A New Sequential Covering Strategy for Inducing Classification Rules with Ant Colony Algorithms

Fernando E. B. Otero, Alex A. Freitas and Colin G. Johnson

Abstract

Ant colony optimization (ACO) algorithms have been successfully applied to discover a list of classification rules. In general, these algorithms follow a sequential covering strategy, where a single rule is discovered at each iteration of the algorithm in order to build a list of rules. The sequential covering strategy has the drawback of not coping with the problem of rule interaction—i.e., the outcome of a rule affects the rules that can be discovered subsequently since the search space is modified due to the removal of examples covered by previous rules. This paper proposes a new sequential covering strategy for ACO classification algorithms to mitigate the problem of rule interaction, where the order of the rules is implicitly encoded as pheromone values and the search is guided by the quality of a candidate list of rules. Our experiments using eighteen publicly available data sets show that the predictive accuracy obtained by a new ACO classification algorithm implementing the proposed sequential covering strategy is statistically significantly higher than the predictive accuracy of state-of-the-art rule induction classification algorithms.

Index Terms

ant colony optimization, data mining, classification, rule induction, sequential covering.

I. Introduction

DATA mining is a research area concentrated on designing and employing computational methods to discover (learn) a model (based on a given knowledge representation) from real-world structured data [1], [2]. Most of research is concentrated on supervised classification. A classification problem involves a set of examples, where each example is described by predictor attributes' values (features) and associated with a class value. The aim of a classification algorithm is to find a model that represents the relationships between predictor and class attributes' values. In general, the classification task involves two phases. In the first phase, the data set being mined is randomly split into training and test sets. Then, a classification model that represents the relationships between predictor and class attributes' values is built by analysing the examples from the training set. Note that the algorithm has access to the information of both predictor and class attributes from the training set. In the second phase, the classification model is used to classify—i.e., predict the value of the class attribute—the examples from the test set. Considering that the classification model was built using only the examples from the training set, the algorithm has no information about the class value of the examples from the test set. The value of the class attribute of a test example is only verified after the classification algorithm predicted its value, in order to evaluate the created classification model. A prediction

The authors are with the School of Computing, University of Kent, Canterbury, Kent, United Kingdom (email: {F.E.B.Otero, A.A.Freitas, C.G.Johnson}@kent.ac.uk).

is considered correct when the predicted value is the same as the actual value of the example; otherwise it is considered incorrect. The more correct predictions on the test set, the more accurate the classification model.

One of the main goals of a classification algorithm is to build a model that maximises the predictive accuracy—i.e., the number of correct predictions divided by the total number of predictions—in the test set, although in some application domains (e.g. credit approval, medical diagnosis and protein function prediction) the comprehensibility of the model plays an important role [3], [4]. For instance, both neural networks and support vector machines (SVMs) are successful methods in term of predictive accuracy when applied to classification, but they produce classification models that are not easily interpretable. Rule induction algorithms are widely used to produce comprehensible classification models in the form of a list of *IF-THEN* classification rules, which generally can be expressed in natural language.

Ant colony optimization (ACO) [5], [6], [7] algorithms have been successfully applied to discover a list of *IF-THEN* classification rules. Ant colonies, despite the lack of centralised control and the relative simplicity of their individuals' behaviours, are self-organised systems which can accomplish complex tasks by having their individual ants interact with one another and with their environment. The intelligent behaviour of the colony emerges from the indirect communication between the ants mediated by small modifications of the environment. Many ant species, even with limited visual capabilities or completely blind, are able to find the shortest path between a food source and the nest by using pheromone as a communication mechanism. Ants drop pheromone on the ground as they walk from a food source to the nest, thereby creating a pheromone trail on the used path. The pheromone concentration of a path influences the choices ants make, and the more pheromone the more attractive a path becomes. Given that shorter paths are traversed faster than longer ones, they have a stronger pheromone concentration after a period time, contributing to being selected and reinforced more often. Ultimately the majority of ants will be following the same path, most likely the shortest path between the food source and the nest.

ACO algorithms use a colony of artificial ants, where ants build candidate solutions to optimization problems by iteratively selecting solution components based on their associated pheromone and heuristic information—where the latter corresponds to a measure of how good a solution component is for the problem at hand. The colony cooperates by using pheromone to identify prominent components of a solution and the components with higher concentration of pheromone have a greater chance of being selected by an ant. Components used to create good solutions have their pheromone increased, while components not used will have their pheromone gradually decreased. At the end of the iterative process of building candidate solutions guided by pheromone, the colony converges to optimal or near-optimal solutions.

This paper presents a discussion of the strategy commonly used by ACO classification algorithms to build a list of classification rules and proposes a new strategy that mitigates its potential disadvantages. We are particular interested in improving the search performed by the ACO algorithm using the quality of a candidate list of rules as a feedback—represented by pheromone values—for building other lists, instead of using the feedback for creating a single rule. We evaluate the impact of the new strategy in terms of both predictive accuracy and size of the classification model (discovered list of rules), and compare the results against state-of-the-art rule induction algorithms.

The remainder of this paper is organised as follows. Section II presents a discussion of the current strategy used in ACO classification algorithms. Section III presents the new sequential covering strategy proposed in this paper. The computational

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

Fig. 1. High-level pseudocode of the Ant-Miner algorithm.

results are presented in Section IV. Finally, Section V concludes this paper and presents future research directions.

II. Ant Colony Classification Algorithms

Research on ant colony optimization algorithms for the classification task initiated with the Ant-Miner algorithm, proposed by Parpinelli et al. [8], [9]. Ant-Miner aims at discovering a list of classification rules by applying a sequential covering strategy, which consists of creating one-rule-at-a-time until all training examples are covered by one of the rules in the discovered list. In order to create a rule, Ant-Miner uses an ant colony optimization (ACO) procedure to find the best rule given a set of training examples. The high-level pseudocode of Ant-Miner is presented in Fig. 1.

In summary, Ant-Miner works as follows. It starts with an empty list of rules and iteratively (outer *while* loop) adds one rule at a time to that list while the number of uncovered training examples is greater than a maximum uncovered value in a sequential covering fashion. At each iteration, a rule is created by an ACO procedure (inner *while* loop). In order to create a rule, ants probabilistically select terms to be added to their current partial rule based on the values of the amount of pheromone and a problem-dependent heuristic information. Ants keep adding a term to their partial rule until any term added to their rule's antecedent would make their rule cover fewer training examples than a minimum value in order to avoid too specific and unreliable rules, or until all attributes have already been used.

Once the rule construction process has finished, the rule created by an ant is pruned to remove irrelevant terms from the rule antecedent. The pruning procedure can be seen as a local search operator that explores neighbour solutions by attempting to remove terms from the antecedent of a rule while the quality of the rule does not decrease. Then, the consequent of a rule is chosen to be the class value most frequent amongst the set of training examples covered by the rule in question. Finally, pheromone trails are updated using the best rule—based on a quality measure Q—of the current iteration and the best-so-far rule across all iterations is stored/updated. The process of constructing a rule is repeated until the maximum number of iterations has been reached, or the best rule of the current iteration is exactly the same as the best rule constructed by a specified number of previous iterations, which works as a stagnation test. The best rule found along this iterative process is added to the list of rules and the covered training examples (training examples that satisfy the antecedent of the best rule) are removed from the training set and the procedure of creating a rule is repeated.

Following the introduction of Ant-Miner, several variations were proposed in the literature, as recently reviewed in [10]. These variations include the evaluation of different measures/procedures of the algorithm—i.e., heuristic information [11]; pheromone update and rule construction procedures [12]; pruning procedure [13]. More elaborated variations are able to cope with both nominal and continuous attributes [14], [15], overcoming Ant-Miner's limitation of being able to cope only with nominal attributes; provide a new construction graph exploiting the difference between nominal and ordinal attributes, where ants first select the class value predicted by the rule, allowing the algorithm to use class-specific heuristic information [16]; and the use of both class-specific heuristic information and pheromone matrices [17], [18].

Overall, Ant-Miner and its variations share the same sequential covering strategy in order to build a list of rules that covers all training examples. The algorithm starts with an empty list of rules and iteratively adds one-rule-at-a-time to the list using an ACO procedure (lines 8–22 in Fig. 1) that aims at creating a single rule that maximises a specified rule quality measure, until there are no training cases to cover.

Drawing a comparison concerning the rule discovery strategy with the broader area of evolutionary algorithms, the sequential covering strategy employed in Ant-Miner falls into the iterative rule learning (IRL) approach [19], [20]. In the IRL approach, each run of the evolutionary procedure—analogous to the ACO procedure in Ant-Miner case—discovers a single rule (the best rule produced over all iterations) and the procedure is repeated multiple times in order to discover a list of rules. Two other approaches for rule discovery have been used in the evolutionary algorithm literature: the Michigan [21], [22] and the Pittsburgh [23], [24] approaches. In the Michigan approach, each individual corresponds to a rule and a list of rules is represented by the entire population, using some mechanism to ensure that different rules cover different regions of the data space. Hence, a single run of an evolutionary procedure following a Michigan approach discovers a complete list of rules. Similarly, in the Pittsburgh approach, each run of the evolutionary procedure discovers a complete list of rules (the best list of rules produced over all iterations).

