF     press_speed > 2025.0
        AND viscosity > 52.5
        AND paper_type = coated
     THEN   class = noband

ELSE IF     ink_temperature > 16.9
     THEN   class = band

ELSE IF     viscosity > 62.5
        AND solvent_type = line
        AND proof_ctd_ink = yes
     THEN   class = band
```

(a) Ordered rules used by cAnt-Miner$_{\text{PB}}$.

```
     IF     press = 815
        AND current_density > 36.0
     THEN   class = band [51,15]

     IF     humifity > 89.5
        AND varnish_pct <= 0.25
     THEN   class = band [13,9]

     IF     type_cylinder = no
        AND solvent_type = line
        AND ink_type = uncoated
        AND hardener > 0.95
     THEN   class = band [23,18]
```

```
     IF     viscosity > 64.5
     THEN   class = band [16,12]

     IF     type_cylinder = no
        AND ink_type = uncoated
        AND paper_type = uncoated
        AND wax <= 2.55
     THEN   class = band [20,10]

     IF     viscosity > 62.5
        AND plating_tank = 1910
        AND proof_ctd_ink = yes
     THEN   class = band [16,9]
```

(b) Rules used by Unordered cAnt-Miner$_{\text{PB}}$ [A] (the class frequencies are shown between brackets).

