Swarm Intell (2011) 5:149–182
DOI 10.1007/s11721-011-0057-9

Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm

Khalid M. Salama · Ashraf M. Abdelbar · Alex A. Freitas

Received: 22 October 2010 / Accepted: 16 May 2011 / Published online: 11 June 2011

Abstract Ant-Miner is an ant-based algorithm for the discovery of classification rules. This paper proposes five extensions to Ant-Miner: (1) we utilize multiple types of pheromone, one for each permitted rule class, i.e. an ant first selects the rule class and then deposits the corresponding type of pheromone; (2) we use a quality contrast intensifier to magnify the reward of high-quality rules and to penalize low-quality rules in terms of pheromone update; (3) we allow the use of a logical negation operator in the antecedents of constructed rules; (4) we incorporate stubborn ants, an ACO variation in which an ant is allowed to take into consideration its own personal past history; (5) we use an ant colony behavior in which each ant is allowed to have its own values of the α and β parameters (in a sense, to have its own personality). Empirical results on 23 datasets show improvements in the algorithm's performance in terms of predictive accuracy and simplicity of the generated rule set.

Keywords Ant Colony Optimization (ACO) · Data mining · Classification · Multipheromone · Stubborn Ants

1 Introduction

Data mining is an active research area involving the development and analysis of algorithms for extracting interesting knowledge (or patterns) from real-world datasets. In this paper we focus on the classification task of data mining, where the goal is to discover, from labeled cases, a model that can be used to predict the class of unlabeled cases (Jaiwei and Kamber

Electronic supplementary material The online version of this article (doi:10.1007/s11721-011-0057-9) contains supplementary material, which is available to authorized users.

K.M. Salama (✉) · A.M. Abdelbar
Department of Computer Science & Engineering, American University in Cairo, Cairo, Egypt
e-mail: abdelbar@aucegypt.edu

A.A. Freitas
School of Computing, University of Kent, Canterbury, UK

2006). Ant-Miner is an Ant Colony Optimization (ACO) (Dorigo and Stützle 2004, 2010) algorithm, proposed by Parpinelli et al. (2002), which discovers classification rules of the form

$$\textbf{\textit{IF}}\ \langle \textit{Term-1} \rangle\ \textbf{\textit{AND}}\ \langle \textit{Term-2} \rangle\ \textbf{\textit{AND}} \ldots \langle \textit{Term-n} \rangle\ \textbf{\textit{THEN}} \langle \textit{Class} \rangle$$

where each term is of the form $\langle attribute = value \rangle$, and the consequent of a rule is the predicted class. In this paper, we propose a number of extensions to the Ant-Miner algorithm and then empirically evaluate them using 23 widely-used datasets.

In Sect. 2, we present a brief description of the original Ant-Miner algorithm, followed by a brief review of related work in Sect. 3. We then present each of our proposed extensions in Sects. 4 through 8. In Sect. 4 we introduce the multipheromone ant system, in which a different pheromone type is associated with each permitted class; an ant selects the rule class first and then deposits pheromone of the type associated with the selected class. Section 5 presents our novel quality contrast intensifier. Section 6 describes the use of a logical negation operator in the construction of rule antecedents. Section 7 proposes the use of stubborn ants, an ACO variation in which an ant is influenced by its own rule construction history. Section 8 describes the use of ants with personality, an ACO extension in which each ant has its own values of the α and β parameters. Finally, Sects. 9 and 10 discuss our experimental methodology and results, respectively, and some final remarks are presented in Sect. 11.

This paper builds on our earlier work (Salama and Abdelbar 2010) in which four of the five proposed extensions were introduced. More precisely, this paper extends the work reported in Salama and Abdelbar (2010) in three ways. Firstly, the quality contrast intensifier is introduced. Secondly, the number of datasets used in the experimental evaluation is increased from 4 to 23. Thirdly, we report results for variations of the proposed algorithm, involving 12 different combinations of Ant-Miner extensions.

2 Ant-Miner algorithm

The proposed modifications presented in this paper are based on the original Ant-Miner algorithm introduced in Parpinelli et al. (2002). Algorithm 1 presents a high-level pseudocode description of the Ant-Miner Algorithm. For a more detailed discussion, the reader is referred to Parpinelli et al. (2002), and to Dorigo and Stützle (2004, Sect. 5.6.1).

Ant-Miner discovers an ordered list of IF-THEN classification rules (whose form was described in the Introduction) and is applicable only to datasets with categorical attributes. Datasets with real-valued attributes need a pre-processing step to discretize these attributes into categorical intervals before applying the algorithm. An important characteristic of ACO algorithms is the construction graph used to represent the trails followed by the artificial ants. In the case of Ant-Miner, the nodes of the construction graph correspond to the terms (attribute-value pairs) available in the dataset being mined. Hence, a trail in Ant-Miner consists of following a sequence of vertices in the construction graph, adding one term at a time to a rule antecedent.

The algorithm consists of two nested loops: the outer loop (*while* loop), where a single rule in each iteration is added to the discovered rule list; and the inner loop (*repeat–until* loop) where an ant in each iteration constructs a rule as follows. Each ant first constructs a rule's antecedent by selecting terms probabilistically according to the pheromone amount for that term and a heuristic function involving information gain (Quinlan 1993), until all the attributes have been used (an attribute can be used only once in a rule antecedent), or

Algorithm 1 Pseudo-code of Ant-Miner.

```
Begin Ant-Miner
training_set ← all training cases;
discovered_rule_set ← φ;
InitializePheromoneAmounts();
while |training_set| > max_uncovered_cases do
    CalculateHeuristicValues();
    R_best ← φ;
    i ← 0;
    repeat
        ConstructRuleAntecedent(ant_i);
        ComputeRuleClass(ant_i);
        R_current ← PruneRule(ant_i);
        Q_current ← CalculateRuleQuality(R_current);
        UpdatePheromone(R_current);
        if Q_current > Q_best then
            R_best ← R_current;
        end if
        i ← i + 1;
    until i = max_trials OR Convergence()
    discovered_rule_set ← discovered_rule_set + R_best;
    training_set ← training_set − Cases(R_best);
end while
End
```

until adding any other term to the rule antecedent would make the rule coverage less than `min_cases_per_rule`. Then, the rule consequent is chosen by a deterministic procedure, which chooses the class value with maximum occurrence in the set of cases matching the rule antecedent.

Next, a rule pruning procedure is carried out on the rule antecedent, in order to increase the rule's accuracy and/or improve its simplicity (reduce its size). Then, the ant updates the pheromone level by depositing pheromone on the terms contained in the just-constructed rule antecedent in proportion to the quality of the rule. This is done in order to increase the probability that the following ants will select the terms involved in the rule.

When the execution of the inner loop finishes, the best rule constructed in that loop is added to the list of discovered rules, and the training cases matched by that rule are removed from the training set (since those cases do not need to be covered by the next rules to be discovered). This set of steps is considered an iteration of the outer loop and is repeated until the number of training cases remaining in the training set becomes less than or equal to the value determined by the `max_uncovered_cases` parameter, or until the same rule is generated for a number of consecutive trials specified by the `no_rules_converg` parameter. The values of `min_cases_per_rule`, `max_uncovered_cases`, and `no_rules_converg` are user-specified thresholds.

2.1 Pheromone initialization and update

At the beginning of each outer loop, the pheromone is initialized for each term with the same value given by the function

$$\tau_{ij}(t=0) = \frac{1}{\sum_{r=1}^{a} b_r} \tag{1}$$

where a is the total number of attributes, i is the index of an attribute, j is the index of a value in the domain of attribute i, and b_r is the number of values in the domain of attribute r.

After an ant constructs a rule, the rule quality is evaluated and the pheromone amount is increased for the terms belonging to the rule according to its quality. This is calculated as follows:

$$Q = \underbrace{\frac{TP}{TP+FN}}_{\text{sensitivity}} \times \underbrace{\frac{TN}{TN+FP}}_{\text{specificity}} \tag{2}$$

where TP (true positives) is the number of cases covered by the rule and labeled by the class predicted by the rule, FP (false positives) is the number of cases covered by the rule and labeled by a class different from the class predicted by the rule, FN (false negatives) is the number of cases that are not covered by the rule but are labeled by the class predicted by the rule, and TN (true negatives) is the number of cases that are not covered by the rule and are not labeled by the class predicted by the rule.

The formula governing the increase in pheromone amount (according to rule quality) is:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q \tag{3}$$

where Q is the constructed rule's quality, computed using (2). The pheromone values for all terms are normalized to simulate evaporation, so that the pheromone levels are increased in the nodes selected in the constructed rule and decreased in the rest of the nodes in the construction graph. Normalization takes place by rescaling the entries of τ so that the following condition is true:

$$\sum_{r=1}^{a} \sum_{s=1}^{b_r} \tau_{rs} = 1 \tag{4}$$

2.2 Term selection

The term is selected probabilistically according to two components, as shown in the following formula:

$$P_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} [\tau_{rs}(t)]^{\alpha} \cdot [\eta_{rs}]^{\beta}} \tag{5}$$

The terms α and β in (5) are typically assigned to 1 in Ant-Miner; therefore, (5) can be written more simply as

$$P_{ij} = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} \tau_{rs}(t) \cdot \eta_{rs}} \tag{6}$$

Table 1 An example of eight cases with two attributes and a class

Case	Condition	Safety	Class
1	Excellent	Bad	Buy
2	Very good	Very good	Buy
3	Good	Good	Buy
4	Good	Very good	Buy
5	Bad	Very good	Wait
6	Bad	Very good	Wait
7	Bad	Good	Do not buy
8	Bad	Bad	Do not buy

In this equation, P_{ij} is the probability of selecting the term $\langle attribute_i = value_j \rangle$ (referred to as $term_{ij}$), and η_{ij} is a problem-dependent heuristic function that involves information gain (Quinlan 1993) and is computed as follows:

$$\eta_{ij} = \frac{\log_2(m) - entropy(T_{ij})}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (\log_2(k) - entropy(T_{rs}))} \tag{7}$$

where the measure of entropy associated with $term_{ij}$ is calculated as follows:

$$entropy(T_{ij}) = -\sum_{w=1}^{m} \left(\frac{freq(T_{ij}^w)}{|T_{ij}|} \right) \cdot \log_2 \left(\frac{freq(T_{ij}^w)}{|T_{ij}|} \right) \tag{8}$$

where m is the number of classes, T_{ij} is the subset of cases in which attribute i is equal to value j, $|T_{ij}|$ is the number of cases in the subset T_{ij}, and $freq(T_{ij}^w)$ is the number of cases in subset T_{ij} labeled with class w. The second component in the term selection formula is τ_{ij}, which is the amount of pheromone on $term_{ij}$. The higher the value of η_{ij}, the better for classification the $term_{ij}$ is, thus leading to a higher probability of being selected. The same applies for pheromone amount τ_{ij}.

Example Suppose that we have the set of 8 cases shown in Table 1 taken from a dataset that has two categorical attributes and a class attribute. The `Condition` attribute has four possible values, and `Safety` has three values. Thus, the construction graph will contain seven attribute-value nodes: four nodes for the `Condition` attribute and three nodes for the `Safety` attribute. Specifically, τ would contain seven entries: τ[`Condition`, `Excellent`], τ[`Condition`, `Very Good`], τ[`Condition`, `Good`], τ[`Condition`, `Bad`], τ[`Safety`, `Very Good`], τ[`Safety`, `Good`], and τ[`Safety`, `Bad`]. From (1), these seven entries would be initialized to 1/7. Applying (7), we find that the η heuristic values are:

η[`Condition`, `Excellent`] = 0.22
η[`Condition`, `Very Good`] = 0.22
η[`Condition`, `Good`] = 0.22
η[`Condition`, `Bad`] = 0.08
η[`Safety`, `Very Good`] = 0.08
η[`Safety`, `Good`] = 0.08
η[`Safety`, `Bad`] = 0.08

Suppose that after some number of trials, the τ values are as follows:

$$\tau\,[\texttt{Condition}, \texttt{Excellent}] = 0.05 \qquad \tau\,[\texttt{Safety}, \texttt{Very Good}] = 0.18$$
$$\tau\,[\texttt{Condition}, \texttt{Very Good}] = 0.24 \qquad \tau\,[\texttt{Safety}, \texttt{Good}] = 0.16$$
$$\tau\,[\texttt{Condition}, \texttt{Good}] = 0.06 \qquad \tau\,[\texttt{Safety}, \texttt{Bad}] = 0.23$$
$$\tau\,[\texttt{Condition}, \texttt{Bad}] = 0.08$$

An ant constructing a rule in trial t would then make its decisions based on the following probabilities (6):

$$P\,[\texttt{Condition}, \texttt{Excellent}] = 0.09 \qquad P\,[\texttt{Safety}, \texttt{Very Good}] = 0.11$$
$$P\,[\texttt{Condition}, \texttt{Very Good}] = 0.41 \qquad P\,[\texttt{Safety}, \texttt{Good}] = 0.10$$
$$P\,[\texttt{Condition}, \texttt{Good}] = 0.10 \qquad P\,[\texttt{Safety}, \texttt{Bad}] = 0.14$$
$$P\,[\texttt{Condition}, \texttt{Bad}] = 0.05$$

Suppose that the ant chooses as its first term the highest-probability term, which is $\langle \texttt{Condition} = \texttt{Very Good} \rangle$. The ant then chooses its next term from the three possible values of the `Safety` attribute. Suppose that the ant chooses $\langle \texttt{Safety} = \texttt{Very Good} \rangle$, which has the second-highest probability of the three remaining available attribute-value pairs. The ant then selects the most frequently occurring class for this combination of attribute-values, which in this simple dataset is the class `Buy`. The constructed rule is thus:

IF $\langle \texttt{Condition} = \texttt{Very Good} \rangle$ **AND** $\langle \texttt{Safety} = \texttt{Very Good} \rangle$ **THEN** $\langle \texttt{Buy} \rangle$

For this rule: $TP = 1$, $FP = 0$, $FN = 3$, and $TN = 4$; therefore, the sensitivity and specificity are $1/4$ and $4/4$, respectively, and the rule quality (14) is 0.25.