One of the main differences between IRL/Michigan and Pittsburgh approaches is that in the latter a complete list of rules, which constitutes an individual, is evaluated instead of a single rule, in order to guide the discovery process. As discussed in [25], evaluating the quality of a rule individually, instead of the quality of a list of rules as a whole, has difficulty with the problem of rule interaction—i.e., the list of best rules is not necessarily the best list of rules.

III. A New Sequential Covering Strategy for Ant Colony Classification Algorithms

In Ant-Miner's sequential covering strategy, the discovery of a rule can be seen as an independent search problem for the best rule given the current set of training examples. In each iteration of the sequential covering, a rule is constructed by an ACO procedure and the examples covered by the rule are removed from the data set. This iterative process of constructing a rule is repeated until the training set is empty (or almost empty). Although rules are discovered in an one-at-a-time fashion, the outcome of a rule (i.e., the examples covered by the rule) affects the rules that can be discovered subsequently since the search space is modified due to the removal of examples covered by previous rules. Therefore, the sequential covering performs a greedy search for a list (sequence) of rules which is not guaranteed to be the best list of rules that covers the training set, since the interaction between them is not taken into account during the search.

The proposed strategy incorporates ideas of the Pittsburgh approach into Ant-Miner's sequential covering strategy in order to mitigate the problem of rule interaction. While it still relies on a sequential covering strategy to create a list of rules, an *ant creates a complete list of rules* at each iteration of the algorithm instead of a single rule and the *search is guided by the quality of a list of rules*. This is accomplished by using a sequential covering strategy in which a rule created at each iteration of the covering process does not necessarily correspond to the best rule; and by having pheromone values, which are updated according to the quality of the best candidate list of rules amongst all lists built in an iteration, guiding the rule construction process.

Fig. 2 presents the high-level pseudocode of the new sequential strategy proposed. In summary, the new strategy works as follows. An ant in the colony (corresponding to an iteration of the *for* loop) starts with an empty list of rules and adds one rule at a time to that list while the number of uncovered training examples is greater than a user-specified maximum value. After a rule is created and pruned, the training examples covered by the rule are removed and the rule is added to the current list of rules. Note that the heuristic information is re-calculated at each iteration of the list creation process (*while* loop) in order to reflect the potential changes in the predictive power of the terms due to the removal of training examples covered by previous rules. When an ant finishes the list creation process, the iteration-best list is updated if the quality of the newly created list is greater than the quality of the iteration-best list. After all ants create a candidate list of rules, pheromone values are updated using the iteration-best list of rules and the global-best list of rules is updated, if the quality of the iteration-best list is greater than the quality of the global-best list.

In order to use pheromone to create multiple rules covering different set of training examples, the pheromone matrix is extended to include a *tour* identification, which corresponds to the number of the rule being created (e.g., 1 for the first rule, 2 for the second rule and so forth). Each entry in the pheromone matrix corresponding to an edge of the construction graph is represented not just by a pair ($vertex_i$, $vertex_j$)—where $vertex_i$ and $vertex_j$ correspond to the vertices connected by $edge_{ij}$—but rather it is represented by a triple ($tour$, $vertex_i$, $vertex_j$). This way, an ant will use the pheromone entries corresponding to the number of the rule (tour) being created during the rule construction process. The probability of an ant to follow the edge leading to a vertex v_j when creating the rule t and located at vertex v_i is given by

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

Fig. 2. High-level pseudocode of the new sequential covering strategy.

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

where $\tau_{(t,v_i,v_j)}$ is the amount of pheromone associated with the entry (t, v_i, v_j) 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 neighbour 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.

The pheromone update also takes into account the tour identification and the update procedure is accomplished in two steps. Firstly, pheromone evaporation is simulated by decreasing the amount of pheromone of each entry by a user-defined factor ρ. Secondly, the amount of pheromone of the entries used in the iteration-best list of rules is increased based on the quality of the list of rules, which corresponds to its predictive accuracy measured on the training set. The pheromone update rule is given by

$$\tau_{(t,v_i,v_j)} = \begin{cases} \rho \cdot \tau_{(t,v_i,v_j)}, & \text{if } (t, v_i, v_j) \notin \textit{list}_{ib}; \\ \rho \cdot \tau_{(t,v_i,v_j)} + Q(\textit{list}_{ib}), & \text{if } (t, v_i, v_j) \in \textit{list}_{ib}; \end{cases} \tag{2}$$

where ρ is the evaporation factor, $\tau_{(t,v_i,v_j)}$ is the amount of pheromone associated with the entry (t, v_i, v_j)—t is the tour identification (i.e., the number of the rule where the edge between vertices v_i and v_j was used), v_i is the start vertex of the edge and v_j is the end vertex of the edge—and $Q(list_{ib})$ is the quality of the iteration-best list of rules, measured as the predictive accuracy (number of correct predictions divided by the total number of predictions) in the training set. The values given by Eq. 2 are limited to the interval $[\tau_{min}, \tau_{max}]$, following the same approach as the $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [26], [27]. $\mathcal{MM}$AS imposes explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone values to constrain all pheromone values $\tau_{(t,v_i,v_j)}$ to the range $\tau_{min} \leq \tau_{(t,v_i,v_j)} \leq \tau_{max}$. These limits are dynamically updated each time a new best solution is found, as detailed in [27]. Additionally, the τ_{min} and τ_{max} values are also used to determine the stagnation of the search. When all edges followed by the ant that created the iteration-best list of rules are associated with τ_{max} and the remaining edges are associated with τ_{min}, the search has become stagnant and the algorithm stops.

The proposed sequential covering strategy is implemented in a new ACO classification algorithm, named cAnt-Miner$_{PB}$ (cAnt-Miner based on the Pittsburgh approach). The other aspects of the cAnt-Miner$_{PB}$ algorithm—e.g., rule construction process, pruning procedure and heuristic information—are based on the cAnt-Miner2$_{MDL}$ algorithm, as discussed in [14], [15]. Note that the proposed cAnt-Miner$_{PB}$ algorithm—like the cAnt-Miner2$_{MDL}$ algorithm—can cope with both nominal and continuous attributes, unlike the original Ant-Miner algorithm, which can cope with nominal attributes only.

An important characteristic of the proposed sequential covering strategy is that there is no pre-defined number of rules required to create a candidate list of rules and ants have the flexibility of creating lists of differences lengths. The number of rules that an ant creates depends on the available training examples at each iteration of the list creation process (inner *while* loop in Fig. 2), which varies according to examples covered by the previous rules created by the ant. The use of a different set of pheromone values for each rule they are creating indirectly encodes the order (sequence) that ants create the rules, which represents the interaction between them. This highlights the main difference of the proposed algorithm: the aim of the algorithm is to converge to the best list of rules, instead of converge to the list of best rules as in previous Ant-Miner variations.

IV. Computational Results

In order to evaluate the proposed cAnt-Miner$_{PB}$ algorithm, we carried out experiments using eighteen publicly available data sets from the UCI Machine Learning repository [28]. The data sets involve binary (two class values) and multiclass (more than two class values) classification problems, with both nominal and continuous predictor attributes. Table I presents a summary of the data sets used in the experiments.

To compare with the results obtained by cAnt-Miner$_{PB}$, we have selected commonly used state-of-the-art and also a previously proposed ACO-based rule induction classification algorithms:

- cAnt-Miner2$_{MDL}$ [14], [15] – a variation of the Ant-Miner algorithm that copes with both nominal and continuous attributes directly by using a minimum description length (MDL) principle [29] to dynamically create thresholds on continuous attributes' domain values during the rule construction process. This is also the base algorithm from which the proposed algorithm is built on;

TABLE I
SUMMARY OF THE DATA SETS USED IN THE EXPERIMENTS.

data set	attributes		classes	size
	nominal	continuous		
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

- PSO/ACO2 [30] – 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 by dividing the rule construction process into two steps: in the first step, only nominal attributes are considered to create the antecedent of a rule; then, in the second step, continuous attributes are considered to extend the antecedent of a rule;
- CN2 [31] – a well-known rule induction algorithm following a sequential covering strategy, which uses a beam search at each iteration to create a rule to build a list of rules. Therefore, CN2 uses the same strategy than Ant-Miner, with the difference that the latter uses a ACO procedure to create a rule;
- C4.5rules [32] – a rule induction algorithm that extracts a set of classification rules from an unpruned decision tree created by the well-known C4.5 algorithm [32], [33]. The algorithm first converts every path of the tree from a leaf node towards the root node to a rule and then applies a rule post-pruning procedure. Finally, the rules are sorted according to their confidence (predictive accuracy on the training set) to create the final set of rules;
- PART [34], [35] – a rule induction algorithm that combines a sequential covering strategy with a decision tree induction procedure to create a rule. At each iteration of the sequential covering, PART builds a decision tree—using the well-known C4.5 algorithm—for the current set of training examples and then selects the leaf with the largest coverage to create a rule, discarding the rest of the tree;
- JRip [35] (Weka's implementation of RIPPER [36]) – a rule induction algorithm that employs a global optimisation step in order to produce a set of rules, which takes into account both the quality and length of the rules. It starts by creating a set of rules for each class value using a sequential covering strategy. Then, each rule is reconsidered and variants are produced using a reduced-error pruning. If one of the variants produces a smaller length, it replaces the rule.