```
     IF     press_type = woodhoe70
        AND press = 815
        AND wax <= 2.65
        AND roller_durometer > 34.0
        AND solvent_type = line
     THEN   class = band
```

(c) Rule used by Unordered cAnt-Miner$_{\text{PB}}$ [S] (the one with the highest quality).

Figure 5: Rules used to classify a particular test example from the cylinder-bands dataset: (a) in cAnt-Miner$_{\text{PB}}$, rules need to be evaluated following their order—rule 10 (bottom rule from the second column) is used only if all previous rules do not classify the example (total of 23 attribute-conditions); (b) Unordered cAnt-Miner$_{\text{PB}}$ [A] can make the same prediction using 6 rules (total of 16 attribute-conditions); (c) Unordered cAnt-Miner$_{\text{PB}}$ [S] uses only 1 rule (total of 5 attribute-conditions) to make the same prediction.

attribute-conditions over 7 rules; U-cAM$_{\text{PB}}$ [A] makes the same (correct) prediction using a total of 12 attribute conditions over 2 rules—the numbers enclosed in brackets represent the class frequencies of the rule used to decide the final class prediction by the subset of rules that cover the example; U-cAM$_{\text{PB}}$ [S] uses only 3 attribute-conditions over 1 rule to make the same prediction, emphasising the advantage of unordered rules in combination with the rule selection strategy. A similar behaviour is observed in the *cylinder-bands* dataset shown in Figure 5: cAnt-Miner$_{\text{PB}}$ uses a total of 23 attribute-conditions over 10 rules; U-cAM$_{\text{PB}}$ [A] makes the same (correct) prediction using a total of 16 attribute-conditions over 6 rules; U-cAM$_{\text{PB}}$ [S] uses only 5 attribute-conditions over 1 rule to make the same prediction. The behaviour illustrated in these figures is reflected in the average values presented in Table 5, where U-cAM$_{\text{PB}}$ [S] consistently uses a smaller number of attribute-conditions to make the predictions.

Overall, these results are positive: we were able to improve the interpretability of the rules (measured as the *prediction-explanation size*) by discovering unordered rules, with no negative impact on the predictive accuracy. The Unordered cAnt-Miner$_{\text{PB}}$ with the rule selection strategy was the algorithm that achieved the best rank in terms of both predictive accuracy and prediction-explanation size measure, significantly outperforming state-of-the-art rule induction algorithms.

4.4 Comparisons with SVM (Support Vector Machines)

We also compared the predictive accuracy of Unordered cAnt-Miner$_{\text{PB}}$ against a SVM classifier (Vapnik, 2000) using a RBF (radial basis function) kernel. A SVM with a RBF kernel function produces a *black-box* model, which is not straightforward human-interpretable, by applying non-linear transformations to the input data in order to find optimal class separation—the input is projected into a high-dimensional space created by the kernel function. SVMs are generally considered the state-of-the-art in classification regarding predictive accuracy, specially useful in applications where the comprehensibility of the model is not important. While in this paper we address the comprehensibility of classification rules, it is informative to compare the predictive accuracy of Unordered cAnt-Miner$_{\text{PB}}$ against a SVM classifier nevertheless.

The experiments with the SVM classifier were carried out using the same datasets from Table 1. For each dataset, the parameters of the SVM were determined using a grid search procedure.[3] The results (*average* [*standard error*]) are presented in Table 7, using the same tenfold cross-validation procedure—the results of the 2 configurations for the Unordered cAnt-Miner$_{\text{PB}}$ (selection [S] and aggregation [A] conflict resolution strategies) are shown again for convenience. Table 8 presents the statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test—same test applied in the experiments described in Subsections 4.2 and 4.3—comparing the SVM results against Unordered cAnt-Miner$_{\text{PB}}$ results. As can be seen in Table 8, there are no statistically significant differences between the performances of the algorithms. At the same time, Unordered cAnt-Miner$_{\text{PB}}$ has the advantage of producing a comprehensible (*white-box*) classification model in the form of classification rules, while SVM produces a *black-box* model.

[3] We used the SVM implementation from `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. The parameters were determined using the `easy.py` grid search script.

Table 7: Average predictive accuracy (*average* [*standard error*]) in %, measured by ten-fold cross-validation. The value of the most accurate algorithm for a given datasets is marked with a △ symbol.