The two entries $\tau[\texttt{Condition}, \texttt{Very Good}]$ and $\tau[\texttt{Safety}, \texttt{Very Good}]$ are then increased to become 0.30 and 0.23, respectively. τ is then normalized to simulate evaporation, and the final state of τ at the end of the trial is:

$$\tau\,[\texttt{Condition}, \texttt{Excellent}] = 0.05 \qquad \tau\,[\texttt{Safety}, \texttt{Very Good}] = 0.20$$
$$\tau\,[\texttt{Condition}, \texttt{Very Good}] = 0.27 \qquad \tau\,[\texttt{Safety}, \texttt{Good}] = 0.14$$
$$\tau\,[\texttt{Condition}, \texttt{Good}] = 0.05 \qquad \tau\,[\texttt{Safety}, \texttt{Bad}] = 0.21$$
$$\tau\,[\texttt{Condition}, \texttt{Bad}] = 0.07$$

As discussed in Parpinelli et al. (2002), there are several elements of similarity between the Ant-Miner algorithm and decision tree algorithms such as C4.5. The entropy-based heuristic function η used by Ant-Miner is the same kind of heuristic function used in decision tree algorithms. The main difference is that in the case of decision trees, weighted entropy is computed for each attribute on the data partition resulting from the different attribute values (or the choice of a threshold for a quantitative attribute), while, in the case of Ant-Miner, entropy is computed for an attribute-value pair only, since an attribute-value pair is chosen to expand a rule. In conventional tree algorithms, entropy is typically the only heuristic used during tree building. In contrast, in Ant-Miner, entropy is used in conjunction with pheromone information, which makes the rule-construction process of Ant-Miner more robust, since the feedback provided by pheromone updating tends to offset the shortsightedness of the entropy measure.

Note that the entropy measure considers one attribute at a time and therefore is sensitive to potential attribute interaction issues. On the other hand, pheromone updating tends to deal better with attribute interactions, because it is directly based on the performance of a rule as

a whole and thus directly takes into account interactions among all attributes occurring in the rule. The search strategy of Ant-Miner's rule pruning is also very similar to the pruning procedure suggested by Quinlan (1993), although the rule quality evaluation functions used in the two procedures are very different from one another.

3 Related work

3.1 Related work on ACO-based classification rule discovery

Chan and Freitas (2005) have proposed a new rule pruning procedure for Ant-Miner that led to the discovery of simpler (shorter) rules and improved the computational time in datasets with a large number of attributes, although in some datasets this led to a smaller predictive accuracy. Liu et al. presented two extensions, AntMiner2 (Liu et al. 2002) and AntMiner3 (Liu et al. 2003). AntMiner2 employs a density-based heuristic function for calculating the heuristic value for a term, while AntMiner3 is based on a new state transition approach. A pseudorandom proportional transition rule was used by Wang and Feng (2004).

Smaldon and Freitas (2006) introduced the idea of selecting the rule consequent class before rule construction—this idea is the inspiration for our multipheromone ant system modification described in Sect. 4—and producing an unordered rule set. Their approach was based on constructing rules for each class separately: an extra For-Each (class value) loop is added as an outer loop for the original algorithm. The consequent of the rule is known by the ant during rule construction and does not change. An ant tries to choose terms that improve the accuracy for a rule predicting the class value in the current iteration of the For-Each loop. This approach generates better rules in comparison with the original Ant-Miner, where a term is chosen for a rule in order to decrease entropy in the class distribution of cases matching the rule under construction. However, the entire execution (with the complete training set) is repeated separately for each class value until the number of positive cases (belonging to the current class) remaining in the dataset that have not been covered by the discovered rules is less than or equal to `max_uncovered_ cases`. For a more detailed description of the algorithm, refer to Smaldon and Freitas (2006).

Martens et al. (2007) have introduced a new ACO-based classification algorithm, named AntMiner+, which employs different pheromone initialization and update procedures based on the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) (Stützle and Hoos 2000). It makes a distinction between nominal and ordinal attributes. Instead of creating a pair ⟨attribute = value⟩ for each value of an ordinal attribute, AntMiner+ creates two types of bounds that represent the intervals of values to be chosen by the ants. Edges in the construction graph are considered the decision components, and, in addition, the α and β parameters are included as nodes in the construction graph, so that their values are selected, and adapted automatically during the algorithm's run, not statically set before execution. Moreover, AntMiner+ includes special handling of discrete attributes having ordered values (as opposed to nominal attributes having unordered attributes such as "male" and "female"), allowing for interval rules to be constructed. In addition, an extra vertex group is added at the start of the construction graph containing class values to allow the selection of class first. This is similar to considering the class as another variable. Rules with different classes can be constructed in the same iteration. Different heuristic values are applied according to the selected class in order to choose the terms that are relevant to the prediction of the selected class. However, pheromone information is shared by all ants constructing rules with different consequents.

Galea and Shen (2006) presented an ACO approach for the induction of fuzzy rules, named FRANTIC-SRL, which runs several ACO algorithm instances in parallel, each one

generating rules for a particular class. Swaminathan (2006) proposed an extension to Ant-Miner which enables interval conditions in the rules. For each discrete interval, a node is added to the construction graph, and the pheromone value associated to the node is calculated using a mixed kernel probability density function (PDF).

Otero et al. (2008) introduced a version of Ant-Miner that copes with continuous attributes named *c*Ant-Miner, by having the ability to create discrete intervals for continuous attributes "on-the-fly." Later, an extended version of *c*Ant-Miner was introduced in Otero et al. (2009).

The reader is referred to Martens et al. (2011) for a recent survey of swarm intelligence approaches to data mining.

3.2 Other related work

The extensions described in Sects. 7 and 8 of this paper are based on the ideas of stubborn ants and ants with personality, respectively, which were proposed by Abdelbar (2008). These ideas are motivated by the following argument. In most ACO methods, each ant generates its solution in a given iteration according to (5). In other words, each ant stochastically generates its solution, in a given iteration, based on the same pheromone τ and heuristic information η as every other ant. In a given iteration, the probability that a given candidate solution will be generated by a given ant k is identical to the probability that it will be generated by any other given ant ℓ.

Stubborn ants is an ACO variation, proposed in Abdelbar (2008) in the context of $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System applied to the traveling salesman problem, in which if an ant k generated a particular candidate solution $\mathcal{S}_{t-1}^{(k)}$ in iteration $t-1$, then the solution components of $\mathcal{S}_{t-1}^{(k)}$ will have a higher probability of being selected in the candidate solution $\mathcal{S}_{t}^{(k)}$ generated by ant k in iteration t. The idea is to increase diversity by making the probability distribution used to generate candidate solutions different from one ant to another based on each ant's past experience.

A further variation called ants with personality proposed in the future work section of Abdelbar (2008) takes the idea of promoting diversity further by allowing each ant to have its own values of the α_k and β_k parameters (where α_k and β_k represent the values used by ant k in applying (5)). In this way, some ants will give more importance to pheromone information, while others will give more importance to heuristic information. The α_k and β_k parameters for each ant are randomly generated from a normal distribution centered at some global α_{global} and β_{global}.

4 Utilizing multiple pheromone types

In the original Ant-Miner, the consequent of a rule is chosen after its antecedent's terms are selected by determining the class value with maximum occurrence in the cases matching the rule antecedent. The main principle of the multipheromone system is that the class is chosen before the rule antecedent's construction, so that the antecedent's terms are selected with respect to the current selected class. As discussed in Sect. 3.1, the idea of selecting the rule consequent prior to rule construction has been introduced in the literature in different flavors by Smaldon and Freitas (2006) and in AntMiner+ (Martens et al. 2007).

A major difference between our work and AntMiner+ is that in AntMiner+ every ant is influenced by the pheromone deposited by every other ant constructing similarly or differently labeled rules, as pheromone is shared by all ants. This can negatively affect the quality

Algorithm 2 Pseudo-code of Multipheromone Ant-Miner.

```
Begin Multipheromone Ant-Miner
training_set ← all training cases;
discovered_rule_set ← φ;
while |training_set| > max_uncovered_cases do
    InitializePheromoneAmounts();
    CalculateHeuristicValues();
    R_best ← φ;
    i ← 0;
    repeat
        SelectRuleClass(ant_i);
        ConstructRuleAntecedent(ant_i);
        R_current ← PruneRule(ant_i);
        Q_current ← CalculateRuleQuality(R_current);
        UpdatePheromone(R_current);
        if Q_current > Q_best then
            R_best ← R_current;
        end if
        i ← i + 1;
    until i = max_trials OR Convergence()
    discovered_rule_set ← discovered_rule_set + R_best;
    training_set ← training_set − Cases(R_best);
end while
End
```

of the constructed rules, as the terms that lead to constructing a good rule with class C_x as a consequent do not necessarily lead to constructing a good rule with C_y as a consequent.

Our work is different from that of Smaldon and Freitas (2006) in that, in their approach, the entire execution (with the complete training set) is repeated separately for each class value until the number of positive cases (belonging to the current class) remaining in the dataset that have not been covered by the discovered rules is less than or equal to `max_uncovered_cases`.

In contrast, our proposed multipheromone Ant-Miner system executes the course of operations only once during the entire training process. Even though we use a class-based strategy in term selection and pheromone update, ants can construct rules with different consequent classes in the same iteration simultaneously. However, an ant is only influenced by the ants that have constructed rules with the same consequent.

First, an ant probabilistically selects the rule consequent prior to the antecedent based on pheromone and heuristic information as described below. Then, it tries to choose terms that are relevant to predicting this class. The rule is then evaluated, and the pheromone is updated. But, unlike the version of Ant-Miner in Martens et al. (2007), the ant deposits different kinds of pheromone, as many as the number of permitted classes. The next ant is only influenced by the pheromone deposited for the class for which it is trying to construct a rule. In this case, pheromone information is not shared among ants constructing rules for different classes. This allows choosing terms that are only relevant to the selected class. A high-level pseudo-code description of multipheromone Ant-Miner is presented in Algorithm 2.

The idea of multipheromone Ant-Miner is that each class has a different pheromone type to be deposited on the terms in the construction graph. In essence, we are replacing the tra-

ditional two-dimensional pheromone structure (attribute, value) by a new three-dimensional pheromone structure (attribute, value, class). This also applies to the heuristic values structure.

During rule construction, the rule class is already set, and an ant is only influenced by the amount of pheromone in the pheromone array element corresponding to its rule class. Similarly in pheromone update, an ant deposits pheromone in the array element corresponding to the current rule class in each node belonging to the trail followed by the ant (i.e., for each term in the rule antecedent). Here, a term's pheromone value is a representation of that term's relevance for predicting a specific class (the one that is preselected for the rule consequent).

Class values are also represented as nodes in the construction graph, and pheromone can be deposited on them. The class value is selected probabilistically according to the pheromone amount and heuristic value associated with it, using (6). The heuristic value η_k is based on the empirical prior probability of class k and is calculated as follows:

$$\eta_k = \frac{freq(k)}{|TrainingSet|} \tag{9}$$

where $freq(k)$ is the frequency of class k in the current training set. In pheromone update, the pheromone level increases in the node of the constructed rule class according to the quality of the rule, similar to any other decision components (attribute values) in the construction graph.

We choose a class-based heuristic function that calculates the quality of an antecedent with respect to a specific class, which focuses on the term's relevance for predicting the preselected rule consequent. Laplace-corrected confidence (Smaldon and Freitas 2006) is used in the multipheromone system as a heuristic function and is given by

$$\eta_{ij,k} = \frac{|term_{ij},\ k| + 1}{|term_{ij}| + m} \tag{10}$$

where $\eta_{ij,k}$ is the heuristic value for $term_{ij}$ given that class k has been selected, $|term_{ij}, k|$ is the number of training cases which include $term_{ij}$ and the current selected class k, $|term_{ij}|$ is the number of training cases which include $term_{ij}$, and m is the number of classes. The probability of selecting $term_{ij}$ given that class k has been chosen is calculated as follows:

$$P_{ij,k} = \frac{\tau_{ij,k}(t) \cdot \eta_{ij,k}}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (\tau_{rs,k}(t) \cdot \eta_{rs,k})} \tag{11}$$

where $\tau_{ij,k}$ is the pheromone amount of type class k associated with $term_{ij}$. The amount of pheromone $\tau_{ij,k}$ is a representation of the quality of $term_{ij}$ in the prediction of class k.

Pheromone normalization (which is used to simulate evaporation) is applied separately for the partition of τ corresponding to each class. Specifically, for a given class k, normalization causes τ entries to be rescaled so that the following condition holds:

$$\sum_{r=1}^{a} \sum_{s=1}^{b_r} \tau_{rs,k} = 1 \tag{12}$$

In addition, for the class value elements themselves, normalization rescales the entries of τ corresponding to class values such that the following condition holds:

$$\sum_{h=1}^{m} \tau_h = 1 \tag{13}$$

where m is equal to the number of classes.