Note that C4.5rules and PART are not 'pure' rule induction algorithms, and are in fact hybrid decision tree/rule induction algorithms. They have been included in the experimental comparison because both are strong, well-known algorithms that eventually produce a list of rules in the same representation as the list of rules produced by the proposed *c*Ant-Miner$_{PB}$ algorithm.

We have performed a ten-fold cross-validation procedure, which consists of dividing the data set into ten partitions, maintaining a similar number of examples and class distribution across all partitions. For each partition, the classification algorithm is run using the remaining nine partitions as the training set and its performance is evaluated using the unseen (hold-out) partition. Since *c*Ant-Miner2$_{MDL}$, PSO/ACO2 and *c*Ant-Miner$_{PB}$ are stochastic algorithms, they are run fifteen times using a different random seed to initialise the search for each partition of the cross-validation; the remaining algorithms are deterministic and they are run just once for each partition of the cross-validation.

A. Parameter Settings

To determine suitable values for the user-defined parameters of *c*Ant-Miner$_{PB}$, we have used the F-Race [37] racing procedure to find a good configuration of parameters. *c*Ant-Miner$_{PB}$ has five parameters: the *maximum number of iterations*, the *colony size* (number of ants), the $\mathcal{MAX}-\mathcal{MIN}$ *evaporation factor*, the *minimum number of examples* covered by a rule and the *maximum uncovered* training examples. Since the values of *minimum number of examples* and *maximum uncovered* parameters are related—i.e., the *maximum uncovered* should be at least the same as the *minimum number of examples*—we have use the same value for both, specified by the *minimum number of examples* parameter. Additionally, we have set the *maximum number of iterations* to 500, given that the convergence test is able to stop the search before reaching the maximum value. The values of the remaining three parameters (*colony size*, *evaporation factor* and *minimum number of examples*) are determined by the F-Race procedure. We have considered the values of *colony size* $\in \{5, 10, 50, 100, 200\}$, *evaporation factor* $\in \{0.85, 0.90, 0.95\}$ and *minimum number of examples* $\in \{2, 6, 10, 14\}$. Each possible combination of values represents a different configuration, leading to $5 \times 3 \times 4 = 60$ configurations that are subjected to the racing procedure.

We have selected additional eight tuning data sets from the UCI Machine Learning repository[1] to evaluate the set of candidate configurations using the F-Race. Note that the aim of the racing procedure is not to optimise the parameters for a particular data set, but to find robust values that work well across the tuning data sets. We then use the robust parameter settings found by the racing procedure in the eight tuning data sets as the parameter settings for the different set of eighteen data sets used in the our experiment. This evaluates the generalization ability of the parameter settings found by the racing procedure across new data sets, unused for parameter tuning, as usual in supervised machine learning – the standard approach in the literature to evaluate a new classification algorithm is to run it with the same parameter settings across a number of data sets, rather than optimizing the parameters for each data set in turn.

In summary, the racing procedure in F-Race evaluates candidate configurations in a subset of the data sets available and eliminates the poor ones as soon as it detects that they are statistically inferior—according to the non-parametric Friedman test [38]—than the best one. The configurations that survive an evaluation step are re-evaluated in an extended subset and

[1]automobile, blood-transfusion, ecoli, statlog heart, hepatitis, horse-colic, voting-records, zoo.

undergo another elimination step. The racing procedure is iterated until only one configuration is left or when all data sets are used. In the case that more than one configuration survives the procedure, the configuration with the highest rank is selected. In our case, each configuration evaluation corresponds to the predictive performance of cAnt-Miner$_{PB}$ using the configuration parameters over a ten-fold cross-validation. At the end of the F-Race procedure, we ended up with 11 (out of the 60 available) configurations and the configuration *colony size* $= 5$, *evaporation factor* $= 0.90$ and *minimum number of examples* $= 10$ was the one with the higest rank, and therefore, the one used in our experiments.

The other algorithms were used with the default values proposed by their correspondent authors, which typically represent robust values that work well across different data sets. None of the algorithms, including cAnt-Miner$_{PB}$, had their parameter values optimised to individual data sets, since the goal is to evaluate a parameter settings' generalisation ability across a wide range of data sets as usual in the classification (supervised machine learning) literature.

B. Comparison with state-of-the-art algorithms

Table II presents the results concerning the predictive accuracy, Table III presents the results concerning the size of the classification model, measured as the total number of terms (rule conditions) in the discovered list—where the smaller the number of terms the simpler is the classification model—and Table IV presents the average number of rules (indicated by rows starting with a symbol '*(r)*') and average number of terms per rule (indicated by rows starting with a symbol '*(t)*') of the discovered list in the eighteen data sets used in our experiments. A value in these tables represents the average value obtained by the cross-validation procedure followed by the standard error (average $\pm$ standard error) for the corresponding algorithm and data set pair. Table V presents the results of the statistical tests for predictive accuracy, size of the classification model and total number of discovered rules, according to the non-parametric Friedman test with the Holm's post-hoc test [39], [40]—the first column shows the average rank, where the lower the rank the better the algorithm's performance; the second 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 (control algorithm); the third shows the Holm's critical value. The values in a row are shown in bold when there is a statistically significant difference at the 5% level between the average ranks of an algorithm and the control algorithm, determined by the fact that the p-value is lower than the critical value, and it shows that the control algorithm is significantly better than the algorithm in that row. The non-parametric Friedman test was chosen as it does not make assumptions about the distribution of the underlying data (e.g., it does not assume the data is normally distributed, a requirement for equivalent parametric tests) and it is a more suitable test to compare a set of classifiers over multiple data sets, according to the guidelines presented in [39], [40].

Considering the predictive accuracy, cAnt-Miner$_{PB}$ achieves the highest performance with an average rank of 2.28 across all data sets, which is statistically significantly better than the average ranks obtained by all the other six algorithms according to the non-parametric Friedman test with the Holm's post-hoc test; C4.5rules, cAnt-Miner2$_{MDL}$, PSO/ACO2 and PART have similar performances, with an average rank of 4.00, 4.05, 411 and 4.11, respectively; JRip and CN2 have the worst performances, with an average rank of 4.67 and 4.78, respectively. cAnt-Miner$_{PB}$ is also the most accurate algorithm in eight of the eighteen data sets, followed by CN2 in three data sets, C4.5rules and JRip in two data sets each, cAnt-Miner2$_{MDL}$, PSO/ACO2 and PART

TABLE II
AVERAGE PREDICTIVE ACCURACY (*average* ± *standard error*) IN %, MEASURED BY 10-FOLD CROSS-VALIDATION. THE VALUE OF MOST ACCURATE ALGORITHM FOR A GIVEN DATA SET IS SHOWN IN BOLD.

data set	*c*Ant-Miner2$_{MDL}$	PSO/ACO2	CN2	C4.5rules	PART	JRip	*c*Ant-Miner$_{PB}$
annealing	95.95 ± 0.20	97.25 ± 0.08	94.99 ± 0.62	94.22 ± 0.62	94.88 ± 0.98	94.43 ± 0.81	**97.60 ± 0.10**
balance-scale	67.92 ± 0.10	79.16 ± 0.19	**88.47 ± 1.81**	74.87 ± 1.16	77.12 ± 1.40	72.95 ± 1.92	76.83 ± 0.24
breast-l	**76.03 ± 0.08**	71.29 ± 0.31	66.77 ± 1.69	68.56 ± 1.93	68.94 ± 1.80	69.26 ± 2.04	72.32 ± 0.31
breast-tissue	69.51 ± 0.26	62.13 ± 0.71	**75.35 ± 3.08**	66.16 ± 2.97	64.36 ± 3.63	60.18 ± 3.35	67.13 ± 0.55
breast-w	93.69 ± 0.20	94.14 ± 0.15	93.84 ± 1.21	94.18 ± 1.14	94.19 ± 1.12	93.66 ± 1.42	**94.29 ± 0.16**
credit-a	86.37 ± 0.12	84.66 ± 0.24	81.31 ± 1.36	85.53 ± 1.53	83.33 ± 1.04	**86.52 ± 1.10**	85.68 ± 0.15
credit-g	71.47 ± 0.19	69.99 ± 0.23	69.50 ± 1.33	71.60 ± 0.92	70.60 ± 1.49	72.20 ± 1.07	**73.63 ± 0.23**
cylinder-bands	72.47 ± 0.28	69.75 ± 0.54	75.93 ± 2.19	**76.48 ± 2.56**	72.41 ± 2.23	68.70 ± 2.33	72.07 ± 0.39
dermatology	89.59 ± 0.41	91.80 ± 0.22	91.79 ± 1.37	93.45 ± 1.22	**94.26 ± 1.17**	88.01 ± 2.25	92.46 ± 0.31
glass	67.58 ± 0.36	70.24 ± 0.51	68.14 ± 2.39	68.63 ± 1.70	72.81 ± 3.42	65.71 ± 3.74	**73.94 ± 0.49**
heart-c	54.66 ± 0.24	55.20 ± 0.51	48.85 ± 3.14	53.12 ± 1.92	53.83 ± 1.33	53.50 ± 1.52	**55.50 ± 0.37**
heart-h	63.71 ± 0.30	63.18 ± 0.45	52.81 ± 3.38	63.31 ± 1.40	63.64 ± 1.58	63.93 ± 1.29	**64.75 ± 0.27**
ionosphere	88.37 ± 0.21	86.55 ± 0.43	88.03 ± 2.68	**90.85 ± 2.59**	90.59 ± 2.00	87.45 ± 2.64	89.65 ± 0.31
iris	92.67 ± 0.18	94.84 ± 0.20	94.66 ± 1.66	95.32 ± 1.42	93.33 ± 1.99	**96.00 ± 1.09**	93.24 ± 0.20
liver-disorders	65.46 ± 0.03	**68.78 ± 0.44**	62.27 ± 2.84	64.90 ± 3.21	62.70 ± 3.40	66.34 ± 2.80	66.72 ± 0.40
parkinsons	86.19 ± 0.33	86.77 ± 0.50	85.08 ± 2.62	83.49 ± 2.23	86.05 ± 2.47	84.53 ± 2.55	**86.98 ± 0.65**
pima	73.99 ± 0.19	73.10 ± 0.33	72.37 ± 1.26	74.32 ± 1.73	71.73 ± 1.71	73.55 ± 1.63	**74.81 ± 0.18**
wine	90.82 ± 0.39	88.14 ± 0.52	**94.96 ± 1.94**	91.03 ± 2.05	91.54 ± 1.52	92.68 ± 2.09	93.57 ± 0.32