dataset	U-cAM$_{\mathrm{PB}}$ [S]	U-cAM$_{\mathrm{PB}}$ [A]	SVM
annealing	97.70 [0.13] △	95.53 [0.23]	97.66 [0.51]
balance-scale	77.82 [0.20]	88.85 [0.21]	100.00 [0.00] △
banknote	98.80 [0.00]	98.64 [0.01]	100.00 [0.00] △
breast-l	74.02 [0.19] △	73.59 [0.35]	72.44 [3.01]
breast-p	75.19 [0.44]	76.07 [0.32] △	75.79 [0.91]
breast-tissue	65.91 [0.67]	67.58 [0.72] △	64.36 [3.63]
breast-w	95.20 [0.12]	95.02 [0.19]	98.06 [0.56] △
cardiotocography	92.46 [0.01]	91.53 [0.01]	93.27 [0.63] △
climate	93.11 [0.26] △	92.91 [0.02]	95.37 [0.84] △
credit-a	84.93 [0.28]	85.68 [0.16] △	84.20 [1.41]
credit-g	72.84 [0.32]	71.08 [0.22]	77.90 [1.04] △
cylinder-bands	72.06 [0.42]	71.83 [0.41]	79.44 [1.60] △
dermatology	90.54 [0.32]	90.08 [0.44]	97.54 [0.64] △
glass	69.46 [0.81]	68.24 [0.81]	71.08 [4.37] △
heart-c	56.66 [0.50]	57.13 [0.52] △	55.44 [1.65]
heart-h	64.64 [0.45]	64.02 [0.33]	68.76 [1.83] △
indian-liver	69.26 [0.27]	70.52 [0.06] △	69.78 [1.03]
ionosphere	89.44 [0.33]	89.85 [0.35]	92.89 [1.05] △
iris	94.33 [0.20]	93.87 [0.19]	95.33 [1.42] △
liver-disorders	69.78 [0.64]	69.49 [0.46]	73.33 [0.95] △
lymphography	79.96 [0.47]	80.82 [0.68]	83.19 [2.82] △
mammographic	83.12 [0.02]	83.26 [0.01]	83.57 [1.14] △
parkinsons	87.75 [0.45]	85.28 [0.62]	93.82 [1.88] △
pima	74.55 [0.30]	74.32 [0.15]	77.08 [1.45] △
thoracic	82.87 [0.02]	84.83 [0.02]	85.11 [0.00] △
thyroid	92.83 [0.30]	92.52 [0.28]	96.69 [1.88] △
tic-tae-toe	91.60 [0.39]	79.97 [0.35]	99.48 [0.32] △
vertebral-column-2c	81.40 [0.44]	82.00 [0.35]	83.87 [1.73] △
vertebral-column-3c	79.40 [0.41]	80.47 [0.31]	85.81 [1.61] △
waveform	80.19 [0.02]	80.64 [0.01]	86.70 [0.45] △
wine	95.46 [0.30]	95.48 [0.32]	97.16 [0.95] △
yeast	57.45 [0.03]	57.39 [0.03]	59.70 [0.96] △

Table 8: Statistical test results of the algorithms' average predictive accuracy according to the non-parametric Friedman test with the Hommel's post-hoc test—no statistically significant differences between SVM and Unordered cAnt-Miner$_{\mathrm{PB}}$ (U-cAM$_{\mathrm{PB}}$) are observed at the $\alpha = 0.05$ significance level.

Algorithm	Avg. Rank	p-value	Hommel
SVM (control)	1.81	–	–
U-cAM$_{\mathrm{PB}}$ [S]	2.06	0.3173	0.05
U-cAM$_{\mathrm{PB}}$ [A]	2.12	0.2112	0.025

Table 9: Average computational time (*average* [*standard error*]) in seconds to complete a fold of the cross-validation (single execution) during training. The deterministic Unordered CN2, C4.5rules, PART and JRip algorithms take on average a second to complete a fold in any of the datasets used in the experiments.

dataset	U-cAM$_{\mathrm{PB}}$ [S]	U-cAM$_{\mathrm{PB}}$ [A]	cAnt-Miner$_{\mathrm{PB}}$	PSO/ACO2	BioHEL	SVM
annealing	414.98 [27.07]	419.31 [91.74]	21.70 [1.49]	15.96 [0.20]	5.85 [0.11]	16.60 [0.16]
balance-scale	20.25 [0.27]	21.02 [0.11]	1.49 [0.01]	9.61 [0.04]	30.90 [0.04]	6.80 [0.13]
banknote	46.03 [4.73]	76.91 [0.13]	15.08 [0.19]	9.50 [0.10]	14.54 [0.06]	10.10 [0.10]
breast-l	13.03 [0.05]	12.95 [0.08]	1.21 [0.01]	2.76 [0.02]	12.35 [0.03]	4.20 [0.13]
breast-p	43.93 [1.50]	40.53 [1.24]	7.33 [0.34]	2.73 [0.02]	4.59 [0.01]	2.40 [0.16]
breast-tissue	12.61 [0.21]	12.05 [0.35]	1.81 [0.01]	0.90 [0.01]	3.80 [0.01]	1.30 [0.15]
breast-w	91.75 [10.49]	98.12 [11.87]	8.98 [0.32]	7.00 [0.07]	7.91 [0.01]	5.50 [0.17]
cardiotocography	657.10 [0.67]	666.50 [3.91]	181.29 [0.93]	63.71 [0.39]	75.93 [0.35]	60.80 [0.33]
climate	106.11 [0.46]	112.61 [0.14]	7.89 [0.04]	3.38 [0.02]	4.04 [0.06]	5.10 [0.10]
credit-a	127.32 [4.49]	118.05 [6.66]	7.03 [0.19]	13.26 [0.49]	34.31 [0.03]	11.30 [0.26]
credit-g	590.59 [13.13]	588.54 [16.35]	49.98 [2.43]	41.34 [0.97]	61.44 [0.05]	38.10 [0.23]
cylinder-bands	428.62 [13.01]	404.43 [17.37]	64.74 [5.84]	16.75 [1.02]	24.96 [0.02]	11.20 [0.13]
dermatology	100.16 [4.87]	105.90 [3.68]	9.83 [0.20]	5.04 [0.03]	9.36 [0.01]	5.30 [0.15]
glass	105.60 [8.15]	104.78 [6.80]	4.65 [0.04]	3.62 [0.02]	7.03 [0.01]	2.20 [0.13]
heart-c	270.46 [8.94]	277.62 [16.77]	6.42 [0.07]	5.08 [0.04]	17.30 [0.01]	4.20 [0.13]
heart-h	163.80 [3.73]	168.25 [5.76]	4.48 [0.03]	4.91 [0.13]	8.40 [0.02]	3.70 [0.15]
indian-liver	62.85 [0.01]	61.67 [0.01]	20.77 [0.01]	15.36 [0.55]	29.80 [0.09]	18.20 [0.49]
ionosphere	60.07 [3.13]	59.27 [4.09]	11.21 [0.90]	3.36 [0.04]	5.22 [0.01]	3.70 [0.15]
iris	4.35 [0.01]	4.15 [0.01]	1.21 [0.00]	0.50 [0.00]	1.83 [0.01]	1.00 [0.00]
livers-disorder	38.29 [0.55]	38.865 [0.67]	5.43 [0.03]	5.32 [0.04]	11.85 [0.01]	8.10 [0.10]
lymphography	10.20 [0.18]	10.112 [0.11]	1.69 [0.01]	1.18 [0.01]	5.96 [0.01]	1.60 [0.16]
mammographic	49.91 [1.07]	56.63 [0.57]	4.36 [0.11]	17.36 [0.28]	41.15 [0.15]	174.10 [9.98]
parkinsons	18.63 [0.24]	18.31 [0.37]	2.70 [0.02]	1.50 [0.01]	3.44 [0.01]	1.70 [0.15]