Example Consider the previously introduced sample dataset shown in Table 1. In the multipheromone system, τ would contain 21 elements, including elements such as $\tau[\texttt{Condition}, \texttt{Good}, \texttt{Buy}]$, $\tau[\texttt{Safety}, \texttt{Bad}, \texttt{Wait}]$, and $\tau[\texttt{Condition}, \texttt{Bad}, \texttt{Do not Buy}]$. In addition, τ would contain three elements corresponding to the possible class values: $\tau[\texttt{Buy}]$, $\tau[\texttt{Wait}]$, and $\tau[\texttt{Do not Buy}]$. η would similarly contain 21 elements for the attribute-value-class triples and three elements for the class values. If an ant selects class Buy, then the probability of selecting $(\texttt{Safety}, \texttt{Good})$ would depend only on $\tau[\texttt{Safety}, \texttt{Good}, \texttt{Buy}]$ and $\eta[\texttt{Safety}, \texttt{Good}, \texttt{Buy}]$. If the constructed rule includes the term $\langle \texttt{Safety} = \texttt{Good} \rangle$, then the value of $\tau[\texttt{Safety}, \texttt{Good}, \texttt{Buy}]$, as well as the value of $\tau[\texttt{Buy}]$, will each be increased by an amount that depends on the rule's quality, while $\tau[\texttt{Safety}, \texttt{Good}, \texttt{Wait}]$ and $\tau[\texttt{Safety}, \texttt{Good}, \texttt{Do not Buy}]$ will remain unchanged.

Rules are evaluated by a function that balances between the support and confidence of the rule as follows:

$$Q(R_t) = \underbrace{\frac{TP}{|TrainingSet|}}_{Support(R_t)} + \underbrace{\frac{TP}{|Matches|}}_{Confidence(R_t)} \tag{14}$$

where $Support(R_t)$ represents the ratio of the number of cases that match R_t's antecedent and are labeled by its class to the total number of cases in the training set, and $Confidence(R_t)$ represents the ratio of the number of cases that match rule R_t's antecedent and are labeled by its class to the total number of cases that match R_t's antecedent. Note that, for any rule R, it is always true that $0 \leq Q(R) \leq 2$.

We employ the quality evaluation function in (14), first used in AntMiner+ (Martens et al. 2007), instead of Ant-Miner's original evaluation function (2), since, in our multipheromone approach, the class is preselected, and a rule's antecedent is constructed based on the selected class. Hence, we evaluate the quality of the rule constructed with respect to the selected class, considering two aspects: predictive accuracy represented by the confidence of the rule antecedent given the selected class, and the rule coverage represented by the support of the rule. In contrast, in the original Ant-Miner, term selection is performed to reduce class entropy, regardless of the rule consequent. Thus the rule is evaluated by its sensitivity and specificity.

As for rule pruning, some alterations were made to take advantage of the preselection of the rule consequent class and the use of multiple types of pheromone. Rule pruning involves speculatively removing each term in turn and evaluating the quality of the rule without that term, then considering the rule with the removed term having the largest increase in rule quality. This process was repeated until no increase in rule quality was observed during the term removal process. A new consequent, the class with the highest occurrence among all cases covered by the rule, was assigned to the rule after each term was speculatively removed. In multipheromone Ant-Miner, the consequent remains the same during this process, and so the rule pruning procedure is simplified. After each term is removed, there is no need

to compute the quality of the new reduced rule for all possible classes of the consequent. This is because all the terms in the rule antecedent are selected based on the consequent class, so it is certain that the current class produces the highest quality with the current terms compared to other classes. Only the quality for the new reduced rule is computed.

Pheromone levels are increased on the terms included in the antecedent of the constructed rule R_t, with respect to the selected class k, according to the rule quality $Q(R_t)$. The pheromone amount is also increased for the selected class k of the constructed rule R_t. The following are the pheromone deposit formulas for the rule's terms and the rule's class, respectively:

$$\tau_{ij,k}(t+1) = \tau_{ij,k}(t) + \tau_{ij,k}(t) \cdot Q(R_t) \tag{15}$$

$$\tau_k(t+1) = \tau_k(t) + \tau_k(t) \cdot Q(R_t) \tag{16}$$

Pheromone normalization (to simulate evaporation) is then applied to the τ attribute-value pair entries corresponding to the class of the constructed rule, as well as to the τ entries corresponding to class values. Equation (12) is applied for the class k of the constructed rule, and (13) is applied to the τ entries corresponding to class values.

After the best rule of the current iteration is selected, all cases covered by this rule are removed from the training set, and the pheromone is initialized, but only in the pheromone array elements corresponding to the class of this rule. Leaving the pheromone in the array elements of other classes unchanged tends not to waste the knowledge that has been collected by the ants in the previous trials for the rest of the classes, leading to faster convergence in the next iterations.

5 Quality contrast intensifier

Along with the multipheromone system, we propose a new pheromone update procedure. The idea is to intensify the contrast between bad solutions, good solutions, exceptionally good solutions, and unvisited solutions during rule construction. Quality contrast intensification takes place as a new strategy for pheromone update. An ant that constructs a solution with good quality is rewarded by magnifying the amount of pheromone to be deposited on its trail. By contrast, an ant that constructs a bad rule is penalized by removing pheromone from its trail according to the weakness of the constructed solution. The quality contrast intensifier is applied—after rule quality evaluation—using the following conditional formula:

$$\Delta\tau(t)_k = \begin{cases} 2Q(R_t) & \text{if } \mathit{confidence}(R_t) \geq \phi_1 \\ Q(R_t) & \text{if } \phi_1 > \mathit{confidence}(R_t) \geq \phi_2 \\ Q(R_t) - 2 & \text{if } \phi_2 > \mathit{confidence}(R_t) \end{cases} \tag{17}$$

where $\Delta\tau_k$ is the amount of pheromone (type k) to be deposited by ant_t on each attribute value belonging to R_t's antecedent as well as class k, $Q(R_t)$ is the quality of R_t calculated using (14), and ϕ_1 and ϕ_2 are the upper and lower thresholds, respectively, for the rule confidence at which the quality is contrasted.

As we can see in (17), if the confidence of the constructed rule exceeds ϕ_1, the amount of the pheromone value to be added is doubled, as if there were two ants choosing the path that led to this high confidence solution. On the other hand, if the confidence of the constructed rule goes below ϕ_2, the pheromone value to be added is negative. This is obtained by subtracting 2 from the value of the rule quality (recall from (2) that the maximum value of rule

quality is 2). In our experiments, we use 0.8 and 0.5 for ϕ_1 and ϕ_2, respectively; this means that rules with a confidence of 80% or higher receive twice the reinforcement, while rules with confidence below 50% are penalized.

This contrast intensification strategy comes with several advantages. First, higher-quality rules get significantly more pheromone than other normal and lower quality rules, which leads to faster convergence. Second, it ensures the balance in the quality of output between the number of generated rules (which is affected by the rule support) and the classification accuracy of these rules (which is affected by the confidence of the rule). For example, some attribute values have a very high frequency of occurrence among the training set cases. This increases the support value in the quality evaluation, which increases the quality of the rule in general according to (14), regardless of the rule's confidence. Thus, this quality contrast intensifier works in favor of rule confidence in order not to generate significantly fewer rules with low classification quality. Finally, penalizing bad rules by removing pheromone from their trail gives an opportunity to unvisited nodes to be selected in further iterations, as their pheromone amount is likely to become higher than that of already-tried bad nodes. This can be expected to enhance the exploration aspect of the algorithm.

Note that the quality contrast intensifier described in this section is intended to be used in combination with the multipheromone extension. This is because, as can be seen in (17), the contrast intensifier rewards or penalizes solutions based on their confidence. Confidence is part of the rule quality evaluation function when multipheromone is used but is not part of the evaluation function in the original Ant-Miner, since in the former we apply a class-based term selection strategy, where the confidence measure applies, while in the latter, term selection is done only to decrease class distribution entropy.

6 Using logical negation operator in rule antecedents

In the original and various versions of Ant-Miner, the construction graph consists of nodes representing attribute values of the problem domain. The set of nodes (N) in the construction graph is

$$N = \bigcup_{i=1}^{a} v_{ij}, \quad j \in \{1, 2, \ldots, b_i\}$$

where i is the ith attribute, a is the number of attributes, b_i is the number of permitted values for attribute i, and v_{ij} is the jth permitted value of the ith attribute. Thus, the constructed rule antecedent will be of the form

$$\textbf{IF } \langle A_i = V_{ij} \rangle \textbf{ AND } \langle A_k = V_{kl} \rangle \textbf{ AND } \ldots$$

To allow using the logical negation operator in the antecedents of constructed rules, the values and their negation per attribute will be added to the construction graph. The set of nodes (N) in the construction graph will be

$$N = \bigcup_{i=1}^{a} v_{ij} \cup \bigcup_{i=1}^{a} \overline{v}_{ij}, \quad j \in \{1, 2, \ldots, b_i\}$$

Thus, the available decision components in the construction graph allow constructing rule antecedents of the form

$$\textbf{IF } \langle A_i = V_{ij} \rangle \textbf{ AND } \langle A_k \textbf{ NOT} = V_{kl} \rangle \textbf{ AND} \ldots$$

Negation nodes are added for an attribute if it has more than two values in its domain. Pheromone is updated normally on these terms, and a heuristic value is calculated for the negation attribute values in the same way as it is calculated for regular attribute values. An example of a generated rule using the logical negation operator is: "**IF** ⟨`Price` = `Low`⟩ **AND** ⟨`Condition` **NOT** = `Bad`⟩ **THEN** ⟨`Buy`⟩". In general, terms that have logical negation match more cases than the regular terms. This leads to the construction of rules with high coverage.

Example Consider the previously introduced sample dataset shown in Table 1. If the logical negation operator is used for constructing classification rules for this dataset, only three ordered rules will be needed to correctly classify the whole dataset. These rules are as follows:

1. **IF** ⟨`Condition` **NOT** = `Bad`⟩
   THEN⟨`Buy`⟩.
2. **ELSE IF** ⟨`Condition` = `Bad`⟩ **AND** ⟨`Safety` = `Very Good`⟩
   THEN ⟨`Wait`⟩.
3. **ELSE** ⟨`Do not Buy`⟩.

Because the rules generated with the logical negation operator have higher coverage, the output rule set size becomes smaller than the rule set generated without using logical negation. The following ordered list of rules is generated without using logical negation:

1. **IF** ⟨`Condition` = `Bad`⟩ **AND** ⟨`Safety` = `Very Good`⟩
   THEN ⟨`Wait`⟩.
2. **ELSE IF** ⟨`Condition` = `Bad`⟩ **AND** ⟨`Safety` = `Good`⟩
   THEN ⟨`Don't Buy`⟩.
3. **ELSE IF** ⟨`Condition` = `Bad`⟩ **AND** ⟨`Safety` = `Bad`⟩
   THEN ⟨`Do not Buy`⟩.
4. **ELSE** ⟨`Buy`⟩.

At least four rules are needed to correctly classify these cases without using the logical negation operator.

Note that our use of the logical negation operator does not differentiate between nominal and ordinal attributes. For example, given a nominal attribute "`Color`" with 3 values in its domain {`red`, `green`, `blue`}, the terms that can be generated from this attribute are ⟨`Color` **NOT** = `red`⟩, ⟨`Color` **NOT** = `green`⟩ and ⟨`Color` **NOT** = `blue`⟩. Similarly, given an ordinal attribute "`Blood Pressure`" with 3 values in its domain {`high`, `moderate`, `low`}, the terms that can be generated using logical negation are ⟨`BloodPressure` **NOT** = `high`⟩, ⟨`BloodPressure` **NOT** = `moderate`⟩, and ⟨`BloodPressure` **NOT** = `low`⟩.

In AntMiner+ (Martens et al. 2007), intervals are produced from ordinal attributes, generating terms of the form ⟨`Blood Pressure` ≥ `moderate`⟩. However, this strategy does not allow generating terms that cover cases having the upper and the lower values of the ordinal attribute (e.g., the case in which the `Blood Pressure` is `high` or `low`) and sharing the same class. This can be covered using the logical negation operator by generating the term ⟨`BloodPressure` **NOT** = `moderate`⟩.[1]

[1] Of course, in some applications, domain experts do not prefer terms such as **NOT** `moderate`. In such domains, the use of **NOT** can be limited to the highest and lowest values of an attribute.

Although using negated attributes doubles the size of the construction graph, it enables the construction of rules that have greater coverage of the training cases. Consequently, a lower number of rules is produced, which improves the simplicity of the output.

While our logical negation operator allows the construction of terms such as ⟨`Condition` **NOT** = `good`⟩ and ⟨`color` **NOT** = `green`⟩, which can only be generated in AntMiner+ by mining multiple rules, the interval approach used in AntMiner+ can also generate terms such as ⟨`Condition` ∈ [`Good` : `Very Good`]⟩ that can only be generated with our logical negation operator by mining multiple rules. Luckily, the two approaches (logical negation and generated intervals) are not mutually exclusive. It is possible to include both approaches in a single implementation, and we would like to consider this in future work. Depending on the preference of the domain expert, or the nature of the problem domain, the user would designate which attributes will use generated intervals, and which will use logical negation.

7 Incorporating stubborn ants

As discussed in Sect. 3.2, stubborn ants (Abdelbar 2008) are an ACO variation in which each ant has its own bias, based on its personal search history. Here, we adapt the stubborn ants idea and use it in the context of Ant-Miner. In Abdelbar (2008), each ant was biased toward the solution it had constructed in the previous iteration. In the approach we use here, each ant retains a memory of the best-ever solution it has personally constructed in the past, and is biased toward this personal historical-best solution.

The original Ant-Miner algorithm can be thought of as employing a large number of ants (specified by the `max_trials` parameter), each making a single trial. In order to use stubborn ants, we will consider that we have some number of ants (specified by a new parameter `number_of_stubborn_ants`), each making a number of trials equal to the result of dividing the parameter `max_trials` by the parameter `number_of_stubborn_ants`. Thus, the total number of trials remains the same and equal to the `max_trials` parameter. Algorithm 3 shows high-level pseudo-code for Ant-Miner with stubborn ants.