TABLE III
AVERAGE NUMBER OF TERMS (RULE CONDITIONS) IN THE DISCOVERED LIST (*average* ± *standard error*) MEASURED BY 10-FOLD CROSS-VALIDATION. THE VALUE OF THE ALGORITHM WITH THE LOWEST AVERAGE FOR A GIVEN DATA SET IS SHOWN IN BOLD.

data set	*c*Ant-Miner2$_{MDL}$	PSO/ACO2	CN2	C4.5rules	PART	JRip	*c*Ant-Miner$_{PB}$
annealing	**16.35 ± 0.15**	28.60 ± 0.91	52.10 ± 2.72	43.00 ± 3.05	85.90 ± 3.10	26.50 ± 1.81	22.11 ± 0.33
balance-scale	12.77 ± 0.08	39.87 ± 0.53	90.20 ± 3.51	72.60 ± 3.27	45.90 ± 3.00	22.60 ± 1.99	**12.64 ± 0.03**
breast-l	7.42 ± 0.03	23.20 ± 1.80	141.60 ± 1.98	16.50 ± 2.60	35.30 ± 3.70	**4.10 ± 0.82**	19.15 ± 0.40
breast-tissue	8.74 ± 0.05	12.73 ± 0.46	23.60 ± 0.79	19.40 ± 1.05	21.80 ± 0.85	7.50 ± 0.73	**6.55 ± 0.05**
breast-w	12.79 ± 0.13	13.87 ± 0.70	19.40 ± 0.75	18.70 ± 0.98	11.70 ± 0.73	9.70 ± 0.30	**8.55 ± 0.12**
credit-a	13.31 ± 0.12	80.27 ± 2.10	96.30 ± 2.57	37.10 ± 3.06	75.50 ± 8.79	**7.30 ± 1.77**	17.54 ± 0.32
credit-g	16.91 ± 0.20	227.73 ± 3.51	222.50 ± 2.19	76.90 ± 4.99	219.60 ± 10.82	**6.80 ± 0.83**	64.75 ± 1.50
cylinder-bands	**16.34 ± 0.18**	56.67 ± 1.99	135.20 ± 2.81	95.30 ± 3.36	94.90 ± 2.66	18.00 ± 2.95	63.37 ± 1.03
dermatology	**20.43 ± 0.37**	30.80 ± 0.31	48.60 ± 0.93	45.00 ± 1.89	25.00 ± 1.12	24.50 ± 1.57	44.47 ± 0.63
glass	16.34 ± 0.13	60.87 ± 1.12	43.90 ± 1.08	48.30 ± 2.05	41.20 ± 2.15	12.90 ± 1.69	**10.73 ± 0.14**
heart-c	20.66 ± 0.22	62.20 ± 1.16	110.10 ± 2.48	50.80 ± 4.61	136.30 ± 4.76	**6.80 ± 1.50**	27.65 ± 0.58
heart-h	15.79 ± 0.28	48.67 ± 2.43	108.30 ± 2.00	47.80 ± 2.80	73.50 ± 3.84	**5.80 ± 1.00**	21.49 ± 0.41
ionosphere	14.06 ± 0.19	11.73 ± 1.34	23.30 ± 0.87	19.50 ± 1.74	19.80 ± 1.46	**8.00 ± 1.12**	11.04 ± 0.17
iris	3.93 ± 0.01	**2.00 ± 0.00**	9.50 ± 0.62	6.20 ± 0.20	3.80 ± 0.42	3.00 ± 0.00	4.92 ± 0.08
liver-disorders	8.77 ± 0.06	67.73 ± 1.07	76.50 ± 1.37	33.50 ± 3.42	20.10 ± 2.33	**8.50 ± 0.82**	11.78 ± 0.08
parkinsons	9.19 ± 0.07	10.33 ± 0.45	15.30 ± 0.33	18.90 ± 1.29	11.10 ± 0.50	**6.30 ± 0.67**	7.02 ± 0.11
pima	13.17 ± 0.17	126.93 ± 2.56	135.50 ± 2.05	36.60 ± 4.09	14.60 ± 0.60	**6.80 ± 0.71**	15.93 ± 0.14
wine	7.21 ± 0.05	6.27 ± 0.43	7.90 ± 0.31	8.90 ± 0.48	5.20 ± 0.47	5.40 ± 0.37	**4.75 ± 0.08**

TABLE IV

AVERAGE NUMBER OF RULES, INDICATED BY ROWS STARTING WITH A SYMBOL '*(r)*', AND AVERAGE NUMBER OF TERMS PER RULE, INDICATED BY ROWS STARTING WITH A SYMBOL '*(t)*'. EACH VALUE REPRESENTS THE AVERAGE (*average* ± *standard error*) MEASURED BY 10-FOLD CROSS-VALIDATION. THE VALUE OF THE ALGORITHM WITH THE LOWEST AVERAGE NUMBER OF RULES FOR A GIVEN DATA SET IS SHOWN IN BOLD.