pima	152.13 [6.06]	144.69 [4.57]	10.08 [0.16]	19.74 [0.09]	29.26 [0.03]	26.10 [0.23]
thoracic	58.01 [0.37]	49.06 [0.68]	6.08 [0.14]	7.24 [0.15]	20.69 [0.03]	10.30 [0.21]
thyroid	6.86 [0.01]	7.07 [0.01]	1.66 [0.01]	0.81 [0.01]	2.22 [0.01]	1.20 [0.13]
tic-tac-toe	51.69 [1.66]	47.52 [2.02]	3.84 [0.01]	9.48 [0.03]	30.13 [0.03]	52.00 [0.87]
vertebral-column-2c	18.58 [0.02]	18.41 [0.02]	2.34 [0.01]	2.47 [0.01]	7.13 [0.01]	3.40 [0.16]
vertebral-column-3c	35.48 [0.15]	34.15 [0.35]	3.13 [0.01]	3.16 [0.03]	8.33 [0.01]	2.90 [0.10]
waveform	8655.02 [86.83]	9915.83 [99.01]	242.94 [92.70]	826.80 [8.72]	416.87 [2.10]	461.40 [0.75]
wine	1.46 [0.01]	5.28 [0.19]	1.68 [0.01]	0.90 [0.00]	2.59 [0.01]	1.40 [0.16]
yeast	8545.91 [79.14]	9228.54 [94.72]	46.13 [4.30]	312.75 [2.72]	102.90 [0.90]	145.60 [1.37]

4.5 Computational Time

The average computational time (*average* [*standard error*]) in seconds[4] taken by each algorithm to complete a fold of the cross-validation (single execution) during training are presented in Table 9. The deterministic Unordered CN2, C4.5rules, PART and JRip algorithms take on average a second to complete a fold in any of the datasets used in the experiments, given that they employ heuristics to discover classification rules without the need to evaluate multiple candidate solutions. There are no differences in computational time during testing and all algorithms take on average a second to classify the test data.

Overall, the computational time taken by Unordered cAnt-Miner$_{\mathrm{PB}}$ is greater than

[4] On a 2.53GHz Intel Xeon CPU with 24GB RAM.

[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

the time taken by cAnt-Miner$_{\text{PB}}$, regardless of the conflict resolution strategy used. The main reason for the increase in time is the extra loop (line 8 of the pseudocode in Figure 3) to iterate over each class value, which is used to create rules for each class value using the examples associated with the remaining class values as negative examples—this is the main modification in order to create unordered rules. Since more iterations over the training data are required, greater increases in the computational time are observed in the *dermatology*, *glass*, *waveform* and *yeast* datasets over cAnt-Miner$_{\text{PB}}$: these datasets have either a greater number of classes and/or number of examples. Therefore, we can observe a trade-off of computational time and interpretability between Unordered cAnt-Miner$_{\text{PB}}$ and cAnt-Miner$_{\text{PB}}$: Unordered cAnt-Miner$_{\text{PB}}$ discovers modular rules that improve the interpretability of the classification model at the expense of a greater computational time. The remaining algorithms—with the exception of the deterministic algorithms—have a similar computational time to cAnt-Miner$_{\text{PB}}$, where increases in computational time are observed in larger datasets.

Note that many data mining applications are off-line and the time spent collecting, cleaning and structuring the data is much greater than the computational time taken by the classification algorithm to induce a model. Therefore, the computational time tend to have a minor importance and other aspects—such as predictive accuracy and interpretability—are more important. For applications where the computational time becomes significant, ACO algorithms can be easily parallelised by running each ant on an individual processing unit (e.g., core or processor) to reduce the overall computational time, since each ant builds and evaluates a candidate solution independently from all other ants.

5 Conclusion

In this paper we have proposed an extension to the cAnt-Miner$_{\text{PB}}$ in order to discover unordered rules, called Unordered cAnt-Miner$_{\text{PB}}$. 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 also proposed a new measure to characterise the interpretability of the discovered rules, called *prediction-explanation size*.

We compared the proposed Unordered cAnt-Miner$_{\text{PB}}$ algorithm against state-of-the-art rule induction algorithms in 32 publicly available datasets. The Unordered cAnt-Miner$_{\text{PB}}$ algorithm achieved the best results in terms of both predictive accuracy and prediction-explanation model size, outperforming state-of-the-art rule induction algorithms with statistically significant differences. 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. We also compared the Unordered cAnt-Miner$_{\text{PB}}$ algorithm against a SVM classifier. Our results shows that there are no statistically significant differences in the predictive accuracy obtained by Unordered cAnt-Miner$_{\text{PB}}$ and the SVM classifier. Overall, these results are positive: we were able to improve the interpretability of the rules by discovering unordered rules, with no negative impact on the predictive accuracy.

While in this work we have evaluated different rule quality and list quality functions, the Unordered cAnt-Miner$_{\text{PB}}$ algorithm's more specific parameters were not optimised, since we focused in comparing the differences between ordered and unordered

[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

rules. It might be possible to further improve the algorithm by performing an evaluation of different parameter settings. The use of different conflict resolution strategies can also lead to improvements to the predictive accuracy of the discovered rules. Similarly, the application of post-processing operators—such as the ones proposed in (Franco et al., 2012)—on the set of rules to either reduce their complexity or improve the predictive accuracy is also a research direction worth further exploration.

References

Bacardit, J., Burke, E., and Krasnogor, N. (2009). Improving the scalability of rule-based evolutionary learning. *Memetic Computing*, 1(1):55–67.

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

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, San Francisco, CA, USA. Morgan Kaufmann.

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

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

Franco, M., Krasnogor, N., and Bacardit, J. (2012). Post-processing Operators for Decision Lists. In *Genetic and Evolutionary Computation Conference (GECCO-2012)*, pages 847–854, New York, NY, USA. ACM Press.

Frank, E. and Witten, I. (1998). Generating Accurate Rule Sets Without Global Optimization. In Shavlik, J., editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 144–151, San Francisco, CA, USA. Morgan Kaufmann.

Freitas, A. (2013). Comprehensible classification models: a position paper. *ACM SIGKDD Explorations*, 15(1):1–10.

Freitas, A., Parpinelli, R., and Lopes, H. (2008). Ant colony algorithms for data classification. In *Encyclopedia of Information Science and Technology*, volume 1, pages 154–159. IGI Global, Hershey, PA, USA, 2nd edition.

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

Holden, N. and Freitas, A. (2008). A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *Journal of Artificial Evolution and Applications (JAEA), special issue on Particle Swarms: The Second Decade*. 11 pages.

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

[PRE-PRINT] Accepted for publication in: Evolutionary Computation, MIT Press, June 2015 (DOI: 10.1162/EVCO_a_00155)

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

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

Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. [http://archive.ics.uci.edu/ml].

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

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

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

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

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

Otero, F. and Freitas, A. (2013). Improving the Interpretability of Classification Rules Discovered by an Ant Colony Algorithm. In *Genetic and Evolutionary Computation Conference (GECCO-2013)*, pages 73–80, New York, NY, USA. ACM Press.

Otero, F., Freitas, A., and Johnson, C. (2008). *c*Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. In Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., and Winfield, A., editors, *Proceedings of the 6th International Conference on Swarm Intelligence (ANTS 2008), Lecture Notes in Computer Science 5217*, pages 48–59. Springer-Verlag.

Otero, F., Freitas, A., and Johnson, C. (2009). Handling continuous attributes in ant colony classification algorithms. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Data Mining (CIDM 2009)*, pages 225–231. IEEE.

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

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

Piatetsky-Shapiro, G. and Frawley, W. (1991). *Knowledge Discovery in Databases*. AAAI Press. 540 pages.

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

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, CA, USA. 316 pages.

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

Smaldon, J. and Freitas, A. (2006). A new version of the ant-miner algorithm discovering unordered rule sets. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 43–50, New York, NY, USA. ACM Press.

Tsumoto, S. (2000). 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, Berlin. Springer.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. Springer, 2nd edition. 314 pages.

Witten, H. and Frank, E. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition. 664 pages.