Basically, each ant carries out several trials in the execution of the algorithm. Each ant_t memorizes the best solution R_t^+ that it has constructed during its own trials. During rule construction, if $term_{ij}$ belongs to the antecedent of rule R_t^+, then $term_{ij}$ will have an amplified probability of being selected by ant_t, with the degree of amplification depending on the quality of the solution R_t^+. The probability that a term will be added to the current rule is given by the following formula:

$$P_{ij}(t) = \frac{V_{ij}}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (V_{rs})} \tag{18}$$

where

$$V_{ij} = \begin{cases} \eta_{ij} \cdot \tau_{ij}(t) + \eta_{ij} \cdot \tau_{ij}(t) \cdot Q(R_t^+) & \text{if } term_{ij} \text{ belongs to } R_t^+ \\ \eta_{ij} \cdot \tau_{ij}(t) & \text{otherwise} \end{cases} \tag{19}$$

where $Q(R_t^+)$ represents the quality of ant_t's historical-best memorized rule R_t^+ (recall that Q always returns nonnegative values).

Note that the parameter `number_of_stubborn_ants` affects the behavior of the algorithm; as the number of stubborn ants decreases, the stubbornness effect is stronger, given that the `max_trials` parameter remains unchanged. For example, suppose that the

Algorithm 3 Pseudo-code of Ant-Miner with the Stubborn Ants Extension.

```
Begin Ant-Miner with Stubborn Ants
training_set ← all training cases;
discovered_rule_set ← φ;
InitializePheromoneAmounts();
while |training_set| > max_uncovered_cases do
    CalculateHeuristicValues();
    R_best ← φ;
    i ← 0;
    repeat
        for t = 0 to number_of_stubborn_ants do
            ConstructRuleAntecedent(ant_t);
            ComputeRuleClass(ant_t);
            R_current ← PruneRule(ant_t);
            Q_current ← CalculateRuleQuality(R_current);
            UpdatePheromone(R_current);
            if Q_current > Q_best then
                R_best ← R_current;
            end if
            if Q_current > Q+_best then
                R+_t ← R_current;
            end if
            i ← i + 1;
        end for
    until i = max_trials OR Convergence()
    discovered_rule_set ← discovered_rule_set + R_best;
    training_set ← training_set − Covered_Cases(R_best);
end while
End
```

maximum trials allowed is 3,000; if the `number_of_stubborn_ants` is 3,000, then each ant will carry out only one trial, which is the case in the original Ant-Miner algorithm. On the other hand, if the `number_of_stubborn_ants` is 30, then each ant can carry out up to 100 trials. If the `number_of_stubborn_ants` is 10, then the number of trials that can be performed by a single ant is 300, in which case the stubbornness effect is more pronounced.

In our experimental results, the `number_of_stubborn_ants` is 5, and the `max_trials` parameter is equal to 1,500—which means that each ant will carry out 300 trials (unless convergence occurs sooner). Setting `number_of_stubborn_ants` to 5 means that up to 5 different previously-generated solutions could be memorized and used to influence future solution construction.

8 Giving ants personality

As discussed in Sect. 3.2, in most ACO systems, each ant probabilistically generates its solution in a given iteration according to (5). The exponents α and β in this equation are used to adjust the relative emphases of the pheromone (τ) and heuristic information (η),

respectively. In this section, we adapt the "ants with personality" approach proposed as future work in Abdelbar (2008) to the Ant-Miner framework. Each ant k is allowed to have its own *personality* by allowing it to have its own values of the α_k and β_k parameters. In other words, some ants will give more importance to pheromone information, while others will give more importance to heuristic information. The α_k and β_k parameters are each independently drawn from a Gaussian distribution centered at 2 with a standard deviation of 1.

The idea of setting different values of α and β for each ant was explored as early as 2007 in AntMiner+ (Martens et al. 2007). As mentioned in Sect. 3.1, in AntMiner+, the α and β values are considered as decision components in the construction graph. They are selected probabilistically by each ant before rule construction using pheromone information. However, the values of these parameters are limited to integer values between 1 and 3. Moreover, since α and β values are selected according to pheromone information, ants could potentially converge on specific values at some point, which can limit further exploration in the rest of the algorithm's execution.

9 Experimental evaluation methodology

9.1 Datasets

The performance of Ant-Miner with the proposed extensions was evaluated using 23 public-domain datasets from the UCI dataset repository (Asuncion and Newman 2007). The main characteristics of the datasets are shown in Table 2.

Since Ant-Miner does not handle continuous attributes directly, the datasets containing continuous attributes were discretized in a preprocessing step, using the C4.5-Disc (Kohavi and Sahami 1996) discretization algorithm. Briefly, the C4.5-Disc algorithm works as follows. For each continuous attribute, a two-attribute dataset is constructed. The first attribute of the constructed dataset contains the values (extracted from the training set) of the numeric attribute to be discretized, and the second is the class attribute. The C4.5 decision tree generation algorithm is then applied to this reduced dataset. Thus, C4.5 constructs a decision tree in which all internal nodes refer to the attribute being discretized. Each path from the root to a leaf node in the constructed decision tree corresponds to the definition of a categorical interval produced by C4.5.

For each cross-validation fold, we separately discretized (using C4.5-Disc) the training set and then used the created discrete intervals to discretize the test set. This separation is necessary because if we had discretized the entire dataset before creating the cross-validation folds, the discretization method would have had access to the test data—which would have compromised the reliability of the experiments. The ten datasets that contained continuous attributes and required preprocessing are: breast cancer (wisconsin), contraceptive method choice, statlog credit (australian), statlog credit (german), dermatology, glass, heart (cleveland), ionosphere, iris, and wine.

9.2 Algorithms evaluated

A list of the classification algorithms used in our experiments is presented in Table 3. In this paper, we have presented five extensions to the Ant-Miner algorithm. It would not be practical to experimentally evaluate every possible combination of these extensions because this would lead to $2^5 = 32$ variations in theory (although several are not very meaningful, as

Table 2 Description of datasets used in the experiments

Dataset	Size	Attributes	Classes
Audiology	266	69	24
Balance scale	625	4	3
Breast cancer (wisconsin)	286	9	2
Car evaluation	1,728	6	4
Contraceptive method choice	1,473	9	3
Statlog credit (australian)	690	14	2
Statlog credit (german)	1,000	20	2
Dermatology	366	33	6
Glass	214	10	7
Hayes-roth	160	4	3
Heart (cleveland)	303	12	3
Ionosphere	350	34	2
Iris	150	4	3
Monks	432	6	2
Mushrooms	8,124	22	2
Post operative patient	90	8	3
Soybean	307	35	19
SPECT heart	267	22	2
Teaching assistant evaluation	151	5	3
Tic-tac-to	958	9	2
Voting records	435	16	2
Wine	178	13	3
Zoo	101	17	7

discussed below). Therefore, in our experiments, we restrict ourselves to 12 combinations of these variations, as shown in Table 3. These 12 combinations were chosen based on the following rationale.

We divide the five extensions presented in this paper into three groups:

(a) The multipheromone extension and the quality contrast intensifier extension, denoted collectively as μAnt-Miner. The multipheromone extension without the quality contrast intensifier is denoted μ^-Ant-Miner. As discussed earlier, the quality contrast intensifier is not intended to be used without the multipheromone extension.
(b) The logical negation operator extension, denoted Ant-Miner¬. This extension has been considered as a separate group by itself because, out of all the 5 extensions, it is the only one that modifies the construction graph (i.e., the search space) of Ant-Miner. (The other 4 extensions modify the way the search is performed, rather than modifying the search space.)
(c) The stubborn ants and the ants with personality extensions, denoted collectively as ψAnt-Miner. Stubborn ants, without personality, are denoted ψ^-Ant-Miner, and personality without stubbornness is denoted ψ^*Ant-Miner.

We experimentally evaluate all possible combinations of these three groupings, eight variations in total: Ant-Miner, μAnt-Miner, Ant-Miner¬, ψAnt-Miner, μAnt-Miner¬, ψAnt-Miner¬, $\mu\psi$Ant-Miner, and $\mu\psi$Ant-Miner¬.

Table 3 Summary of the classification algorithms used in the experiments

Algorithm	Abbr.	Description
Ant-Miner	AM	Original Ant-Miner algorithm
μAnt-Miner	μAM	Ant-Miner with multiple pheromones types and quality contrast intensifier
μ^{-}Ant-Miner	μ^{-}AM	μAnt-Miner without quality contrast intensifier
Ant-Miner$^{\neg}$	AM$^{\neg}$	Ant-Miner using logical negation operator
ψAnt-Miner	ψAM	Ant-Miner with stubborn ants with personality
ψ^{-}Ant-Miner	ψ^{-}AM	Ant-Miner with stubborn ants only
ψ^{*}Ant-Miner	ψ^{*}AM	Ant-Miner with personality only
μAnt-Miner$^{\neg}$	μAM$^{\neg}$	μAnt-Miner with logical negation operator
μ^{-}Ant-Miner$^{\neg}$	μ^{-}AM$^{\neg}$	μAnt-Miner$^{\neg}$ without quality contrast intensifier
ψAnt-Miner$^{\neg}$	ψAM$^{\neg}$	ψAnt-Miner using logical negation operator
$\mu\psi$Ant-Miner	$\mu\psi$AM	μAnt-Miner using stubborn ants with personality
$\mu\psi$Ant-Miner$^{\neg}$	$\mu\psi$AM$^{\neg}$	Ant-Miner with all of the five proposed extensions
Ripper	JRip	Weka (Witten and Frank 2005) implementation of the RIPPER algorithm (Cohen 1995)
PART	PART	Weka implementation of the PART algorithm (Frank and Witten 1998)
C4.5-Rules	C4.5r	Quinlan's (1993) implementation of the C4.5-Rules algorithm (Release 8)
Ripper (PD)	JRip$_{pd}$	JRip with prior discretization of the dataset
PART (PD)	PART$_{pd}$	PART with prior discretization of the dataset
C4.5-Rules (PD)	C4.5r$_{pd}$	C4.5r with prior discretization of the dataset

Additionally, in order to isolate the effect of the quality contrast intensifier, we evaluate the combinations μ^{-}Ant-Miner and μ^{-}Ant-Miner$^{\neg}$. This will allow us to compare μ^{-}Ant-Miner to μAnt-Miner and to compare μ^{-}Ant-Miner$^{\neg}$ to μAnt-Miner$^{\neg}$. Further, to isolate the effect of each of the extensions combined into the ψ grouping, we also evaluate ψ^{-}Ant-Miner and ψ^{*}Ant-Miner.

We compare each of the 12 Ant-Miner variations to the original Ant-Miner algorithm, as well as to three state-of-the-art conventional rule induction algorithms, Ripper (Cohen 1995), PART (Frank and Witten 1998), and C4.5-Rules (Quinlan 1993).

These three conventional rule induction algorithms are able to process continuous attributes and do not require prior discretization of the dataset. Therefore, for the sake of fairness, we applied the following methodology for the ten datasets with continuous attributes: each of three rule induction algorithms was run twice, once on the original dataset and a second time with prior discretization of the dataset (using the C4.5-Disc algorithm, fold by fold, as described in Sect. 9.1). As indicated in Table 3, we use JRip$_{pd}$, PART$_{pd}$, and C4.5r$_{pd}$, to refer to each of the three rule induction algorithms, respectively, run with prior discretization of the dataset.

9.3 Experimental setup

The experiments were conducted using a ten-fold cross-validation procedure for each dataset. A ten-fold cross-validation procedure consists of dividing the dataset into ten partitions of cases, wherein each partition has a similar number of cases and class distribution. 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.

Table 4 Algorithm parameters used in the experiments

	Parameter	Value
General ant-miner parameters	`max_trials`	1,500
	`max_uncovered_cases`	10
	`min_cases_per_rule`	5
	`no_rules_converg`	10
Quality contrast intensifier	ϕ_1	0.8
	ϕ_2	0.5
Stubborn ants	`number_of_stubborn_ants`	5
Personality	Mean value of α and β	2
	Standard deviation of α and β	1

For stochastic classification algorithms—i.e., Ant-Miner and its variations using the proposed extensions—the algorithm is run fifteen times using a different random seed to initialize the search for each partition of the cross-validation. In the case of the deterministic algorithms—i.e., Ripper, PART, and C4.5-Rules—each of them is run just once for each partition of the cross-validation.

The number of rules generated (which represents the simplicity of the output) and the predictive accuracy of the generated rules were recorded to evaluate the performance of the algorithm.

Although dynamic parameter adaptation schemes have been investigated for ACO algorithms (Stützle et al. 2010), we use static parameter settings in this paper in order to isolate the effects of the proposed extensions. Table 4 shows the parameter settings used in our experiments. The general parameters follow the parameter settings used in Parpinelli et al. (2002). The extension-specific parameters were determined based on initial ad hoc experimentation.

The source code, in the C# programming language, for our extended version of Ant-Miner, including all extensions presented in this paper, is available as online supplementary material.

10 Experimental results

10.1 Results for ant-miner extensions

Tables 5 and 6 show predictive accuracy results, while Tables 7 and 8 show model size results for the Ant-Miner extensions. Results of predictive accuracy and model size for the first 12 datasets are shown in Tables 5 and 7, respectively, while the results for the remaining 11 datasets are shown in Tables 6 and 8.