data set		*c*Ant-Miner2$_{MDL}$	PSO/ACO2	CN2	C4.5rules	PART	JRip	*c*Ant-Miner$_{PB}$
annealing	*(r)*	**11.31 ± 0.06**	15.51 ± 0.09	19.80 ± 0.47	20.30 ± 0.89	27.80 ± 0.83	11.80 ± 0.39	16.12 ± 0.13
	(t)	1.45 ± 0.01	1.91 ± 0.01	2.67 ± 0.10	2.10 ± 0.07	3.09 ± 0.05	2.23 ± 0.09	1.37 ± 0.01
balance-scale	*(r)*	12.82 ± 0.05	25.23 ± 0.16	50.80 ± 1.57	35.70 ± 1.43	29.90 ± 1.46	**12.10 ± 0.78**	13.64 ± 0.03
	(t)	0.99 ± 0.00	1.54 ± 0.00	1.79 ± 0.01	2.03 ± 0.01	1.52 ± 0.03	1.84 ± 0.04	0.93 ± 0.00
breast-l	*(r)*	5.89 ± 0.01	12.97 ± 0.13	60.20 ± 0.77	8.60 ± 1.08	18.40 ± 1.65	**3.10 ± 0.35**	10.36 ± 0.11
	(t)	1.26 ± 0.00	1.75 ± 0.02	2.37 ± 0.03	1.83 ± 0.10	1.90 ± 0.06	1.20 ± 0.16	1.80 ± 0.02
breast-tissue	*(r)*	6.41 ± 0.01	**5.91 ± 0.03**	12.20 ± 0.36	9.00 ± 0.36	10.90 ± 0.31	6.00 ± 0.30	6.73 ± 0.03
	(t)	1.37 ± 0.01	1.96 ± 0.03	2.02 ± 0.05	2.15 ± 0.04	2.00 ± 0.06	1.24 ± 0.10	0.97 ± 0.00
breast-w	*(r)*	6.01 ± 0.03	6.29 ± 0.03	9.90 ± 0.31	8.90 ± 0.28	6.70 ± 0.30	**5.40 ± 0.16**	8.33 ± 0.10
	(t)	2.13 ± 0.02	2.24 ± 0.03	2.06 ± 0.04	2.09 ± 0.05	1.74 ± 0.07	1.81 ± 0.08	1.02 ± 0.01
credit-a	*(r)*	7.97 ± 0.04	25.30 ± 0.11	35.50 ± 0.70	14.00 ± 0.89	31.10 ± 2.63	**4.20 ± 0.57**	12.31 ± 0.10
	(t)	1.68 ± 0.02	3.19 ± 0.02	2.74 ± 0.05	2.63 ± 0.09	2.38 ± 0.09	1.44 ± 0.24	1.41 ± 0.02
credit-g	*(r)*	8.79 ± 0.05	54.15 ± 0.12	74.40 ± 0.93	27.50 ± 1.27	71.70 ± 2.88	**3.60 ± 0.22**	28.57 ± 0.27
	(t)	1.92 ± 0.02	4.24 ± 0.01	3.01 ± 0.02	2.78 ± 0.07	3.06 ± 0.05	1.84 ± 0.12	2.23 ± 0.04
cylinder-bands	*(r)*	7.61 ± 0.03	13.11 ± 0.10	37.30 ± 0.62	29.90 ± 0.81	33.80 ± 0.76	**7.30 ± 0.84**	27.63 ± 0.25
	(t)	2.14 ± 0.02	4.32 ± 0.05	3.65 ± 0.06	3.19 ± 0.06	2.81 ± 0.08	2.26 ± 0.22	2.28 ± 0.02
dermatology	*(r)*	**9.24 ± 0.10**	10.29 ± 0.08	18.80 ± 0.44	19.60 ± 0.34	9.60 ± 0.34	13.10 ± 0.62	19.17 ± 0.11
	(t)	2.20 ± 0.03	2.57 ± 0.01	2.65 ± 0.05	2.29 ± 0.08	2.60 ± 0.07	1.86 ± 0.06	2.31 ± 0.02
glass	*(r)*	8.74 ± 0.03	20.34 ± 0.08	16.70 ± 0.26	14.90 ± 0.50	15.20 ± 0.47	**7.20 ± 0.57**	9.41 ± 0.08
	(t)	1.87 ± 0.01	2.98 ± 0.02	2.69 ± 0.08	3.24 ± 0.06	2.70 ± 0.08	1.73 ± 0.09	1.14 ± 0.01
heart-c	*(r)*	9.21 ± 0.04	15.87 ± 0.05	34.40 ± 0.40	14.90 ± 1.08	41.30 ± 1.28	**3.30 ± 0.47**	12.85 ± 0.14
	(t)	2.24 ± 0.02	3.96 ± 0.03	3.23 ± 0.06	3.37 ± 0.10	3.31 ± 0.01	1.94 ± 0.15	2.13 ± 0.03
heart-h	*(r)*	7.41 ± 0.06	14.75 ± 0.08	35.10 ± 0.75	12.90 ± 0.59	25.10 ± 1.02	**3.30 ± 0.33**	10.96 ± 0.12
	(t)	2.12 ± 0.03	3.50 ± 0.02	3.12 ± 0.05	3.69 ± 0.06	2.93 ± 0.09	1.59 ± 0.21	1.94 ± 0.02
ionosphere	*(r)*	7.07 ± 0.07	**3.36 ± 0.07**	10.40 ± 0.31	10.00 ± 0.36	8.20 ± 0.42	5.80 ± 0.44	9.18 ± 0.09
	(t)	1.98 ± 0.01	3.08 ± 0.07	2.34 ± 0.07	1.92 ± 0.12	2.41 ± 0.12	1.31 ± 0.10	1.20 ± 0.01
iris	*(r)*	4.10 ± 0.00	**3.00 ± 0.00**	6.80 ± 0.29	5.00 ± 0.00	3.80 ± 0.42	4.00 ± 0.00	4.82 ± 0.05
	(t)	0.95 ± 0.00	0.67 ± 0.00	1.54 ± 0.06	1.24 ± 0.04	1.00 ± 0.00	0.75 ± 0.00	1.01 ± 0.01
liver-disorders	*(r)*	5.85 ± 0.02	21.53 ± 0.07	29.10 ± 0.46	12.20 ± 0.93	7.50 ± 0.65	**4.30 ± 0.21**	10.39 ± 0.07
	(t)	1.51 ± 0.00	3.16 ± 0.02	2.66 ± 0.04	2.69 ± 0.12	2.66 ± 0.17	1.94 ± 0.12	1.13 ± 0.07
parkinsons	*(r)*	5.01 ± 0.03	4.79 ± 0.03	8.60 ± 0.22	8.80 ± 0.42	6.40 ± 0.34	**3.80 ± 0.20**	7.11 ± 0.07
	(t)	1.83 ± 0.02	2.18 ± 0.02	1.90 ± 0.03	2.13 ± 0.06	1.75 ± 0.07	1.62 ± 0.12	0.98 ± 0.01
pima	*(r)*	8.43 ± 0.06	33.55 ± 0.13	49.50 ± 0.48	12.50 ± 0.92	7.50 ± 0.34	**3.50 ± 0.22**	14.89 ± 0.09
	(t)	1.55 ± 0.01	3.84 ± 0.02	2.76 ± 0.02	2.86 ± 0.13	1.97 ± 0.08	1.90 ± 0.08	1.07 ± 0.01
wine	*(r)*	4.33 ± 0.01	**3.08 ± 0.02**	4.90 ± 0.18	5.40 ± 0.22	4.30 ± 0.21	3.90 ± 0.23	5.44 ± 0.07
	(t)	1.67 ± 0.01	1.96 ± 0.03	1.83 ± 0.07	1.64 ± 0.04	1.19 ± 0.05	1.38 ± 0.03	0.87 ± 0.01

in one data set each.

In order to evaluate simplicity we focus here on the analysis of the results in Table III, about the total size of the classification model (total number of terms in all rules), since it is simpler to analyse this single measure of simplicity than to analyse the two measures of simplicity in Table IV (number of rules and average number of terms per rule) simultaneously. In terms of size of the classification model, JRip is the algorithm that discovers the lowest number of terms, with an average rank of 1.67 across all data sets; cAnt-Miner2$_{MDL}$ is second, with an average rank of 2.50; cAnt-Miner$_{PB}$ is third, with an average rank of 2.67; PSO/ACO2 and PART have similar results, with an average rank of 4.61 and 4.72, respectively; C4.5rules is sixth, with an average rank of 5.22; CN2 is the algorithm that consistently discovered the greatest number of terms and performs last, with an average rank of 6.61. There are no statistically significant differences between the average ranks of the top three algorithms, namely JRip, cAnt-Miner2$_{MDL}$ and cAnt-Miner$_{PB}$. The average rank of JRip is statistically significantly better than the average ranks of PSO/ACO2, PART, C4.5rules and CN2. JRip is the algorithm that discovered the simplest list in nine of the eighteen data sets, followed by cAnt-Miner$_{PB}$ in five data sets, cAnt-Miner2$_{MDL}$ in three data sets and PSO/ACO2 in one data set; both PART and CN2 have consistently discovered lists with a greater number of terms than the other algorithms.

We have also performed a comparison with respect to the total number of discovered rules (presented in Table IV), as it can be considered a measure related to the end-users' comprehensibility of the classification model [41]. JRip is the algorithm that discovers the lowest number of rules, with an average rank of 1.50 across all data sets; cAnt-Miner2$_{MDL}$ is second, with an average rank of 2.33; PSO/ACO2 is third, with an average rank of 3.67; cAnt-Miner$_{PB}$ is fourth, with an average rank of 4.28; PART is fifth, with an average rank of 4.67; C4.5rules is sixth, with an average rank of 5.06; CN2 is the algorithm that consistently discovered the greatest number of rules and performs last, with an average rank of 6.50. The average rank of JRip is statistically significantly better than the average ranks of PSO/ACO2, cAnt-Miner$_{PB}$, PART, C4.5rules and CN2. JRip is the algorithm that discovers the lowest number of rules, in twelve of the eighteen data sets, followed by PSO/ACO2 in four data sets and cAnt-Miner2$_{MDL}$ in two data sets; cAnt-Miner$_{PB}$, PART, C4.5rules and CN2 have consistently discovered lists with a greater number of rules than the other algorithms.

Overall, the results obtained by cAnt-Miner$_{PB}$ are very positive. It achieved the best average rank in terms of predictive accuracy and outperformed all the other algorithms with statistically significant differences. Hence, cAnt-Miner$_{PB}$ is the most accurate algorithm in our experiments. In terms of size of the classification model, it is amongst the ones with the lowest total number of terms in the discovered lists, achieving the third best average rank (out of seven algorithms) and there is no statistically significant difference between its average rank and the best average rank. Although cAnt-Miner$_{PB}$ is competitive considering the size of the classification model, measured as the total number of terms in the discovered list, it discovered lists with a greater number of rules on average that is statistically significant different than JRip. This difference highlights the characteristics of the discovered lists: while JRip discovered lists with a smaller number of longer rules (rules with a greater number of terms), cAnt-Miner$_{PB}$ discovered lists with a greater number of shorter rules (rules with a smaller number of terms).

Fig. 3 illustrates the average predictive accuracy rank versus the average model size rank (left graph in Fig. 3) and average predictive accuracy rank versus the average number of rules rank (right graph in Fig. 3) of the algorithms used in our experiments. In each graph, the Pareto front is indicated by a line connecting the non-dominated algorithms. According to the

TABLE V
STATISTICAL TEST RESULTS ACCORDING TO THE NON-PARAMETRIC FRIEDMAN TEST WITH THE HOLM'S POST-HOC TEST FOR $\alpha = 0.05$.

algorithm	average rank	p	*Holm*
(i) Predictive Accuracy			
cAnt-Miner$_{PB}$ (control)	2.28	–	–
C4.5rules	***4.00***	***0.017***	***0.0500***
cAnt-Miner2$_{MDL}$	***4.05***	***0.013***	***0.0250***
PSO/ACO2	***4.11***	***0.011***	***0.0167***
PART	***4.11***	***0.011***	***0.0125***
JRip	***4.67***	***9.08E-4***	***0.0100***
CN2	***4.78***	***5.17E-4***	***0.0083***
(ii) Model Size			
JRip (control)	1.67	–	–
cAnt-Miner2$_{MDL}$	2.50	0.247	0.0500
cAnt-Miner$_{PB}$	2.67	0.165	0.0250
PSO/ACO2	***4.61***	***4.33E-5***	***0.0167***
PART	***4.72***	***2.20E-5***	***0.0125***
C4.5rules	***5.22***	***7.90E-7***	***0.0100***
CN2	***6.61***	***6.6E-12***	***0.0083***
(iii) Number of Rules			
JRip (control)	1.50	–	–
cAnt-Miner2$_{MDL}$	2.33	0.247	0.0500
PSO/ACO2	***3.67***	***0.003***	***0.0250***
cAnt-Miner$_{PB}$	***4.28***	***1.14E-4***	***0.0167***
PART	***4.67***	***1.09E-5***	***0.0125***
C4.5rules	***5.06***	***7.90E-7***	***0.0100***
CN2	***6.50***	***3.8E-12***	***0.0083***

concept of Pareto dominance in multi-objective optimization, an algorithm A_1 dominates another algorithm A_2 if and only if the following two conditions are true: (1) A_1 is not worse than A_2 with respect to both objectives—i.e., both accuracy and model size (left graph) or number of rules (right graph)—and (2) A_1 is strictly better than A_2 according to at least one objective. An algorithm is included in the Pareto front if it is not dominated by any other algorithm. As can be seen in Fig. 3, in both the left and the right graphs, the Pareto front includes three non-dominated algorithms, namely JRip, cAnt-Miner2$_{MDL}$ and cAnt-Miner$_{PB}$.