For each dataset, each table shows the mean and standard deviation ($^{mean}_{\pm\ stdv.}$) of the measure related to the table for each of the used algorithms. In addition, an entry is underlined if, for the corresponding dataset, the value obtained by the corresponding algorithm is the best (highest in accuracy or lowest in model size) among all values achieved by all 12 evaluated extensions. Further, a value is shown in boldface if the difference between it and the best value is less than the average of the two standard deviations (i.e., the average of the entry's standard deviation and the best entry's standard deviation). For example, in Table 5, for the `aud` dataset, the best value is 82.92 (corresponding to $\mu\psi$ Ant-Miner)—therefore, this value is underlined—and its corresponding standard deviation is 2.2. The value, for example,

Table 5 Predictive accuracy results (%) for the first group of datasets. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	aud	bal	bcw	car	cmc	c-a	c-g	drm	gls	hay	hrt	ion
AM	61.48	66.52	92.53	77.80	44.18	**83.39**	69.84	89.06	53.27	46.06	55.89	83.07
	± 1.4	*± 2.1*	*± 1.1*	*± 1.8*	*± 3.5*	*± 3.2*	*± 0.9*	*± 2.5*	*± 3.6*	*± 4.4*	*± 3.4*	*± 2.7*
μAM	**81.95**	**78.41**	**94.05**	**94.79**	**47.61**	**82.88**	70.70	**96.00**	**65.94**	**66.46**	**58.28**	**85.95**
	± 0.8	*± 1.3*	*± 1.8*	*± 1.9*	*± 1.5*	*± 3.5*	*± 0.1*	*± 1.3*	*± 3.7*	*± 4.5*	*± 3.5*	*± 2.6*
μ^{-}AM	80.00	76.58	**93.39**	92.06	43.89	**83.42**	70.08	**95.37**	**62.54**	**64.02**	**57.92**	82.06
	± 1.0	*± 1.5*	*± 1.7*	*± 2.3*	*± 1.7*	*± 3.6*	*± 0.6*	*± 1.6*	*± 4.6*	*± 5.1*	*± 3.9*	*± 3.0*
AM$^{\neg}$	67.95	70.68	**92.83**	84.76	**46.81**	**84.32**	**71.03**	90.28	61.17	50.76	**57.92**	**86.49**
	± 1.1	*± 1.8*	*± 1.8*	*± 1.8*	*± 1.9*	*± 2.7*	*± 0.8*	*± 2.9*	*± 3.6*	*± 4.7*	*± 3.7*	*± 2.9*
ψAM	76.58	65.70	**93.89**	76.00	**46.62**	**83.61**	70.30	94.15	63.06	48.66	**57.14**	84.24
	± 1.8	*± 2.2*	*± 1.5*	*± 1.7*	*± 1.6*	*± 2.4*	*± 0.8*	*± 2.5*	*± 3.1*	*± 4.2*	*± 3.4*	*± 2.3*
ψ^{-}AM	76.22	67.50	**93.57**	79.40	45.01	**83.26**	70.82	93.11	**63.21**	48.41	56.08	**86.20**
	± 1.8	*± 2.1*	*± 1.6*	*± 1.5*	*± 1.7*	*± 2.4*	*± 0.6*	*± 2.2*	*± 3.4*	*± 4.1*	*± 3.1*	*± 2.0*
ψ^{*}AM	63.61	65.74	**93.00**	75.93	44.37	**83.07**	69.30	91.86	60.44	49.38	**57.29**	83.07
	± 1.8	*± 2.4*	*± 2.1*	*± 1.2*	*± 1.8*	*± 2.6*	*± 0.7*	*± 2.7*	*± 3.0*	*± 4.3*	*± 3.8*	*± 2.1*
μAM$^{\neg}$	78.30	70.59	92.66	88.77	**46.51**	**84.78**	70.95	90.82	60.71	58.52	**59.62**	80.49
	± 0.9	*± 2.2*	*± 1.5*	*± 1.6*	*± 1.6*	*± 3.2*	*± 0.2*	*± 3.3*	*± 3.5*	*± 4.4*	*± 3.5*	*± 3.1*
μ^{-}AM$^{\neg}$	75.33	68.28	91.23	73.76	43.19	**83.29**	70.01	84.69	56.44	54.52	54.91	80.33
	± 0.6	*± 2.5*	*± 2.5*	*± 0.0*	*± 0.5*	*± 3.3*	*± 0.0*	*± 3.1*	*± 3.7*	*± 4.9*	*± 3.0*	*± 3.4*
ψAM$^{\neg}$	74.81	70.01	**93.02**	80.10	**46.02**	**84.68**	**71.64**	93.83	**63.38**	52.65	**57.63**	**85.41**
	± 1.7	*± 2.0*	*± 1.4*	*± 1.8*	*± 1.9*	*± 2.8*	*± 0.8*	*± 2.6*	*± 3.4*	*± 4.0*	*± 3.6*	*± 2.8*
$\mu\psi$AM	**82.92**	**79.73**	**94.51**	**94.83**	**47.76**	**82.57**	70.38	**97.40**	**66.61**	**68.46**	**57.82**	**86.95**
	± 2.2	*± 4.5*	*± 1.8*	*± 1.3*	*± 2.8*	*± 2.5*	*± 0.2*	*± 2.5*	*± 3.8*	*± 6.1*	*± 3.0*	*± 2.6*
$\mu\psi$AM$^{\neg}$	80.30	68.41	92.60	89.33	**46.72**	**84.82**	70.11	87.78	62.85	58.37	**60.07**	81.52
	± 1.6	*± 2.6*	*± 1.6*	*± 1.4*	*± 1.1*	*± 3.1*	*± 0.3*	*± 3.4*	*± 3.3*	*± 5.2*	*± 3.3*	*± 3.1*

corresponding to μAnt-Miner is shown in bold because its value (81.95) differs from the best value (82.92) by less than the average of the two standard deviations (2.2 and 0.8).

Table 9 shows the average rank of each extension for predictive accuracy and model size, with the best performances shown in bold. The average rank for a given algorithm g is obtained by first computing the rank of g on each dataset individually. The individual ranks are then averaged across all datasets to obtain the overall average rank, which will be in the range from 1 to 12. Note that the lower the value of the rank, the better the algorithm.

It should be noted that the summary of results in Table 9 and the corresponding discussion in the remainder of the paper are based on the average ranking of the algorithms for each performance criteria (accuracy and model size), rather than the direct average values of those two criteria. The rationale for this is as follows. Statistically speaking, the average accuracy and model size across all datasets has no clear meaning, because an average should be computed across different values of the same random variable, but the value of the

Table 6 Predictive accuracy results (%) for the second group of datasets. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	irs	mon	msh	pop	soy	spt	tae	ttt	vot	win	zoo
AM	**92.93**	60.99	97.09	**71.74**	55.89	78.59	**44.65**	69.73	92.69	88.13	95.48
	± *3.0*	± *1.8*	± *0.3*	± *3.5*	± *3.3*	± *1.5*	± *7.7*	± *3.1*	± *1.4*	± *3.2*	± *1.5*
μAM	**94.66**	**63.78**	**98.20**	**73.95**	**86.86**	79.12	**44.15**	94.50	92.76	**93.17**	99.08
	± *1.9*	± *0.6*	± *0.4*	± *3.0*	± *8.1*	± *0.6*	± *8.7*	± *2.9*	± *1.0*	± *2.2*	± *0.6*
μ^{-}AM	**92.90**	**63.25**	**98.53**	**72.50**	**86.05**	**79.93**	**40.29**	90.38	**94.41**	90.58	98.26
	± *2.1*	± *0.6*	± *0.4*	± *3.1*	± *3.0*	± *0.8*	± *7.4*	± *3.4*	± *1.9*	± *2.4*	± *0.8*
AM$^{\neg}$	**93.60**	62.38	98.13	**71.86**	69.73	77.74	**45.41**	71.87	**94.34**	92.56	96.63
	± *3.4*	± *1.4*	± *0.4*	± *2.9*	± *4.9*	± *0.7*	± *10.0*	± *2.0*	± *1.7*	± *2.4*	± *2.2*
ψAM	**93.74**	60.62	97.91	70.83	59.87	**78.92**	**44.05**	70.38	**94.54**	91.52	95.33
	± *3.2*	± *1.4*	± *0.5*	± *2.6*	± *2.4*	± *1.4*	± *7.1*	± *2.4*	± *1.5*	± *3.0*	± *1.6*
ψ^{-}AM	**93.98**	59.71	97.82	69.33	60.37	**78.87**	**43.14**	69.88	**94.50**	90.09	95.00
	± *3.0*	± *1.6*	± *0.4*	± *2.4*	± *3.1*	± *1.4*	± *8.2*	± *2.0*	± *1.1*	± *3.1*	± *1.0*
ψ^{*}AM	**93.19**	60.07	97.41	70.17	51.36	**78.75**	**45.10**	69.98	92.29	89.34	95.75
	± *3.3*	± *1.8*	± *0.4*	± *3.2*	± *2.4*	± *1.8*	± *8.5*	± *2.7*	± *1.3*	± *3.2*	± *1.4*
μAM$^{\neg}$	90.12	55.32	95.08	**74.00**	**85.67**	**79.25**	**39.75**	79.26	92.87	89.15	95.42
	± *3.5*	± *2.0*	± *1.5*	± *3.2*	± *3.5*	± *0.7*	± *9.2*	± *3.1*	± *0.6*	± *2.6*	± *2.2*
μ^{-}AM$^{\neg}$	84.41	58.50	91.21	**72.92**	**86.38**	79.24	**42.06**	65.26	90.69	87.30	96.01
	± *4.3*	± *2.2*	± *1.7*	± *3.6*	± *4.1*	± *0.4*	± *5.3*	± *0.0*	± *3.0*	± *2.3*	± *2.3*
ψAM$^{\neg}$	**93.51**	**62.85**	98.02	70.31	68.33	76.15	**44.77**	71.96	**95.16**	91.72	95.89
	± *3.4*	± *1.6*	± *0.3*	± *2.1*	± *4.6*	± *1.4*	± *7.6*	± *2.2*	± *1.5*	± *2.5*	± *1.5*
$\mu\psi$AM	**94.61**	**63.37**	**98.52**	**72.50**	**87.44**	**79.77**	**45.18**	**98.76**	**93.25**	**95.22**	**99.75**
	± *3.7*	± *0.9*	± *0.4*	± *3.8*	± *4.7*	± *1.3*	± *10.0*	± *2.7*	± *2.5*	± *2.5*	± *0.5*
$\mu\psi$AM$^{\neg}$	88.20	56.24	97.00	**74.25**	**83.51**	**79.25**	**42.38**	75.94	92.30	87.46	92.75
	± *2.6*	± *2.4*	± *1.3*	± *2.8*	± *3.8*	± *1.3*	± *8.0*	± *3.3*	± *1.6*	± *3.3*	± *2.9*

accuracy and model size in each dataset is a different random variable, since each dataset has a unique probability distribution of attribute values and classes. To see the problem in practice, one should note that the same given value of accuracy, say 70%, can be considered a very low accuracy in some datasets (e.g., in the bcw dataset, where all algorithms have accuracy higher than 90% in Table 5) but a very high accuracy in other datasets (e.g., in cmc, where all algorithms have accuracy around or smaller than 45% in Table 5). Hence, analyzing the results based on average accuracy and model size across all datasets is statistically meaningless and can lead to potentially misleading results. In contrast, an analysis based on average rankings mitigates the above problem, since a given rank obtained by an algorithm in a given dataset can be interpreted in the same way regardless of the relative difficulty of the dataset, so that the average ranking is a clearly interpretable and statistically meaningful measure.

Table 7 Model size results (number of rules) for the first group of datasets. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	aud	bal	bcw	car	cmc	c-a	c-g	drm	gls	hay	hrt	ion
AM	9.46 ± *0.1*	12.00 ± *0.0*	6.08 ± *0.3*	12.07 ± *0.2*	8.38 ± *0.3*	7.36 ± *0.4*	8.53 ± *0.2*	7.98 ± *0.1*	**5.89** ± *0.2*	4.95 ± *0.1*	7.04 ± *0.4*	5.22 ± *0.2*
μAM	13.85 ± *0.2*	29.42 ± *1.4*	6.70 ± *0.4*	24.61 ± *1.5*	20.00 ± *2.3*	9.56 ± *0.2*	14.89 ± *1.2*	7.19 ± *0.1*	9.89 ± *0.8*	8.69 ± *0.6*	14.33 ± *0.7*	5.70 ± *0.5*
μ^{-}AM	13.15 ± *0.2*	25.53 ± *1.2*	**4.57** ± *0.6*	20.12 ± *1.2*	10.31 ± *1.3*	4.20 ± *0.4*	**2.00** ± *0.0*	**7.11** ± *0.1*	9.91 ± *0.7*	9.18 ± *0.7*	11.77 ± *0.8*	4.26 ± *0.4*
AM$^{\neg}$	**8.07** ± *0.2*	10.18 ± *0.2*	5.21 ± *0.5*	10.31 ± *0.4*	7.67 ± *0.2*	6.76 ± *0.1*	7.80 ± *0.7*	**7.04** ± *0.1*	7.56 ± *0.2*	5.55 ± *0.2*	6.88 ± *0.2*	**4.07** ± *2.9*
ψAM	9.85 ± *0.4*	10.13 ± *0.2*	5.94 ± *0.3*	11.26 ± *0.4*	7.61 ± *0.3*	5.85 ± *0.2*	8.79 ± *0.2*	7.63 ± *0.1*	6.77 ± *0.3*	5.03 ± *0.2*	7.64 ± *0.4*	5.43 ± *0.2*
ψ^{-}AM	12.17 ± *0.5*	10.00 ± *0.3*	6.03 ± *0.2*	11.23 ± *0.3*	7.40 ± *0.6*	6.27 ± *0.5*	8.66 ± *0.6*	7.41 ± *0.2*	7.22 ± *0.2*	4.93 ± *0.4*	8.09 ± *0.2*	5.35 ± *0.3*
ψ^{*}AM	8.71 ± *0.6*	11.30 ± *0.4*	6.19 ± *0.3*	10.60 ± *1.8*	6.85 ± *0.5*	6.16 ± *0.6*	8.73 ± *0.5*	7.34 ± *0.3*	**5.98** ± *0.2*	4.93 ± *0.3*	7.17 ± *0.5*	4.94 ± *0.2*
μAM$^{\neg}$	12.45 ± *0.3*	22.75 ± *0.4*	6.93 ± *0.4*	18.63 ± *1.9*	15.96 ± *0.4*	6.04 ± *0.9*	11.73 ± *1.0*	9.21 ± *0.6*	8.02 ± *0.7*	5.49 ± *0.4*	11.73 ± *0.8*	5.07 ± *0.3*
μ^{-}AM$^{\neg}$	11.72 ± *0.2*	**6.01** ± *0.6*	**4.17** ± *0.5*	**2.00** ± *0.0*	**3.50** ± *0.4*	**3.00** ± *0.0*	**2.00** ± *0.0*	8.14 ± *0.7*	7.36 ± *0.3*	**3.63** ± *0.4*	8.00 ± *0.8*	**3.31** ± *0.1*
ψAM$^{\neg}$	8.83 ± *0.3*	10.00 ± *0.2*	5.02 ± *0.6*	10.26 ± *0.4*	6.95 ± *0.4*	5.16 ± *0.3*	7.78 ± *0.3*	7.31 ± *0.2*	7.72 ± *0.3*	5.84 ± *0.3*	**6.18** ± *0.4*	4.28 ± *0.2*
$\mu\psi$AM	14.01 ± *0.6*	29.59 ± *1.5*	**4.85** ± *1.7*	21.50 ± *1.1*	11.60 ± *1.6*	5.52 ± *0.4*	11.20 ± *1.2*	**7.29** ± *0.6*	9.05 ± *0.9*	6.56 ± *0.4*	11.02 ± *1.1*	4.95 ± *0.4*
$\mu\psi$AM$^{\neg}$	10.78 ± *0.5*	17.58 ± *1.6*	6.03 ± *0.7*	18.28 ± *1.5*	17.63 ± *1.9*	6.02 ± *0.2*	13.03 ± *0.8*	9.45 ± *1.2*	7.13 ± *0.5*	4.77 ± *0.3*	12.86 ± *0.6*	4.66 ± *0.8*

The multipheromone extension, combined with the quality contrast intensifier, generally tends to improve the average predictive accuracy rank. For example, μAnt-Miner has a much better accuracy rank than Ant-Miner, and $\mu\psi$Ant-Miner has a much better accuracy rank than ψAnt-Miner. An exception is that $\mu\psi$Ant-Miner$\neg$ has a worse accuracy rank than ψAnt-Miner$\neg$. However, in general, there is an accompanying relative decline in the model size rank. For example, μAnt-Miner has a worse model size rank compared to Ant-Miner; $\mu\psi$Ant-Miner has a slightly worse model size rank than ψAnt-Miner; and, $\mu\psi$Ant-Miner$\neg$ has a worse size rank than ψAnt-Miner$\neg$.

The use of the quality contrast intensifier in conjunction with the multipheromone system (μAnt-Miner) generally improves predictive accuracy compared to the multipheromone system without the contrast intensifier (μ^{-}Ant-Miner), however, this often comes at the expense of increasing the model size—this is not unexpected since the quality contrast intensifier favors confidence over support. Comparing μAnt-Miner to μ^{-}Ant-Miner, we find that

Table 8 Model size results (number of rules) for the second group of datasets. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	irs	mon	msh	pop	soy	spt	tae	ttt	vot	win	zoo
AM	4.90 ± 0.2	7.36 ± 0.3	6.53 ± 0.4	4.08 ± 0.2	**7.65** ± 0.5	10.93 ± 0.3	5.40 ± 0.8	7.30 ± 0.1	4.33 ± 0.1	5.02 ± 0.2	5.95 ± 0.2
μAM	**4.05** ± 0.3	10.07 ± 0.7	5.31 ± 0.1	5.17 ± 0.1	21.25 ± 0.3	21.50 ± 0.2	5.34 ± 0.5	10.88 ± 0.5	3.64 ± 0.2	5.10 ± 0.1	6.00 ± 0.0
μ^{-}AM	**4.00** ± 0.3	4.44 ± 0.8	5.01 ± 0.2	2.30 ± 0.0	20.85 ± 0.3	21.07 ± 0.2	4.15 ± 0.4	10.51 ± 0.6	**3.00** ± 0.1	**4.04** ± 0.0	6.00 ± 0.0
AM$^{\neg}$	**4.06** ± 0.2	7.91 ± 0.4	5.02 ± 0.1	3.68 ± 0.6	10.74 ± 1.0	8.87 ± 0.1	5.00 ± 0.8	6.27 ± 0.6	3.99 ± 0.3	4.83 ± 0.1	**5.04** ± 0.3
ψAM	4.41 ± 0.3	7.43 ± 0.4	5.75 ± 0.5	4.68 ± 0.4	9.90 ± 0.7	9.40 ± 1.0	5.90 ± 0.8	7.41 ± 0.2	4.67 ± 0.2	5.81 ± 0.2	5.91 ± 0.2
ψ^{-}AM	4.55 ± 0.3	7.33 ± 0.3	6.20 ± 0.2	4.49 ± 0.6	11.72 ± 0.8	9.33 ± 0.8	5.00 ± 0.7	7.26 ± 0.5	4.40 ± 0.4	5.01 ± 0.2	5.98 ± 0.2
ψ^{*}AM	4.83 ± 0.2	7.46 ± 0.3	6.70 ± 0.3	4.51 ± 0.6	9.13 ± 0.8	8.56 ± 0.6	5.06 ± 0.8	7.47 ± 0.4	4.03 ± 0.3	4.68 ± 0.2	5.56 ± 0.3
μAM$^{\neg}$	**4.25** ± 0.3	3.53 ± 0.5	5.87 ± 0.1	2.52 ± 0.0	20.79 ± 0.5	12.77 ± 0.4	4.07 ± 0.4	8.37 ± 0.3	5.95 ± 0.3	5.65 ± 0.1	5.74 ± 0.1
μ^{-}AM$^{\neg}$	**4.00** ± 0.0	**2.00** ± 0.0	**3.84** ± 0.3	**2.00** ± 0.0	17.52 ± 0.5	**2.00** ± 0.0	**3.06** ± 0.1	**2.00** ± 0.0	5.52 ± 0.5	**4.00** ± 0.1	**5.00** ± 0.0
ψAM$^{\neg}$	4.60 ± 0.2	7.53 ± 0.4	5.25 ± 0.4	3.80 ± 0.6	10.65 ± 0.9	8.22 ± 0.2	4.04 ± 0.6	6.58 ± 0.4	4.11 ± 0.2	4.60 ± 0.1	5.50 ± 0.2
$\mu\psi$AM	**4.05** ± 0.5	4.51 ± 0.5	5.40 ± 0.5	2.50 ± 0.2	22.01 ± 0.7	20.35 ± 0.7	4.55 ± 0.5	8.86 ± 0.7	**3.00** ± 0.0	4.51 ± 0.2	6.00 ± 0.0
$\mu\psi$AM$^{\neg}$	**4.22** ± 0.2	3.93 ± 0.7	6.87 ± 0.6	2.79 ± 0.5	19.18 ± 0.6	12.29 ± 1.2	4.17 ± 0.3	6.69 ± 0.3	5.65 ± 0.5	5.55 ± 0.2	5.45 ± 0.2

the accuracy rank improves, but the size rank declines. Similarly, comparing μAnt-Miner$\neg$ to μ^{-}Ant-Miner$\neg$, we find that the accuracy rank improves and the size rank declines.

Using the logical negation operator generally tends to produce simpler (smaller) classification models. Comparing Ant-Miner$\neg$ to Ant-Miner, ψAnt-Miner$\neg$ to ψAnt-Miner, μAnt-Miner$\neg$ to μAnt-Miner, μ^{-}Ant-Miner$\neg$ to μ^{-}Ant-Miner, and $\mu\psi$Ant-Miner$\neg$ to $\mu\psi$Ant-Miner, we find that the model size rank improves.

The effect of negation on predictive accuracy is less consistent and appears to depend on whether it is used in combination with the multipheromone extension. In the absence of multipheromone, negation tends to improve accuracy. Comparing Ant-Miner$\neg$ to Ant-Miner and ψAnt-Miner$\neg$ to ψAnt-Miner, we find that the accuracy rank improves in both cases. When used in conjunction with multipheromone, we find that negation reduces accuracy. Comparing μAnt-Miner$\neg$ to μAnt-Miner, μ^{-}Ant-Miner$\neg$ to μ^{-}Ant-Miner, and $\mu\psi$Ant-

Table 9 Average rankings of the Ant-Miner algorithms

Algorithm	Predictive accuracy	Model size
AM	9.3	7.0
μAM	3.3	10.0
μ^{-}AM	5.1	6.1
AM$^{\neg}$	5.5	4.7
ψAM	6.8	7.0
ψ^{-}AM	7.4	6.7
ψ^{*}AM	8.9	6.0
μAM$^{\neg}$	6.7	8.3
μ^{-}AM$^{\neg}$	9.6	**2.7**
ψAM$^{\neg}$	5.7	4.5
$\mu\psi$AM	**2.5**	7.3
$\mu\psi$AM$^{\neg}$	6.9	7.2

Miner¬ to $\mu\psi$Ant-Miner, we find that the improvement in model size comes at the expense of a decline in accuracy.

The most likely explanation for why the effect of logical negation on predictive accuracy depends on whether it is used in combination with multipheromone is the following. Without multipheromone, the rule quality evaluation function is Ant-Miner's original evaluation function *sensitivity* × *specificity* (2). With multipheromone, the evaluation function is *support* + *confidence* (14). Terms with negation tend to have high support and may overshadow the confidence component in (14); e.g., a term such as ⟨`Condition` **NOT** = `Good`⟩ is likely to match more cases than any specific value of `Condition` (this effect is even more magnified as the number of possible values for an attribute increases).

The effect of the ψ combination on performance seems to depend on whether it is combined with the use of negation. Without logical negation, the accuracy rank consistently improves, while the model size rank does not decline—we observe this comparing ψAnt-Miner to Ant-Miner and $\mu\psi$Ant-Miner to μAnt-Miner. When the ψ combination is used in conjunction with logical negation, its effect on performance depends on whether multipheromone is also employed. Comparing ψAnt-Miner¬ to Ant-Miner¬, we find that both accuracy and model size improve slightly; however, comparing $\mu\psi$Ant-Miner¬ to μAnt-Miner¬, we find that the accuracy rank declines slightly but the size rank improves.

To isolate the effect of stubborn ants alone without the personality extension, we compare the performance of ψ^{-}Ant-Miner to Ant-Miner and ψAnt-Miner to ψ^{*}Ant-Miner. Comparing ψ^{-}Ant-Miner to Ant-Miner, we find that the accuracy and size ranks both improve. Comparing ψAnt-Miner to ψ^{*}Ant-Miner, we find that accuracy improves while the size rank declines.

Similarly, to isolate the effect of the personality extension, we compare ψ^{*}Ant-Miner to Ant-Miner and ψAnt-Miner to ψ^{-}Ant-Miner. Comparing ψ^{*}Ant-Miner to Ant-Miner, we find that the accuracy and size ranks both improve. Comparing ψAnt-Miner to ψ^{-}Ant-Miner, we find that accuracy improves while the model size rank declines.

Using the vocabulary of multiobjective optimization (Deb 2009), we say an algorithm h is *dominated* by another algorithm g if g is not worse than h in any of the two performance criteria (predictive accuracy and model size) and g is better than h in at least one of those criteria. An algorithm g is said to be *Pareto-optimal* if it is not dominated by any other competing algorithm—this means g cannot be improved upon in any one performance measure

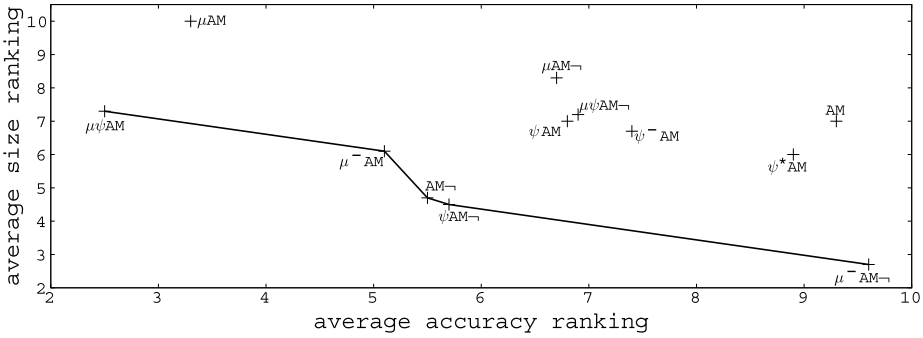

Fig. 1 Plot showing the average accuracy rank (x-axis) versus the average size rank (y-axis). The Pareto-frontier is shown as a connected line and consists of five algorithms: $\mu\psi$Ant-Miner, μ^-Ant-Miner, Ant-Miner¬, ψAnt-Miner¬, and μ^-Ant-Miner¬

Table 10 Results of the Friedmann test with the Nemenyi post-hoc test for predictive accuracy. The entry in row i and column j shows the symbol $\doteq$ if there is no statistically significant difference, at the 0.05 significance level, between algorithm i and algorithm j; if the difference is significant at the 0.05 level, then the symbol $\gg$ is shown if i has a better rank than j, and the symbol $\ll$ is shown if i has a worse rank than j

	AM	μAM	μ^-AM	AM¬	ψAM	ψ^-AM	ψ^*AM	μAM¬	μ^-AM¬	ψAM¬	$\mu\psi$AM	$\mu\psi$AM¬
AM		$\ll$	$\ll$	$\ll$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$
μAM	$\gg$		$\doteq$	$\doteq$	$\doteq$	$\gg$	$\gg$	$\doteq$	$\gg$	$\doteq$	$\doteq$	$\doteq$
μ^-AM	$\gg$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\gg$	$\doteq$	$\gg$	$\doteq$	$\doteq$	$\doteq$
AM¬	$\gg$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\gg$	$\doteq$	$\doteq$	$\doteq$
ψAM	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$
ψ^-AM	$\doteq$	$\ll$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$
ψ^*AM	$\doteq$	$\ll$	$\ll$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$
μAM¬	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\ll$	$\doteq$
μ^-AM¬	$\doteq$	$\ll$	$\ll$	$\ll$	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\ll$	$\ll$	$\doteq$
ψAM¬	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\gg$		$\doteq$	$\doteq$
$\mu\psi$AM	$\gg$	$\doteq$	$\doteq$	$\doteq$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\doteq$		$\gg$
$\mu\psi$AM¬	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	

without sacrificing in another performance measure. The set of Pareto-optimal algorithms are said to form a *Pareto-frontier*.