A closer look at the results obtained by cAnt-Miner$_{PB}$ and cAnt-Miner2$_{MDL}$ is particularly interesting, since the main difference between them is the strategy used to build a list of rules. cAnt-Miner$_{PB}$ is consistently more accurate than cAnt-Miner2$_{MDL}$—with the exception in only 4 (out of 18) data sets, namely *breast-l*, *breast-tissue*, *credit-a* and *cylinder-bands*—highlighting the effectiveness of the new sequential covering strategy in improving predictive accuracy. The increase in accuracy is achieved by discovering lists of rules with a greater total number of terms than cAnt-Miner2$_{MDL}$ in 13 (out of 18) data sets—with large differences in the *credit-g*, *cylinder-bands* and *dermatology*—although there is no statistically significant difference between their average ranks. It is also observed that the lists of rules discovered by cAnt-Miner$_{PB}$ contain a greater number of shorter rules on average compared to cAnt-Miner2$_{MDL}$, which discovered lists of rules with a smaller number of longer rules.

In this paper we focused on classification models represented by a list of *IF-THEN* rules, which is an intuitively comprehensible type of representation for users in general, but of course that is not the only type of comprehensible representation for

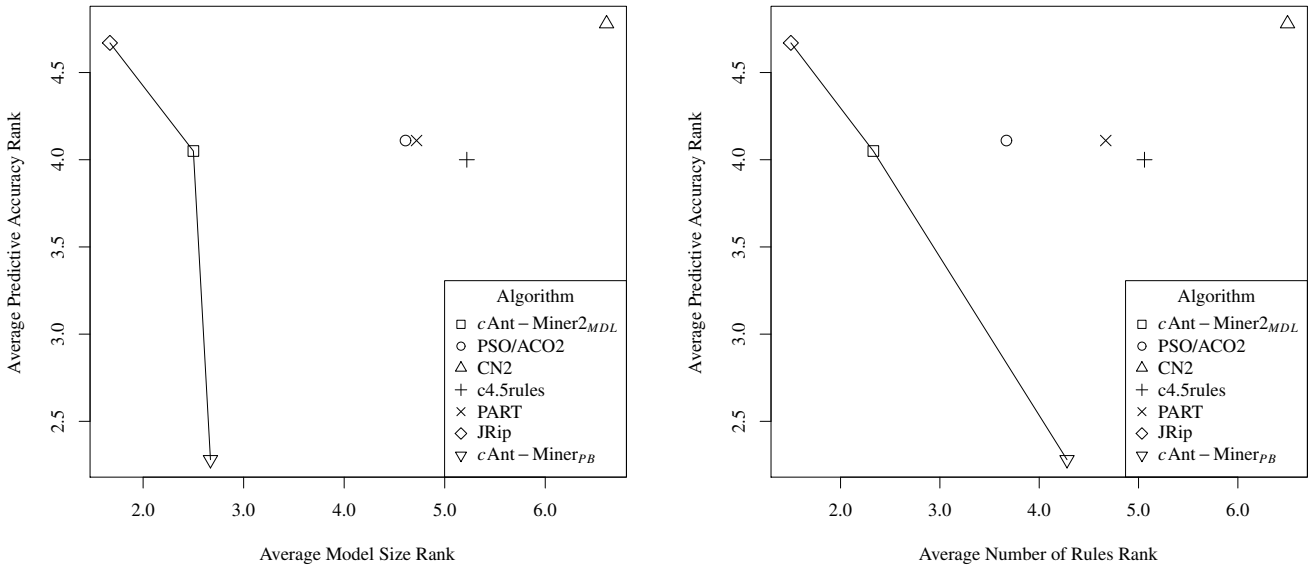

Fig. 3. Plot illustrating the average predictive accuracy rank versus the average model size rank (left graph) and average predictive accuracy rank versus the average number of rules rank (right graph) of the algorithms used in our experiments—the lower the average rank, the better the algorithm's performance. The Pareto front (shown as a connected line) is composed by JRip, cAnt-Miner2_{MDL} and cAnt-Miner$_{PB}$ algorithms—the algorithms that are not outperformed by any algorithm with respect to both predictive accuracy and model size or both predictive accuracy and number of rules.

classification models. Some users might prefer to use, for instance, logistic regression models, which tend to present good results in terms of accuracy in applications such as credit scoring [42] and also produce models that can be considered comprehensible, with the proper mathematical interpretation [43]. Another approach to discover comprehensible models consists of first using an SVM (support vector machines) algorithm—which tends to be a powerful method in terms of predictive accuracy—to discover a very accurate model and then to use a method that extracts rules from the SVM model in a post-processing step, in order to improve the comprehensibility of the model [44], [45]. In this case the extracted rules will tend to explain a large part of (but typically not all) the decisions of the original SVM model.

Although cAnt-Miner$_{PB}$ is competitive in terms of size of the classification model when compared to all other algorithms used in the experiments, the evaluation of a global pruning procedure—which takes advantage that an ant creates a complete list of rules and prunes each rule in the context of the whole list instead of a single rule at a time—could potentially improve the number of discovered rules by preferring more general rules (i.e., rules covering more training examples), reducing the total number of rules required to cover all training examples and, consequently, the total number of terms in the discovered lists. The implementation of such pruning procedure is an interesting direction for future research.

C. Time Complexity Analysis

In this section we discuss the computational time complexity of the proposed cAnt-Miner$_{PB}$ algorithm. The analysis is divided into two parts: 1) the inner *while-loop* of the algorithm presented in Fig. 2, which corresponds to the procedure of creating a complete candidate list of rules; 2) the outer *while-loop* of the algorithm presented in Fig. 2, which corresponds to an entire execution of cAnt-Miner$_{PB}$. Let e be the number of training examples, a the number of attributes representing

the choices available to create a rule[2], m the number of ants (colony size) and t the number of iterations, we can define the computational complexity of each of the aforementioned parts as:

1) *Computational time complexity of creating a list of rules*: A list of rules is created by an iterative sequential covering procedure that builds a single rule. In order to create a rule, the heuristic information for every vertex of the construction graph is computed. This step takes $\mathcal{O}(a)$. Then, an ant will choose k out of the a available terms. Note that k is a highly variable number, depending on the current number of training examples, but in the worse case k will be equal to a. When an ant selects a continuous attribute vertex, there is the need to select a threshold value, which consists of evaluating e different values. Hence, this step takes $\mathcal{O}(a^2 \cdot e)$. After a rule is created, a pruning procedure is applied to remove irrelevant terms. At each step of the pruning procedure a new rule is created by removing the last term of its antecedent and evaluated over the e examples, therefore the pruning involve evaluating at most a - 1 rules over e training examples, taking $\mathcal{O}(a \cdot e)$. The process of creating a rule is repeated e times in the worst case of the sequential covering, assuming that the algorithm creates a rule for each training example, therefore the upper-bound estimation for time complexity of the creation of a list of rules is $\mathcal{O}(a^2 \cdot e^2)$.
2) *Computational time complexity of an entire execution*: An entire execution of the algorithm involves t iterations of a procedure where a colony of m ants creates candidates list of rules and the best candidate list of rules is used to update the pheromone values. The pheromone update consists of increasing/decreasing e pheromone values of a vertices—i.e., the pheromone values of each vertex is increase/decrease for each rule in the candidate list of rules. Therefore, an entire execution takes $\mathcal{O}(t \cdot m \cdot a^2 \cdot e^2) + \mathcal{O}(t \cdot a \cdot e)$, where $\mathcal{O}(t \cdot m \cdot a^2 \cdot e^2)$ represents the upper-bound estimation for time complexity.

Note that the above analysis corresponds to the worst case estimation and it does not take into account that the number of training examples e decreases at each iteration of the sequential covering used to create a list of rules and that the algorithm is able to converge before reaching the maximum number of iterations t. It also uses two pessimistic assumptions. First, it considers that k is equal to a, where in the average case k tends to be much smaller than a. Second, it considers that the number of discovered rules by the sequential covering is equal to e, where in practice the number of discovered rules is much smaller than the number of examples. As a result, the quadratic complexity in terms of the number of examples ('e^2' in the above equation) is too pessimistic, and the time complexity tends to be in general approximately linear in the number of examples. Table IV provides evidence for both claims, showing that the average number of terms in a rule is much smaller than the number of attributes and that the number of discovered rules is much smaller that the number of examples.

D. Computational Time

Table VI presents the average computational time (in seconds)[3] to complete a fold of the cross-validation for cAnt-Miner2$_{MDL}$, PSO/ACO2 and cAnt-Miner$_{PB}$ algorithms over fifteen runs. The deterministic CN2, C4.5rules, PART and JRip algorithms take

[2]The number of attributes will correspond to the number of vertices of the construction graph when dealing with data sets containing only continuous attributes; in the case of nominal attributes, there will be a vertex for each value of a particular attribute. Assuming that the number of values per nominal attributes is relatively small, we can simplify the analysis by considering that the number of vertices is the same as the number of attributes.

[3]Measured on a 3.33GHz Intel Xeon-based computer with 8GB RAM.

TABLE VI
AVERAGE COMPUTATIONAL TIME IN SECONDS (*average* ± *standard error*) TAKEN BY *c*ANT-MINER2$_{MDL}$, PSO/ACO2 AND *c*ANT-MINER$_{PB}$ TO COMPLETE A FOLD OF THE CROSS-VALIDATION.

data set	*c*Ant-Miner2$_{MDL}$	PSO/ACO2	*c*Ant-Miner$_{PB}$
annealing	35.15 ± 0.81	17.16 ± 0.44	1236.60 ± 94.4
balance-scale	2.55 ± 0.08	21.65 ± 1.15	2.90 ± 0.13
breast-l	1.59 ± 0.05	3.21 ± 0.12	3.40 ± 0.27
breast-tissue	2.01 ± 0.06	1.18 ± 0.01	4.90 ± 0.21
breast-w	21.09 ± 0.76	8.85 ± 0.05	66.23 ± 3.01
credit-a	7.27 ± 0.38	15.87 ± 0.30	27.20 ± 1.12
credit-g	11.13 ± 0.27	51.24 ± 0.69	1330.50 ± 66.7
cylinder-bands	26.67 ± 1.16	7.02 ± 0.27	1153.90 ± 20.1
dermatology	35.13 ± 2.80	5.27 ± 0.05	57.70 ± 2.59
glass	5.59 ± 0.24	4.28 ± 0.03	16.51 ± 0.78
heart-c	6.15 ± 0.29	6.19 ± 0.03	17.32 ± 1.35
heart-h	7.49 ± 0.40	5.47 ± 0.03	24.33 ± 1.44
ionosphere	14.79 ± 0.45	4.33 ± 0.06	104.07 ± 4.87
iris	1.04 ± 0.03	1.00 ± 0.00	1.93 ± 0.03
liver-disorders	2.90 ± 0.21	6.25 ± 0.03	18.13 ± 0.74
parkinsons	4.93 ± 0.21	1.91 ± 0.01	8.42 ± 0.43
pima	11.29 ± 0.40	21.35 ± 0.08	75.40 ± 4.27
wine	2.26 ± 0.10	1.11 ± 0.01	1.42 ± 0.09

on average 1 second to complete a fold in any of the data sets used in the experiments, given that they employ heuristics to discover a list of rules without the need to evaluate multiple candidate solutions, and therefore are not present in Table VI.

In general, the computational time taken by *c*Ant-Miner$_{PB}$ is greater than the time taken by both *c*Ant-Miner2$_{MDL}$ and PSO/ACO2, with two exceptions: on *balance-scale*, it is faster than PSO/ACO2 and on *wine* it is faster than *c*Ant-Miner2$_{MDL}$. Recall that the search space of *c*Ant-Miner$_{PB}$ is more complex than the search space of both *c*Ant-Miner2$_{MDL}$ and PSO/ACO2, since *c*Ant-Miner$_{PB}$ is searching for the best list of rules instead of searching for the best rule. Therefore, an increase in computational time is expected. As can be seen in Table VI, there are three data sets where the time taken by *c*Ant-Miner$_{PB}$ is much greater than the others: *annealing*, *credit-g* and *cylinder-bands*. The common features of these data sets are the number of attributes (20 or more attributes) and the number of examples (more than 500 examples), which suggest that the performance of *c*Ant-Miner$_{PB}$ is sensible to the combination of the number of attributes—in particular continuous attributes—and the number of examples present in the data set.

It should be noted that in many data mining applications—e.g., medical diagnosis, bioinformatics and credit scoring—the computational time taken by the algorithm to induce a classification model has a minor importance, since they represent off-line applications and the time spent collecting and preparing the data is usually much greater than the running time of the classification algorithm. In addition, ACO algorithms can be easily parallelised since each ant builds and evaluates a candidate solution (a complete list of rules in *c*Ant-Miner$_{PB}$) independent from all the other ants. Therefore, a large speed up could be obtained by running a parallel version of *c*Ant-Miner$_{PB}$ on a computer cluster or another parallel computing system in applications where *c*Ant-Miner$_{PB}$'s processing time becomes a significant issue.

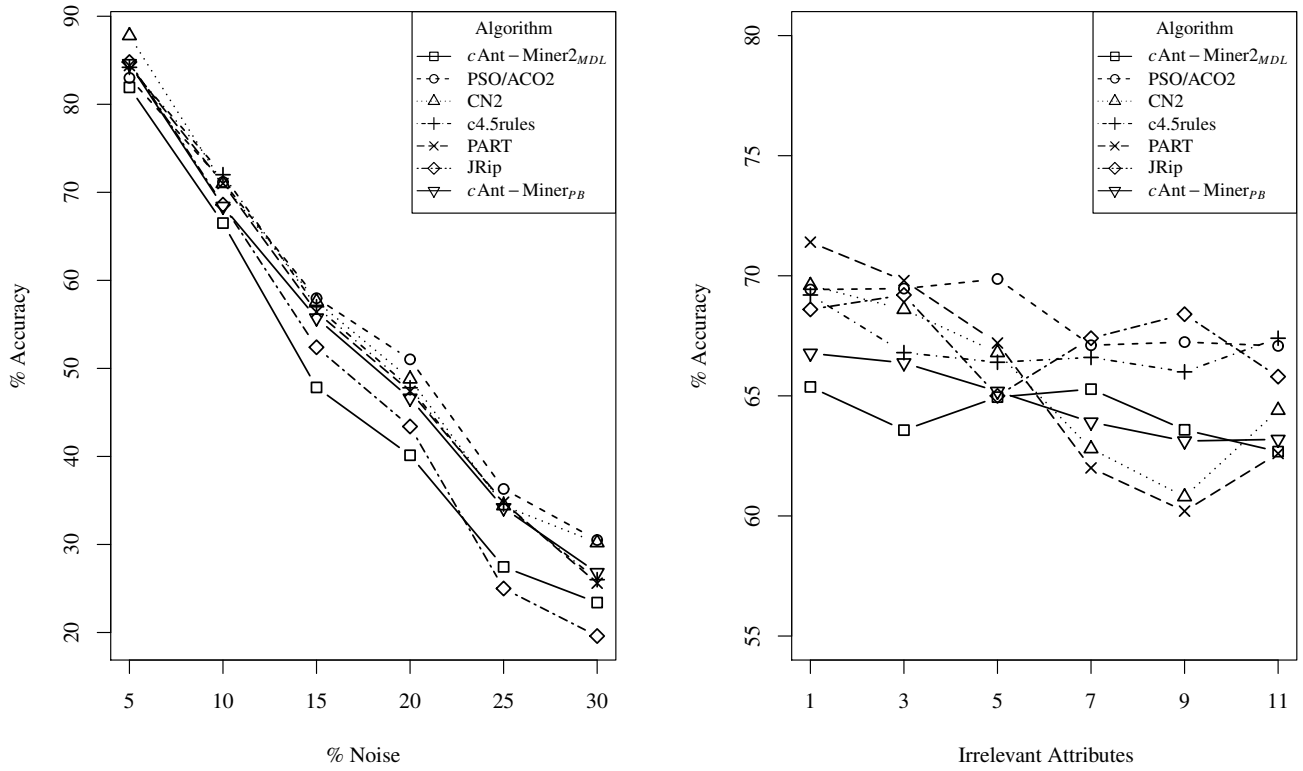

Fig. 4. Performance (measured in terms of predictive accuracy) of the algorithms used in our experiments when presented with irrelevant attributes and noisy data. The graph on the left corresponds to the observations in the data sets with increased noise probability. The graph on the right corresponds to the observations in the data sets with increased number of irrelevant attributes.

E. Artificial Data

In this section we study the behaviour of cAnt-Miner$_{PB}$ when presented with irrelevant attributes and noisy data. We have used an artificial data set generator that simulates the status of a seven Light-Emitting Diodes (LED) display when representing each of the ten decimal digits [46]. Using the artificial data set generator, we generated 12 data sets and divided the experiments into 2 groups:

- *noisy attributes*: 6 data sets with 500 examples were generated by increasing the noise probability of each of the original 7 attributes from 5% to 30% in steps of 5%. In this group of experiments there were no irrelevant attributes;
- *irrelevant attributes*: 6 data sets with 500 examples were generated by fixing the noise probability of each of the original 7 attributes to 10% and increasing the number of irrelevant attributes from 1 to 11 in steps of 2.

We have performed a ten-fold cross-validation and for the stochastic cAnt-Miner2$_{MDL}$, PSO/ACO2 and cAnt-Miner$_{PB}$ algorithms, they are run fifteen times using a different random seed to initialise the search for each partition of the cross-validation, while the deterministic CN2, C4.5rules, PART and JRip algorithms are run just once for each partition of the cross-validation. The results of the experiments with the artificial data sets are presented in Fig. 4, where it is illustrated the effect of noisy and irrelevant attributes in the predictive accuracy of the algorithms. Each result shown in Fig. 4 refers to the average predictive accuracy across the 6 artificial datasets used to investigate the effect of the level of noise (left graph in Fig.

4) or number of irrelevant attributes (right graph in Fig. 4).

When applied to the data sets with noisy attributes, the predictive accuracy of all algorithms decreased with the increase in the noise probability—JRip and cAnt-Miner2$_{MDL}$ are the algorithms more affected. The performance of the algorithms in the data sets with irrelevant attributes are mixed and small changes in the number of irrelevant attributes have either positive or negative effects. A closer look at the results of cAnt-Miner2$_{MDL}$ and cAnt-Miner$_{PB}$ suggests that the new sequential covering strategy is slightly more robust than cAnt-Miner2$_{MDL}$'s strategy, in particular in the presence of noisy attributes.

V. Conclusion And Future Research

In this paper we have discussed the sequential covering strategy commonly used by ant colony classification algorithms and its potential limitations. We have proposed a new strategy to discover a list of classification rules, which guides the search performed by the ACO algorithm using the quality of a candidate list of rules, instead of a single rule. The main motivation is to avoid the problem of rule interaction derived from the order in which the rules are discovered—i.e., the outcome of a rule affects the rules that can be discovered subsequently since the search space is modified due to the removal of examples covered by previous rules. In the new sequential covering strategy proposed, the pheromone matrix used by the ACO algorithm is extended to include a tour identification that indirectly encodes the sequence in which the rules should be created, allowing a more effective search for the best list of rules.

We have implemented the proposed sequential covering strategy in a new algorithm—named cAnt-Miner$_{PB}$—and conducted experiments involving eighteen publicly available data sets, comparing the results against state-of-the-art rule induction algorithms. We regard our results as very positive, given that cAnt-Miner$_{PB}$ is the most accurate algorithm, achieving statistically significantly higher predictive accuracy than all algorithms used in the comparison and discovering lists of rules with competitive size, measured as the total number of terms (conditions) in all rules. The direct comparison of cAnt-Miner$_{PB}$ against cAnt-Miner2$_{MDL}$ shows the advantage of the new sequential covering strategy in improving the predictive accuracy of the discovered list of rules.

There are several interesting directions for future research, which explore the fact that a candidate solution is represented by a complete list of rules. First, it would be interesting to evaluate a global pruning procedure, where a rule is pruned taking into account its effect on the whole list of rules, rather than pruning each rule individually—as discussed in subsection IV-B. Second, the use of a heuristic to reorder the list of rules—e.g., sorting the rules based on their confidence—in order to guide the search can potentially improve the convergence of the algorithm and the quality of the discovered list of rules.

In addition, the design of a Michigan-style ACO classification algorithm, incorporating niching methods to ensure diversity in the colony (i.e., ants representing rules that cover different training examples) and the use of meta-learning techniques [47] to investigate the links between data sets characteristics and cAnt-Miner$_{PB}$'s performance are interesting research directions worth further exploration.

APPENDIX

SOFTWARE AVAILABILITY

The documentation, source-code and binaries of the new cAnt-Miner$_{PB}$ algorithm are available for download at http://sourceforge.net/projects/myra.

REFERENCES

[1] G. Piatetsky-Shapiro and W. Frawley, *Knowledge Discovery in Databases*. AAAI Press, 1991, 525 pages.

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

[3] V. Dhar, D. Chou, and F. Provost, "Discovering interesting patterns for investment decision making with GLOWER – a genetic learner overlaid with entropy reduction," *Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 251–280, 2000.

[4] A. Freitas, D. Wieser, and R. Apweiler, "On the importance of comprehensible classification models for protein function prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 1, pp. 172–182, 2010.

[5] M. Dorigo and G. Di Caro, "The Ant Colony Optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds., 1999, pp. 11–32.

[6] M. Dorigo, G. Di Caro, and L. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[7] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004, 328 pages.

[8] R. Parpinelli, H. Lopes, and A. Freitas, "An ant colony based system for data mining: applications to medical data," in *Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001, pp. 791–798.

[9] ——, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321–332, 2002.

[10] D. Martens, B. Baesens, and T. Fawcett, "Editorial survey: swarm intelligence for data mining," *Machine Learning*, vol. 82, no. 1, pp. 1–42, 2011.

[11] B. Liu, H. Abbass, and B. McKay, "Density-based heuristic for rule discovery with ant-miner," in *6th Australasia-Japan joint workshop on intelligent and evolutionary systems (AJWIS-2002)*, 2002, pp. 180–184.

[12] ——, "Classification rule discovery with ant colony optimization," in *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT-2003)*, 2003, pp. 83–88.

[13] A. Chan and A. Freitas, "A new classification-rule pruning procedure for an ant colony algorithm," in *Artificial Evolution (Proceedings of the EA-2005)*, ser. Lecture Notes in Artificial Intelligence 3871, 2005, pp. 25–36.

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

[15] ——, "Handling continuous attributes in ant colony classification algorithms," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Data Mining (CIDM 2009)*. IEEE, 2009, pp. 225–231.

[16] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651–665, 2007.

[17] K. Salama and A. Abdelbar, "Extensions to the Ant-Miner Classification Rule Discovery Algorithm," in *Proceedings of the 7th International Conference on Swarm Intelligence (ANTS 2010), Lecture Notes in Computer Science 6234*, 2010, pp. 167–178.

[18] K. Salama, A. Abdelbar, and A. Freitas, "Multiple pheromone types and other extensions to the ant-miner classification rule discovery algorithm," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 149–182, 2011.

[19] A. González, R. Pérez, and J. Verdegay, "Learning the structure of a fuzzy rule: a genetic approach," *Fuzzy System and Artificial Intelligence*, vol. 3, pp. 57–70, 1994.

[20] G. Venturini, "SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts," *Machine Learning: ECML-93*, pp. 280–296, 1993.

[21] L. Booker, "Intelligent Behavior as an Adaptation to the Task Environment," Ph.D. dissertation, University of Michigan, 1982, 342 pages.

[22] L. Booker, D. Goldberg, and J. Holland, "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, pp. 235–282, 1989.

[23] S. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, University of Pittsburgh, 1980, 214 pages.

[24] ——, "Flexible learning of problem solving heuristics through adaptive search," in *Proceedings of the 8th International Conference on Artificial Intelligence*. Morgan Kaufmann, 1983, pp. 422–425.

[25] A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002, 264 pages.

[26] T. Stützle and H. Hoos, "Improvements on the Ant System: Introducing $\mathcal{MAX}$-$\mathcal{MIN}$ ant system," in *Proceedings of Artificial Neural Nets and Genetic Algorithms 1997*, 1998, pp. 245–249.

[27] ——, "$\mathcal{MAX}$-$\mathcal{MIN}$ ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[28] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2010, University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[29] U. Fayyad and K. Irani, "On the Handling of Continuous-Valued Attributes in Decision Tree Generation," *Machine Learning*, vol. 8, pp. 87–102, 1992.

[30] N. Holden and A. Freitas, "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*, 2008, 11 pages. [Online]. Available: http://dx.doi.org/10.1155/2008/316145

[31] P. Clark and T. Niblett, "The CN2 rule induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–283, 1989.

[32] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993, 302 pages.

[33] ——, "Improved Use of Continuous Attributes in C4.5," *Artificial Intelligence Research*, vol. 7, pp. 77–90, 1996.

[34] E. Frank and I. Witten, "Generating Accurate Rule Sets Without Global Optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, J. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 144–151.

[35] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005, 560 pages.

[36] W. Cohen, "Fast effective rule induction," in *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[37] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A Racing Algorithm for Configuring Metaheuristics," in *Genetic and Evolutionary Computation Conference (GECCO-2002)*, 2002, pp. 11–18.

[38] W. Conover, *Practical Nonparametric Statistics*. John Wiley & Sons, 1999, 592 pages.

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

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

[41] 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*, vol. 51, no. 1, pp. 141–154, 2011.

[42] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," *The Journal of the Operational Research Society*, vol. 54, no. 6, pp. 627–635, 2003.

[43] D. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd ed. John Wiley & Sons, 2000, 392 pages.

[44] G. Fung, S. Sandilya, and R. Bharat Rao, "Rule extraction from linear support vector machines," in *Proc. of the th11 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-05)*, 2005, pp. 32–40.

[45] H. Núñez, C. Angulo, and A. Català, "Rule extraction from support vector machines," in *Proc. of the European Symposium on Artificial Neural Networks (ESANN-02)*, 2002, pp. 107–112.

[46] J. Ranilla, O. Luaces, and A. Bahamonde, "A heuristic for learning decision trees and pruning them into classification rules," *AI Communications*, vol. 16, no. 2, pp. 71–87, 2003.

[47] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*. Springer, 2009, 176 pages.