Figure 1 presents a visual representation of the average accuracy rank and the average model size rank. In the figure, the Pareto-frontier is shown as a connected line and consists of five algorithms: $\mu\psi$Ant-Miner, μ^-Ant-Miner, Ant-Miner¬, ψAnt-Miner¬, and μ^-Ant-Miner¬.

In order to determine the statistical significance of differences in performance between the 12 considered Ant-Miner variations, we apply a two-tailed Friedmann test with the Nemenyi post-hoc test, as outlined in Demsăr (2006), to the accuracy performance and to the model size performance of the 12 Ant-Miner variations. Table 10 shows the results for predictive accuracy and uses the following notation. The entry in row i and column j shows the symbol $\doteq$ if there is no statistically significant difference, at the 0.05 significance level, between algorithm i and algorithm j; if the difference is significant at the 0.05 level, then

Table 11 Results of the Friedmann test with the Nemenyi post-hoc test for model size. The entry in row i and column j shows the symbol $\doteq$ if there is no statistically significant difference, at the 0.05 significance level, between algorithm i and algorithm j; if the difference is significant at the 0.05 level, then the symbol $\gg$ is shown if i has a better rank than j, and the symbol $\ll$ is shown if i has a worse rank than j

	AM	μAM	μ^{-}AM	AM$^{\neg}$	ψAM	ψ^{-}AM	ψ^{*}AM	μAM$^{\neg}$	μ^{-}AM$^{\neg}$	ψAM$^{\neg}$	$\mu\psi$AM	$\mu\psi$AM$^{\neg}$
AM		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$	$\doteq$	$\doteq$
μAM	$\doteq$		$\ll$	$\ll$	$\doteq$	$\doteq$	$\ll$	$\doteq$	$\ll$	$\ll$	$\doteq$	$\doteq$
μ^{-}AM	$\doteq$	$\gg$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$
AM$^{\neg}$	$\doteq$	$\gg$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$
ψAM	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$	$\doteq$	$\doteq$
ψ^{-}AM	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\ll$	$\doteq$	$\doteq$	$\doteq$
ψ^{*}AM	$\doteq$	$\gg$	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$
μAM$^{\neg}$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$		$\ll$	$\ll$	$\doteq$	$\doteq$
μ^{-}AM$^{\neg}$	$\gg$	$\gg$	$\doteq$	$\doteq$	$\gg$	$\gg$	$\doteq$	$\gg$		$\doteq$	$\gg$	$\gg$
ψAM$^{\neg}$	$\doteq$	$\gg$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\gg$	$\doteq$		$\doteq$	$\doteq$
$\mu\psi$AM	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$		$\doteq$
$\mu\psi$AM$^{\neg}$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\ll$	$\doteq$	$\doteq$	

Table 12 Rules generated by Ant-Miner for `Monk's` dataset

					Supp.	Conf.
1	**IF**	(Attribute-1 = 1)	**THEN**	(0)	0.25	0.71
2	**Else IF**	(Attribute-1 = 2)	**THEN**	(0)	0.30	0.59
3	**Else IF**	(Attribute-4 = 2)	**THEN**	(0)	0.19	0.61
4	**Else IF**	(Attribute-4 = 3)	**THEN**	(0)	0.25	0.53
5	**Else IF**	(Attribute-5 = 4)	**THEN**	(1)	0.13	0.57
6	**Else IF**	(Attribute-5 = 3)	**THEN**	(0)	0.17	0.57
7	**Else IF**	(Attribute-5 = 2)	**THEN**	(0)	0.22	0.58
8	**Else**			(1)	–	–
					Cov.	**Acc.**
					95.00	62.0

the symbol $\gg$ is shown if i has a better rank than j, and the symbol $\ll$ is shown if i has a worse rank than j. Table 11 shows the corresponding results for model size and uses a similar notation.

To better understand how the proposed extensions affect the output model of Ant-Miner, Tables 12, 13, 14, and 15 show representative examples of the rule sets generated by Ant-Miner, Ant-Miner$\neg$, μAnt-Miner, and μ^{-}Ant-Miner$\neg$, respectively, for the `Monk's Problem` dataset. These outputs were generated by applying each algorithm on the same training set/test set pair. Each table shows the generated rules, the support of each rule, the confidence of each rule, and the total coverage of the rule set over the training set, as well as its predictive accuracy.

10.2 Comparison with conventional rule induction algorithms

For each of the three conventional rule induction algorithms (JRip, PART, and C4.5r), Tables 16–17 show the predictive accuracy and the number of generated rules, respectively. As discussed in Sect. 9.2, for each of the datasets with continuous attributes, for the sake of fairness, each of the three algorithms was run twice: once on the original dataset, and a second

Table 13 Rules generated by Ant-Miner¬ for `Monk's` dataset

					Supp.	Conf.
1	**IF**	(Attribute-1 = 1)	**THEN**	(0)	0.25	0.71
2	**Else IF**	(Attribute-2 = 1)	**THEN**	(0)	0.23	0.65
3	**Else IF**	(Attribute-3 = 1)	**THEN**	(1)	0.28	0.55
4	**Else IF**	(Attribute-4 = 2)	**THEN**	(0)	0.25	0.79
5	**Else IF**	(Attribute-2 = 2)	**THEN**	(0)	0.31	0.77
6	**Else IF**	(Attribute-1 **NOT**= 3)	**THEN**	(0)	0.68	0.68
7	**Else**			(1)	–	–
					Cov.	**Acc.**
					100.0	67.85

Table 14 Rules generated by μAnt-Miner for `Monk's` dataset

					Supp.	Conf.
1	**IF**	(Attribute-1 = 1)	**AND**			
		(Attribute-2 = 3)	**AND**			
		(Attribute-3 = 1)	**AND**			
		(Attribute-6 = 1)	**THEN**	(0)	0.04	1.00
2	**ELSE IF**	(Attribute-2 = 1)	**AND**			
		(Attribute-3 = 1)	**AND**			
		(Attribute-5 = 4)	**AND**			
		(Attribute-6 = 1)	**THEN**	(0)	0.02	1.00
3	**ELSE IF**	(Attribute-1 = 3)	**AND**			
		(Attribute-3 = 2)	**AND**			
		(Attribute-4 = 2)	**AND**			
		(Attribute-6 = 2)	**THEN**	(0)	0.03	0.92
4	**ELSE IF**	(Attribute-1 = 1)	**AND**			
		(Attribute-3 = 1)	**AND**			
		(Attribute-6 = 1)	**THEN**	(0)	0.05	0.88
5	**ELSE IF**	(Attribute-6 = 2)	**THEN**	(0)	0.32	0.58
6	**ELSE IF**	(Attribute-6 = 1)	**THEN**	(0)	0.53	0.53
7	**Else**			(1)	–	–
					Cov.	**Acc.**
					100.0	75.00

Table 15 Rules generated by μ^{-}Ant-Miner¬ for `Monk's` dataset

					Supp.	Conf.
1	**IF**	(Attribute-4 **NOT**= 1)	**THEN**	(0)	0.62	0.62
2	**Else**			(1)	–	–
					Cov.	**Acc.**
					100.0	64.28

time with prior discretization of the dataset (using the C4.5-Disc algorithm, fold by fold, as described in Sect. 9.1). As indicated in Table 3, we use JRip_{pd}, PART_{pd}, and C4.5r_{pd} to refer to each of the three algorithms, respectively, coupled with prior discretization of the dataset. Of course, for datasets without continuous attributes, the three variant algorithms JRip_{pd}, PART_{pd}, and C4.5r_{pd} are the same as the three standard algorithms JRip, PART, and C4.5r,

Table 16 Predictive accuracy results (%) for the conventional rule induction algorithms. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	aud	bal	bcw	car	cmc	c-a	c-g	drm	gls	hay	hrt	ion
JRip	71.45 ± 3.9	75.37 ± 2.4	**93.22** ± 2.4	87.60 ± 2.2	**50.90** ± 4.8	**83.41** ± 3.5	70.48 ± 1.5	**93.14** ± 2.9	**65.70** ± 4.9	**72.62** ± 5.2	**54.41** ± 3.9	86.30 ± 3.1
$JRip_{pd}$			**94.12** ± 2.6		48.87 ± 4.1	**82.40** ± 4.2	**72.62** ± 2.2	**92.72** ± 3.4	**65.51** ± 8.3		**55.82** ± 5.6	**88.20** ± 3.7
PART	**78.20** ± 3.7	**78.49** ± 2.9	93.23 ± 1.4	**94.56** ± 1.3	**50.89** ± 3.8	**83.52** ± 3.6	70.07 ± 1.6	**94.31** ± 2.9	**67.42** ± 3.0	63.61 ± 6.0	**54.91** ± 3.1	**88.25** ± 2.5
$PART_{pd}$			**94.40** ± 2.2		**49.82** ± 4.1	**82.52** ± 4.7	**72.70** ± 2.8	**92.40** ± 3.4	**68.31** ± 7.3		**54.91** ± 5.2	**86.43** ± 4.2
C4.5r	**83.34** ± 7.9	**81.84** ± 6.0	92.34 ± 3.0	**94.21** ± 2.7	**53.64** ± 4.1	**82.96** ± 5.3	**70.80** ± 2.8	**95.69** ± 3.4	**65.56** ± 6.8	**75.36** ± 10.8	**56.06** ± 9.1	**86.77** ± 5.6
$C4.5r_{pd}$			**95.17** ± 2.2		**49.73** ± 4.8	**84.13** ± 4.6	**72.30** ± 3.9	**94.85** ± 3.3	**66.11** ± 11.3		**56.77** ± 8.8	**90.60** ± 4.8

	irs	mon	msh	pop	soy	spt	tae	ttt	vot	win	zoo
JRip	**93.93** ± 2.2	**59.80** ± 2.3	**100.00** ± 0.0	**69.33** ± 3.9	**84.05** ± 0.1	**79.26** ± 2.7	**40.78** ± 8.1	**97.54** ± 1.7	**93.66** ± 1.4	**92.59** ± 2.2	88.70 ± 3.7
$JRip_{pd}$	**93.85** ± 4.5									**92.50** ± 3.1	
PART	**94.20** ± 3.1	**60.95** ± 3.2	**100.00** ± 0.0	62.66 ± 6.9	**85.98** ± 4.4	78.06 ± 2.4	**44.80** ± 9.7	93.60 ± 2.1	**94.51** ± 1.2	**92.07** ± 3.4	93.40 ± 3.4
$PART_{pd}$	**94.66** ± 5.5									**90.72** ± 4.3	
C4.5r	**94.66** ± 5.3	55.27 ± 4.3	**100.00** ± 0.0	**61.25** ± 19.9	**85.54** ± 6.3	**81.87** ± 3.6	**45.01** ± 10.7	**98.31** ± 1.9	**95.47** ± 2.8	**93.76** ± 7.2	**100.00** ± 0.0
$C4.5r_{pd}$	**94.67** ± 7.6									**91.90** ± 4.2	

respectively, and therefore, results are shown for them in Tables 16–17 only for datasets with continuous attributes.

The format of Tables 16–17 is similar to that of Tables 5–8. For each dataset, each table shows the mean and standard deviation ($^{mean}_{\pm\ stdv.}$) of the measure related to the table for each of the used algorithms. In addition, an entry is underlined if, for the corresponding dataset, the value obtained by the corresponding algorithm is the best (highest in accuracy or lowest in model size). Further, a value is shown in boldface if it differs from the underlined value by less than the average of the two standard deviations.

Table 18 shows the average accuracy ranking of the six rule induction algorithms across the 23 datasets, with the best average ranking shown in boldface. To compute this average ranking, we first compute the rank of each of the six algorithms for each dataset, then com-

Table 17 Model size results for the conventional rule induction algorithms. An entry is *underlined* if the value obtained by the corresponding algorithm is the best among all values achieved by the 12 considered versions. A value is shown in *boldface* if the difference between it and the best value is less than the average of the two standard deviations

	aud	bal	bcw	car	cmc	c-a	c-g	drm	gls	hay	hrt	ion
JRip	**15.08** ± *1.0*	**12.18** ± *1.3*	**5.30** ± *0.8*	**37.89** ± *2.4*	**5.90** ± *1.5*	30.10 ± *0.6*	**4.20** ± *0.4*	8.70 ± *0.8*	**7.60** ± *0.9*	**6.91** ± *0.6*	**3.40** ± *1.3*	**6.50** ± *1.6*
$JRip_{pd}$			8.40 ± *1.8*		**4.80** ± *1.6*	28.60 ± *1.5*	6.20 ± *2.2*	8.80 ± *0.7*	**8.20** ± *1.4*		**4.20** ± *0.6*	8.30 ± *1.4*
PART	19.67 ± *1.8*	33.14 ± *1.9*	7.90 ± *0.6*	56.52 ± *3.4*	163.60 ± *3.0*	29.70 ± *1.6*	69.20 ± *2.0*	8.90 ± *0.5*	15.20 ± *0.4*	11.32 ± *1.5*	40.50 ± *2.7*	8.50 ± *1.3*
$PART_{pd}$			10.10 ± *0.8*		94.50 ± *8.9*	25.30 ± *2.9*	82.60 ± *6.4*	**7.50** ± *0.5*	13.50 ± *1.2*		52.60 ± *2.6*	12.20 ± *1.7*
C4.5r	23.20 ± *1.1*	39.70 ± *2.6*	8.90 ± *1.0*	79.00 ± *2.6*	37.40 ± *6.8*	15.00 ± *2.0*	22.50 ± *5.0*	9.10 ± *0.7*	15.30 ± *1.9*	11.00 ± *0.7*	15.50 ± *2.1*	9.30 ± *1.3*
$C4.5r_{pd}$			11.00 ± *1.7*		25.30 ± *3.1*	**11.30** ± *2.0*	35.90 ± *4.4*	9.00 ± *0.8*	13.90 ± *1.4*		15.70 ± *2.5*	15.30 ± *1.6*

	irs	mon	msh	pop	soy	spt	tae	ttt	vot	win	zoo
JRip	**3.60** ± *0.3*	**3.12** ± *0.9*	**8.70** ± *0.6*	**2.33** ± *0.5*	23.20 ± *1.3*	**8.73** ± *1.3*	**6.29** ± *1.3*	**10.33** ± *1.4*	**3.70** ± *0.3*	**4.10** ± *0.2*	**7.43** ± *0.8*
$JRip_{pd}$	**3.60** ± *0.3*									6.10 ± *0.4*	
PART	4.50 ± *0.8*	37.75 ± *3.5*	11.57 ± *1.8*	8.40 ± *1.4*	29.39 ± *1.4*	34.53 ± *5.2*	16.74 ± *2.3*	38.85 ± *4.7*	6.60 ± *0.4*	4.40 ± *0.2*	**7.65** ± *0.5*
$PART_{pd}$	5.00 ± *0.0*									**4.40** ± *1.3*	
C4.5r	5.00 ± *0.0*	14.30 ± *2.6*	18.00 ± *0.0*	4.90 ± *1.3*	**14.46** ± *6.3*	32.10 ± *1.8*	13.60 ± *1.1*	24.70 ± *4.7*	7.30 ± *0.7*	5.40 ± *0.5*	9.70 ± *0.5*
$C4.5r_{pd}$	4.60 ± *0.5*									9.80 ± *1.0*	

pute the average across all datasets (thus, the average rank will range between 1 and 6). For each of the variant algorithms $JRip_{pd}$, $PART_{pd}$, and $C4.5r_{pd}$, respectively, we use the results for the corresponding standard algorithm, JRip, PART, and C4.5r, respectively, for the datasets without continuous attributes. We can see from the table that $C4.5r_{pd}$ has the best accuracy rank of the evaluated conventional rule induction algorithms.

We would like to determine if there is a statistically significant difference in accuracy between the best-performing Ant-Miner extension, identified in Table 9 to be $\mu\psi$ Ant-Miner, and the best-performing conventional rule induction algorithm, identified in Table 18 to be $C4.5r_{pd}$. The results for the two algorithms, $\mu\psi$ Ant-Miner and $C4.5_{pd}$, were obtained using 10-fold cross-validation using the same fold partitioning. Consequently, for a given dataset, the results for each fold, for the two algorithms, can be considered a matched pair.

Table 18 Accuracy ranking of the conventional rule induction algorithms. The best average ranking is shown in *boldface*

Algorithm	Predictive accuracy
JRip	3.96
JRip_{pd}	4.00
PART	3.09
PART_{pd}	3.17
C4.5r	2.39
C4.5r_{pd}	**2.09**

Table 19 Results of applying a two-tailed Wilcoxon matched-pairs test (at the 0.05 significance level) to the fold-by-fold accuracy results for each dataset. p-values lower than 0.05 are shown in *boldface*

Datasets	C4.5r_{pd}	$\mu\psi$AM	p	Remark
aud	83.34	82.92	0.878	No diff.
bal	81.84	79.73	0.221	No diff.
bcw	92.34	94.51	**0.010**	$\mu\psi$AM better
c-a	82.96	82.57	1.000	No diff.
car	94.21	94.83	0.386	No diff.
c-g	70.80	70.38	0.625	No diff.
cmc	53.64	47.76	**0.004**	C4.5r better
drm	95.69	97.40	**0.008**	$\mu\psi$AM better
gls	65.56	66.61	0.846	No diff.
hay	75.36	68.46	0.114	No diff.
hrt	56.06	58.66	0.193	No diff.
ion	86.77	86.95	0.492	No diff.
irs	94.66	94.61	0.695	No diff.
mon	55.27	63.37	**0.005**	$\mu\psi$AM better
msh	100.00	98.52	**0.005**	C4.5r better
pop	61.25	72.50	0.108	No diff.
soy	85.54	87.44	0.574	No diff.
spt	81.87	79.77	0.240	No diff.
tae	45.01	45.18	0.944	No diff.
ttt	98.31	98.76	0.799	No diff.
vot	95.47	95.22	0.878	No diff.
win	93.76	93.25	0.432	No diff.
zoo	100.00	99.75	0.157	No diff.

Therefore, for each individual dataset, we apply a two-tailed Wilcoxon matched-pairs test to the results of the 10 folds for the two algorithms under consideration ($\mu\psi$Ant-Miner and C4.5r_{pd}) to determine if there is a significant difference for that dataset.

The results of these Wilcoxon tests for each of the 23 datasets are shown in Table 19. We use the common convention that a p-value less than 0.05 indicates a statistically significant difference, and such p-values are shown in the table in boldface. Table 19 indicates that there is no statistically significant difference for 18 (out of 23) datasets, C4.5r_{pd} is significantly better in 2 datasets, and $\mu\psi$Ant-Miner is significantly better in 3 datasets.

These results do not suggest that there is an overall significant difference in accuracy between the two methods, however, for completeness, we apply a (one-tailed) sign test to the overall results (Demšar 2006). We count 3 wins for $\mu\psi$Ant-Miner, 2 wins for C4.5r$_{pd}$, and 18 ties. We divide the ties evenly between the two methods, giving $\mu\psi$Ant-Miner 12 wins out of 23 datasets. From the binomial distribution, this corresponds to a p-value of 0.500, which confirms that there is no overall significant difference in accuracy performance between the two methods. However, we note that in 22 out of the 23 datasets, $\mu\psi$Ant-Miner produced a smaller average number of rules than C4.5r$_{pd}$ (as can be seen from Tables 7–8 for $\mu\psi$Ant-Miner and Table 17 for C4.5r$_{pd}$).

11 Concluding remarks

This paper has proposed five extensions to the Ant-Miner classification rule discovery algorithm and reported the results of experiments with 12 variations of Ant-Miner, involving different combinations of those five extensions, across 23 public datasets often used as a benchmark in classification research.

Concerning the effectiveness of those extensions, in summary, the use of the multi-pheromone and quality contrast intensifier extensions combined together led to higher predictive accuracies in general, by comparison with several Ant-Miner variations that do not use such extensions. The use of the stubborn ants and ants with personality extensions combined together also led, in general, to predictive accuracies higher than or similar to the accuracies obtained by Ant-Miner variations that do not use these extensions. On the other hand, the logical negation extension improves the simplicity (reduces the size) of the discovered rule set, but sometimes at the expense of some loss in predictive accuracy.

Broadly speaking, out of the 12 Ant-Miner variations evaluated in our experiments, the most successful one was the variation $\mu\psi$Ant-Miner using four of the proposed extensions, namely multipheromone, quality contrast intensifier, stubborn ants, and ants with personality, but not using the logical negation extension. This Ant-Miner variation achieved the best predictive accuracy overall, with an average ranking of 2.5 (out of 12 Ant-Miner variations) across all datasets, and it still had an average performance in terms of the simplicity of the discovered rule set, with an average ranking of 7.3 (out of 12) across all datasets.

We also evaluated the performance of three state-of-the-art rule induction algorithms (Ripper, PART, and C4.5-Rules), and found C4.5-Rules to have the best accuracy performance of the three. We then compared $\mu\psi$Ant-Miner to C4.5-Rules and found that there was no statistically significant difference in accuracy between the two; however, $\mu\psi$Ant-Miner produced, in almost all cases, smaller classification models than C4.5-Rules.

In future work, we would like to explore combining some of our extensions, particularly the multipheromone and logical negation extensions, with the $\mathcal{MM}$AS-based AntMiner+. We would also like to extend the cAnt-Miner algorithm (the version of Ant-Miner that copes with continuous attributes) with the multipheromone system.

In addition, we would like to explore controlling the balance between a rule's coverage and its predictive accuracy in rule quality evaluation—especially when the logical negation operator is combined with the use of multipheromone. This can be achieved by employing different coefficients for the support and confidence components in (14), so that we would be able to decrease the emphasis on the rule coverage if the algorithm generates a small number of rules with reduced predictive accuracy, or enhance the simplicity of the discovered rule

set if a sufficient accuracy level is reached. This should allow the practitioner to have more control over the predictive accuracy and comprehensibility levels of the output classification model.

References

Abdelbar, A. M. (2008). Stubborn ants. In *Proceedings IEEE swarm intelligence symposium (SIS'08)* (pp. 1–5). New York: IEEE Press.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html. Retrieved July 2009.

Chan, A., & Freitas, A. (2005). A new classification-rule pruning procedure for an ant colony algorithm. In *Lecture notes in computer science: Vol. 3871. Artificial evolution (Proc. EA'05)* (pp. 25–36). Heidelberg: Springer.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of 12th international conference on machine learning* (pp. 115–123). San Francisco: Morgan Kaufmann.

Deb, K. (2009). *Multiobjective optimization using evolutionary algorithms*. New York: Wiley.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: overview and recent advances. In M. Gendreau & J. Y. Potvin (Eds.), *Handbook of metaheuristics* (pp. 227–263). New York: Springer.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In *Proceedings of the fifteenth international conference on machine* (pp. 144–151). San Francisco: Morgan Kaufmann.

Galea, M., & Shen, Q. (2006). Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In A. Agraham, C. Grosan, & V. Ramos (Eds.), *Swarm intelligence in data mining* (pp. 75–99). Berlin: Springer.

Jaiwei, H., & Kamber, M. (2006). *Data mining: concepts and techniques*. San Francisco: Morgan Kaufmann.

Kohavi, R., & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the second international conference on knowledge discovery and data mining* (pp. 114–119). Menlo Park: AAAI Press.

Liu, B., Abbass, H. A., & McKay, B. (2002). Density-based heuristic for rule discovery with ant-miner. In *Proceedings of the 6th Australasia–Japan joint workshop on intelligent and evolutionary systems (AJWIS2002)*, Canberra, Australia (pp. 180–184).

Liu, B., Abbass, H. A., & McKay, B. (2003). Classification rule discovery with ant colony optimization. In *Proceedings IEEE/WIC international conference on intelligent agent technology* (pp. 83–88). IEEE Comput. Soc.: Los Alamitos.

Martens, D., Backer, M. D., Haesen, R., Vanthienen, J., Snoeck, M., & Baesens, B. (2007). Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *11*, 651–665.

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

Otero, F., Freitas, A., & Johnson, C. G. (2008). *c*Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. In *Lecture notes in computer science: Vol. 5217. Ant colony optimization and swarm intelligence (Proc. ANTS'08)* (pp. 48–59). Heidelberg: Springer.

Otero, F., Freitas, A., & Johnson, C. G. (2009). Handling continuous attributes in ant colony classification algorithms. In *Proceedings IEEE symposium on computational intelligence and data mining (CIDM'09)* (pp. 225–231). New York: IEEE Press.

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

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

Salama, K. M., & Abdelbar, A. M. (2010). Extensions to the Ant-Miner classification rule discovery algorithm. In *Proceedings seventh international conference on swarm intelligence (ANTS'10)* (pp. 43–50). Berlin: Springer.

Smaldon, J., & Freitas, A. (2006). A new version of the Ant-Miner algorithm discovering unordered rule sets. In *Proceedings genetic and evolutionary computation conference (GECCO-2006)* (pp. 43–50). San Francisco: Morgan Kaufmann.

Stützle, T., Lòpez-Ibàñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., & Dorigo, M. (2010). Parameter adaptation in ant colony optimization. In Y. Hamadi, E. Monfroy, & F. Saubion (Eds.), *Autonomous search*. Heidelberg: Springer (in press).

Stützle, T., & Hoos, H. (2000). $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. *Future Generation Computer Systems*, *16*(8), 889–914.

Swaminathan, S. (2006). *Rule induction using ant colony optimization for mixed variable attributes*. Master's Thesis, Texas Tech. University, Lubbock, Texas.

Wang, Z., & Feng, B. (2004). Classification rule mining with an improved ant colony algorithm. In G. I. Webb & X. Yu (Eds.), *Lecture notes in computer science: Vol. 3339*. *Advances in artificial intelligence (Proc. AI'04)* (pp. 357–367). Heidelberg: Springer.

Witten, H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann.