Classification With Ant Colony Optimization

David Martens, Manu De Backer, Raf Haesen, *Student Member, IEEE*, Jan Vanthienen, Monique Snoeck, and Bart Baesens

***Abstract*—Ant colony optimization (ACO) can be applied to the data mining field to extract rule-based classifiers. The aim of this paper is twofold. On the one hand, we provide an overview of previous ant-based approaches to the classification task and compare them with state-of-the-art classification techniques, such as C4.5, RIPPER, and support vector machines in a benchmark study. On the other hand, a new ant-based classification technique is proposed, named AntMiner+. The key differences between the proposed AntMiner+ and previous AntMiner versions are the usage of the better performing $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, a clearly defined and augmented environment for the ants to walk through, with the inclusion of the class variable to handle multiclass problems, and the ability to include interval rules in the rule list. Furthermore, the commonly encountered problem in ACO of setting system parameters is dealt with in an automated, dynamic manner. Our benchmarking experiments show an AntMiner+ accuracy that is superior to that obtained by the other AntMiner versions, and competitive or better than the results achieved by the compared classification techniques.**

***Index Terms*—Ant colony optimization (ACO), classification, comprehensibility, $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, rule list.**

I. Introduction

CLASSIFICATION is one of the most frequently occurring tasks of human decision making. A classification problem encompasses the assignment of an object to a predefined class according to its characteristics [1], [2]. Many decision problems in a variety of domains, such as engineering, medical sciences, human sciences, and management science can be considered as classification problems. Popular examples are speech recognition, character recognition, medical diagnosis, bankruptcy prediction, and credit scoring.

Throughout the years, a myriad of techniques for classification has been proposed [3], [4], such as linear and logistic regression, decision trees and rules, k-nearest neighbor classifiers, neural networks, and support vector machines (SVMs). Various benchmarking studies indicate the success of the latter two nonlinear classification techniques [5], but their strength is also their main weakness: since the classifiers generated by neural networks and SVMs are described as complex mathematical functions, they are rather incomprehensible and opaque to humans. This opacity property prevents them from being used in many real-life applications where both accuracy and comprehensibility are required, such as medical diagnosis and credit risk evaluation [4], [6]. For example, in credit scoring, since the models concern key decisions of a financial institution, they need to be validated by a financial regulator. Transparency and comprehensibility are, therefore, of crucial importance. Similarly, classification models provided to physicians for medical diagnosis need to be validated [7], demanding the same clarity as for any domain that requires regulatory validation.

Our approach, AntMiner+, as well as the previously proposed AntMiner versions, takes into account the importance of both accuracy and comprehensibility and aims at inferring comprehensible rules using ant colony optimization (ACO) [8], [9].

The remainder of this paper is structured as follows. First, in Section II, the basics of ACO are shortly explained. Section III discusses the use of ACO for data mining, and more specifically, for the classification task. This is further elaborated on in Section IV, where we explain the workings of our approach: AntMiner+. In Section V, the setup and results of our benchmarking experiments on various publicly available data sets are discussed. Section VI concludes this paper and sets out some interesting issues for future research.

II. Ant Colony Optimization (ACO)

Swarm intelligence studies the collective behavior of unsophisticated agents that interact locally through their environment [10]. It is inspired by social insects, such as ants and termites, or other animal societies, such as fish schools and bird flocks. Although each individual has only limited capabilities, the complete swarm exhibits complex overall behavior. Therefore, the intelligent behavior can be seen as an emergent characteristic of the swarm. When focusing on ant colonies, it can be observed that ants communicate only in an indirect manner—through their environment—by depositing a substance called pheromone. Paths with higher pheromone levels will more likely be chosen and thus reinforced, while the pheromone intensity of paths that are not chosen is decreased by evaporation. This form of indirect communication is known as stigmergy, and provides the ant colony shortest-path finding capabilities.

ACO employs artificial ants that cooperate to find good solutions for discrete optimization problems [8]. These software agents mimic the foraging behavior of their biological counterparts in finding the shortest-path to the food source. The first algorithm following the principles of the ACO metaheuristic is the Ant System [11], [12], where ants iteratively construct solutions and add pheromone to the paths corresponding to these solutions. Path selection is a stochastic procedure based on two parameters, the pheromone and heuristic values. The pheromone value gives an indication of the number of ants that chose the trail recently, while the heuristic value is a problem dependent

Manuscript received November 17, 2005; revised June 26, 2006, October 13, 2006, and November 16, 2006. This work was supported in part by the Flemish Research Council (FWO) under Grant G.0615.05, and in part by Microsoft and KBC-Vlekho-K.U. Leuven Research Chairs.

D. Martens, M. De Backer, R. Haesen, J. Vanthienen, and M. Snoeck are with the Department of Decision Sciences, Katholieke Universiteit Leuven, Leuven 3000, Belgium (e-mail: David.Martens@econ.kuleuven.be).

B. Baesens is with the Department of Decision Sciences, Katholieke Universiteit Leuven, Leuven 3000, Belgium, and also with the School of Management, University of Southampton, Southampton SO17 1BJ, U.K.

Digital Object Identifier 10.1109/TEVC.2006.890229

quality measure. When an ant reaches a decision point, it is more likely to choose the trail with the higher pheromone and heuristic values. Once the ant arrives at its destination, the solution corresponding to the ant's followed path is evaluated and the pheromone value of the path is increased accordingly. Additionally, evaporation causes the pheromone level of all trails to diminish gradually. Hence, trails that are not reinforced gradually lose pheromone and will in turn have a lower probability of being chosen by subsequent ants.

To summarize, the design of an ACO algorithm implies the specification of the following aspects.

- An environment that represents the problem domain in such a way that it lends itself to incrementally building a solution to the problem.
- A problem dependent heuristic evaluation function (η), which provides a quality measurement for the different solution components.
- A pheromone updating rule, which takes into account the evaporation and reinforcement of the trails.
- A probabilistic transition rule based on the value of the heuristic function (η) and on the strength of the pheromone trail (τ) that determines the path taken by the ants.
- A clear specification of when the algorithm converges to a solution.

The performance of traditional ACO algorithms, however, is rather poor on large instance problems [13]. Stützle *et al.* [14] advocate that improved performance can be obtained by a stronger exploitation of the best solutions, combined with an effective mechanism for avoiding early search stagnation (the situation where all ants take the same path and thus generate the same solution). The authors propose a $\mathcal{MAX}$-$\mathcal{MIN}$ ant system that differs from the traditionally proposed Ant System in three aspects.

- After each iteration only the best ant is allowed to add pheromone to its trail. This allows for a better exploitation of the best solution found.
- The range of possible pheromone trails is limited to an interval $[\tau_{\min}, \tau_{\max}]$ so as to avoid early stagnation of the search.
- The initial pheromone value of each trail is set at $\tau_{\max}$. This determines a higher exploration at the beginning of the algorithm.

Note that other variants of the initial Ant System have been proposed as well, such as Ant Colony System [15], rank-based Ant System [16], and Elitist Ant System [12]. A detailed overview of these variants can be found in [8].

ACO has been applied to a variety of different problems [8], such as vehicle routing [17]–[19], scheduling [20], [21], timetabling [22], traveling salesman problem [12], [14], [23], and routing in packet-switched networks [24]. Recently, ants have also entered the data mining domain, addressing both the clustering [25], [26], and classification task [27]–[29], which is the topic of interest in this paper. The first application of ACO to the classification task is reported by Parpinelli *et al.* in [27] and was named AntMiner. Extensions were put forward by Liu *et al.* in AntMiner2 [28] and AntMiner3 [29]. The basic concepts of these classification techniques inspired by ants, will be discussed next. Our approach, AntMiner+, differs from these previous AntMiner versions in several ways, resulting in an improved performance, as explained in detail in Section IV.

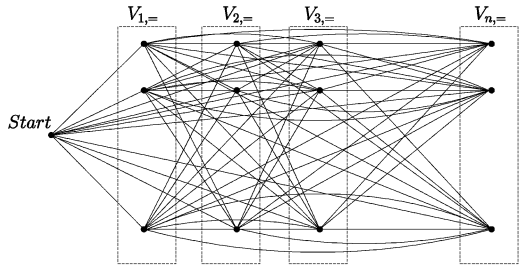

Fig. 1. Construction graph G of AntMiner.

III. AntMiner

The first application of ACO for the classification task was reported in [27] and [30], where the authors introduce the AntMiner algorithm for the discovery of classification rules. The aim is to extract from data simple rules of the form **if** *rule antecedent* **then** *rule consequent*, where the rule antecedent is a conjunction of terms. All attributes are assumed to be categorical; that is, the terms are of the form $Variable = Value$, e.g., $Sex = male$. The extracted classifier is an ordered rule list, meaning that the discovered rules are intended to be interpreted in a sequential order.

First of all, an environment in which the ants operate needs to be defined in a way that when the ants move, they incrementally construct a solution to the problem at hand, in this case, the classification problem. Although the AntMiner environment is originally not conceptually defined as such, we define the environment as a directed graph.[1] Given n variables $V_1, V_2, \ldots, V_n$ with variable V_i having values $Value_{i1}, Value_{i2}, \ldots, Value_{ip_i}$ (with p_i the number of possible values for variable V_i), the construction graph $G = (V, \Gamma)$ is defined as follows.

Vertices: For each variable V_i, we define a vertex $v_{i,j}$ for each of its values $Value_{ij}$, thus defining the complete set V of $\sum_{i=1}^{n} p_i$ vertices, plus one root vertex $Start$. The set of vertices for one variable is defined as a vertex group.

Correspondence: The correspondence Γ is defined as

$$\begin{aligned} \Gamma(Start) = v_{i,j} \quad & i = 1, 2, \ldots, n \\ & j = 1, 2, \ldots, p_i \\ \Gamma(v_{i,j}) = v_{k,l} \quad & k = 1, 2 \ldots, i-1, i+1, \ldots, n \\ & l = 1, 2, \ldots, p_k. \end{aligned}$$

This symmetric graph[2] is shown in Fig. 1. Note that the edges are bidirectional.

The complexity of the construction graph G, measured by the number of edges, is $O((avg^2)/2 \cdot n^2)$ with avg the average number of values per variable

$$n \cdot avg + (n-1) \cdot avg^2 + (n-2) \cdot avg^2 + \ldots + avg^2 \approx avg^2 \cdot \frac{n \cdot (n+1)}{2}.$$

[1]A directed graph G can be described by specifying the set V of vertices and a correspondence Γ which shows how the vertices are related to each other, with Γ being a mapping of V to V [31].

[2]A directed graph is said to be symmetric if, whenever an edge $(v_{i,j}, v_{k,l})$ is one of the edges of G, the opposite edge $(v_{k,l}, v_{i,j})$ is also an edge of G.

The workings of AntMiner are described next, where n denotes the total number of variables, p_i the number of values for variable V_i, and finally, x_i a binary variable which is set to one if variable V_i is not yet used by the current ant and to zero, otherwise. Each ant starts in the $Start$ vertex with an empty rule and chooses an edge to follow, e.g., to vertex $v_{i,k}$, implicitly adding the term $V_i = Value_{ik}$ to its rule. The edge to choose, and thus the term to add next, is dependent on the pheromone (τ_{ij}) and the heuristic value (η_{ij}) associated with each term $V_i = Value_{ij}$ and normalized over all possible terms. This probability P_{ij} of going to vertex $v_{i,j}$ is defined by (1)

$$P_{ij}(t) = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{k=1}^{n} x_k \sum_{l=1}^{p_k} (\tau_{kl}(t) \cdot \eta_{kl})}. \quad (1)$$

This choice is additionally constrained since each variable can occur at most once in a rule to avoid inconsistencies such as $Sex = male$ **and** $Sex = female$. The ant keeps adding terms to its partial rule until either all variables have been used in the rule or if adding any term would make the rule cover less cases than a user-defined minimum. The consequent of the rule is determined by the majority class of the training cases covered by the rule. Finally, the rule is pruned in order to remove irrelevant terms and the pheromone levels are adjusted, increasing the pheromone of the trail followed by the ant, and evaporating all others. Another ant starts with the newly updated pheromone trails to guide its search. This process is repeated until all ants have constructed a rule or when a series of consecutive ants construct the same rule. The best rule among these constructed rules is added to the list of discovered rules and the training cases covered by this rule are removed from the training set. This process is repeated until the number of uncovered training cases is lower than a user-defined threshold.

The heuristic value η_{ij} in AntMiner is defined as an information theoretic measure in terms of the entropy, as defined by (2) and (3) with d the number of classes, T_{ij} the set of remaining (not yet covered by the rule list) data instances having $V_i = Value_{ij}$, and $|T_{ij}|$ the size of the data set defined by T_{ij}. $P(w|T_{ij})$ is the probability of having class w for the data instances with $V_i = Value_{ij}$, and is measured as $(|T_{ij}\ \&\ CLASS = w|)/|T_{ij}|$

$$\eta_{ij} = \frac{\log_2(d) - Info(T_{ij})}{\sum_{i=1}^{n} x_i \sum_{j=1}^{p_i} (\log_2(d) - Info(T_{ij}))} \quad (2)$$

$$Info(T_{ij}) = -\sum_{w=1}^{d} (P(w\,|\,T_{ij}) \cdot \log_2 P(w\,|\,T_{ij})). \quad (3)$$

AntMiner2 [28], on the other hand, uses a simpler, though less accurate, density estimation equation as the heuristic value (4) with the assumption that the small induced errors are compensated by the pheromone level

$$\eta_{ij} = \frac{|T_{ij}\ \&\ CLASS = majority\ class(T_{ij})|}{|T_{ij}|}. \quad (4)$$

This makes AntMiner2 computationally less expensive without a significant degradation of the stated performance.

Two key changes have been proposed in AntMiner3 [29], resulting in a reported increased accuracy: a different update rule is used and more exploration is encouraged by means of a different transition rule that increases the probability of choosing terms not yet used in previously constructed rules. An overview of the definitions for heuristic and pheromone value functions, as well as the pheromone updating rule and initial pheromone value are provided later in this text, in Table II.

TABLE I
ANTMINER+

```
construct graph
while (not early stopping)
  initialize heuristics, pheromones and
    probabilities of edges
  while (not converged)
     create ants
     let ants run from source to sink
     evaporate pheromone on edges
     prune rule of best ant
     update path of best ant
     adjust pheromone levels if outside boundaries
     kill ants
     update probabilities of edges
  end
  extract rule
  flag data points covered by the extracted rule
end
evaluate performance on test set
```

IV. ANTMINER+

Based on the previous AntMiner versions, the main novelties implemented in AntMiner+ are as follows.

- Environment:
  - — The environment is defined as a directed acyclic graph (DAG), so that the ants can choose their paths more effectively.
  - — To allow for interval rules, the construction graph additionally exploits the difference between nominal and ordinal variables.
  - — Inclusion of the weight parameters for the pheromone and heuristic value in the construction graph.
- Implementation of the better performing $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System.
- The usage of more accurate class-specific heuristic values, with the exclusion of the majority class to be used as the final default class.
- Application of an early stopping criterion.

A. The AntMiner+ Algorithm

The main workings of AntMiner+ are described in pseudocode in Table I. First, a directed acyclic construction graph is created that acts as the ants' environment. All ants begin in the $Start$ vertex and walk through their environment to the $Stop$ vertex, gradually constructing a rule. Only the ant that describes the best rule will update the pheromone of its path. Evaporation decreases the pheromone of all edges. Supplementary modifications of the pheromone levels may be needed since the $\mathcal{MAX}$-$\mathcal{MIN}$ approach additionally requires the pheromone levels to lie within a given interval. Since the probabilities are the same for all ants in the same iteration, they are calculated only once, at the beginning of the iteration. Convergence occurs when all the edges of one path have a pheromone level $\tau_{\max}$ and all others edges have pheromone level $\tau_{\min}$. Next, the rule corresponding to the path with $\tau_{\max}$ is extracted and the training data covered by this rule is removed from the training set. This

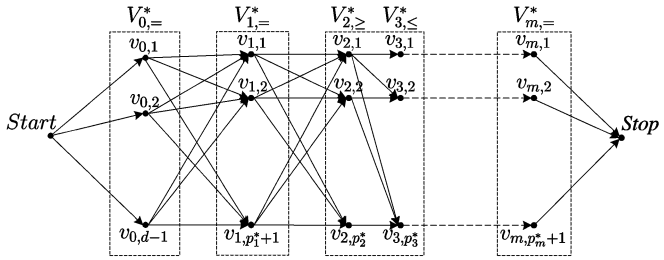

Fig. 2. Construction graph G^+ of AntMiner+.

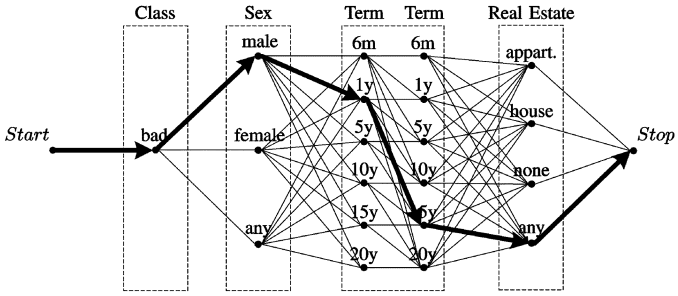

Fig. 3. Example of path described by an ant for the construction graph defined by AntMiner+.

iterative process will be repeated until early stopping occurs (cf. Section IV-E) or until none of the ants describe a rule that covers at least one training point. In the latter case, no rule can be extracted as all the paths have a quality of zero. This will typically be attributed to an abundance of noise in the remaining data, indicating that further rule induction is useless.

Details of AntMiner+ are provided in the following sections.

B. Construction Graph

A first modification to allow for intervals, such as $Age \in [20, 60]$, to be included in the rules, is the expansion of the set of n variables $\{V_1, V_2, \ldots, V_n\}$ into a set of m variables by taking the ordinal variables twice. This results in the new set of variables $\{V^*_1, V^*_2, \ldots, V^*_m\}$, with $m = |V_{Nom}| + 2 \cdot |V_{Ord}|$. The construction graph for AntMiner+ $G^+ = (V^+, \Gamma^+)$ is shown in Fig. 2 and defined as follows.

Vertices: First, the root and sink vertices $Start$ and $Stop$ are defined.

Second, just as for the AntMiner construction graph G, we define a vertex group for each of the variables V^*_i, i.e., a vertex $v_{i,j}$ is created for all values $Value_{ij}$ of the variables $V^*_i, i = 1, 2, \ldots, m$. The first group of vertices, defined for an ordinal variable V^*_i, are $v_{i,j} (j = 1, 2, \ldots, p^*_i)$, with p^*_i the number of values for variable V^*_i. This vertex group should be regarded as the lower bound for variable V^*_i, while the second vertex group $v_{i+1,j} (j = 1, 2, \ldots, p^*_{i+1})$ should be seen as the upper bound for V^*_{i+1}, which in this ordinal case is equal to V^*_i.

Since ants have to pass a vertex for each variable before reaching the final $Stop$ vertex, a dummy vertex v_{i,p^*_i+1} is defined for each nominal variable V^*_i. The value for this vertex is undetermined, implying that any ant choosing a dummy vertex does not make use of the variable V^*_i in its rule. For the ordinal variables, such a dummy vertex is unnecessary since an ant choosing the first vertex and then the last vertex [thus describing $V^*_i \geq \min(V^*_i)$ **and** $V^*_i \leq \max(V^*_i)$] also makes no use of the variable.

To allow the ants to choose the class for which to extract a rule, an extra vertex group is added that comes first in the construction graph. This is similar to considering the class variable as just one of the variables, and will be treated as such when calculating the heuristic values. Therefore, we introduce a new variable V^*_0 that corresponds to the class variable. To extract a rule list that is complete, such that all future data points can be classified, one class is not included in the construction graph and will be the predicted class for the final **else** clause. Hence, the class variable corresponds to a vertex group with $d - 1$ vertices. The class to exclude is taken to be the majority class since the smaller classes are usually the classes of interest: the bad credit customers, the malignant breast masses ... Additionally, this prevents a path with all dummy vertices, which corresponds to a rule predicting the majority class in all cases, to be chosen by the ants. As our experiments will show, excluding one class from the class vertex group results in an improved accuracy.

Correspondence: The correspondence Γ^+ is defined as

$$\begin{aligned}
&\Gamma^+(Start) = v_{0,j} && j = 1, 2, \ldots, d-1 \\
&\Gamma^+(v_{i,j}) = v_{i+1,k} && i = 0, 1, \ldots, m-1 \\
& && k = 1, 2, \ldots, p^*_{i+1} + 1 \quad \text{if } V^*_{i+1} \text{ nom} \\
& && k = 1, 2, \ldots, p^*_{i+1} \quad \text{if } V^*_{i+1} \text{ ord} \\
&\Gamma^+(v_{m,j}) = Stop && j = 1, 2, \ldots, p^*_m + 1 \quad \text{if } V^*_m \text{ nom} \\
& && j = 1, 2, \ldots, p^*_m \quad \text{if } V^*_m \text{ ord.}
\end{aligned}$$

To avoid that ants incorporate inconsistencies in their rule, such as $age \geq 30$ **and** $age \leq 20$, we further constrain the environment by removing the edge between any two vertices corresponding to the same ordinal variable, where the first vertex has a higher value than the second one. Therefore, in Fig. 2, among others, the edge between $v_{2,2}$ and $v_{3,1}$ is missing. This graph is a rooted DAG with a complexity of $O(m \cdot avg^2)$. Although m is a value slightly higher than n ($n \leq m \leq 2 \cdot n$), the complexity is clearly lower than the complexity of the construction graph G defined by AntMiner. Notice that on average, the ants in AntMiner+ need to choose among avg edges at each decision point, where for AntMiner they choose among $avg \cdot n$ edges. By reducing the choices without constraining the rule format, AntMiner+ ants can make their decisions more effectively.

A credit scoring example of the construction graph described so far is shown in Fig. 3. Assume we have three variables: sex of the applicant, term of the loan, and information concerning real estate property of the applicant. The dummy vertices for the nominal variables sex and real estate have the corresponding "any" value.

An ant that follows the path indicated in bold from $Start$ to $Stop$ describes the rule:

if $Sex = male$ **and** $Term \geq 1y$ **and** $Term \leq 15y$
then $customer = bad$

C. Edge Probabilities

The probability for an ant in vertex $v_{i-1,k}$ to choose the edge to vertex $v_{i,j}$ is defined by the pheromone value $(\tau_{(v_{i-1,k}, v_{i,j})})$

of the edge, the heuristic value ($\eta_{v_{i,j}}$) of vertex $v_{i,j}$, and normalized over all possible edge choices

$$P_{ij}(t) = \frac{[\tau_{(v_{i-1,k},v_{i,j})}(t)]^{\alpha} \cdot [\eta_{v_{i,j}}(t)]^{\beta}}{\sum_{l=1}^{p_i}[\tau_{(v_{i-1,k},v_{i,l})}(t)]^{\alpha} \cdot [\eta_{v_{i,l}}(t)]^{\beta}}. \quad (5)$$

For the AntMiner edge probabilities, we need to consider and normalize over all variables $V_i (i = 1, \ldots, n)$, while for AntMiner+, we only need to consider and normalize over the next variable V_i^*. The α and β are weight parameters that indicate the relative importance of the pheromone and heuristic value.

Heuristic Value: The heuristic value gives for each vertex in the construction graph a notion of its quality and importance in the problem domain. For the classification task, we define the importance of a vertex $v_{i,j}$, with value $Value_{ij}$ for variable V_i^*, by the fraction of training cases that are correctly covered (described) by this term, as defined by (6)

$$\eta_{v_{i,j}}(t) = \frac{|T_{ij} \;\&\; CLASS = class_{ant}|}{|T_{ij}|}. \quad (6)$$

Since the heuristic value is dependent on the class chosen by the ant (denoted as $class_{ant}$), each vertex has as many heuristic values as there are class values. This heuristic value for each possible class is more accurate than using just one heuristic for a vertex. The heuristic used in AntMiner2 and AntMiner3 only provides a good quality indication if a rule for the majority class is being extracted, if not, the heuristic is even misleading. As already mentioned, the heuristic function used in AntMiner is more accurate, but cannot provide a search direction that is as good as the one used by AntMiner+. Since AntMiner ants do not know for which class they are extracting a rule until the rule is constructed, guiding the ants with class-specific heuristics is useless. Since, on the other hand, AntMiner+ ants already know the class for which to extract a rule, such a class-specific and accurate heuristic function can be used.

Note that for dummy vertices, the same formula (6) is used to calculate the heuristic value. As T_{ij} stands for $V_i = any$ for a dummy vertex, the heuristic value is simply the percentage of uncovered training points that have the class $class_{ant}$. For the class vertices, a similar value is chosen, that is: the ratio of uncovered training points that have class equal to the class of the vertex. So, if 60% of the uncovered training data is of class 1, the heuristic value for the vertex will be 0.6.

We do not take into account the history of the ant in the heuristic value, that is, we ignore the number of data points already covered by the rule so far. This implies that the sequence of the variables in the construction graph is irrelevant and allows for distribution of the data. Often the data is collected and stored in different sites, interconnected through an intranet or Internet, where the traditional centralized data mining approach requires the transfer of all the data to one central database each time the data has been updated, AntMiner+ can simply send out the ants over the network. In this case, the construction graph is distributed, with the connections between the vertex groups possibly between geographically dispersed databases.

Pheromone Updating: Updating the pheromone trail of the environment of $\mathcal{MAX}$-$\mathcal{MIN}$ Ant Systems is accomplished in two phases: evaporation and reinforcement. Evaporation is accomplished by diminishing the pheromone level of each trail by a factor ρ. Typical values for this evaporation factor ρ lie in the range [0.8,0.99] [14]. Reinforcement of the pheromone trail is only applied to the best ant's path. The best ant can be chosen as either the iteration best ant, or the global best ant. Results described in the literature motivate our choice towards the iteration best ant [14]. This means that, taking into account the evaporation factor as well, the update rule for the best ant's path is described by (7), where the division by ten is a scaling factor that is needed such that both the pheromone and heuristic values lie within the range [0,1]

$$\tau_{(v_{i,j},v_{i+1,k})}(t+1) = \rho \cdot \tau_{(v_{i,j},v_{i+1,k})}(t) + \frac{Q^+_{best}}{10}. \quad (7)$$

Clearly, the reinforcement of the best ant's path should be proportional to the quality of the path Q^+_{best}, which we define as the sum of the *confidence* and the *coverage* of the corresponding rule. Confidence measures the fraction of remaining (not yet covered by any of the extracted rules) data points covered by the rule, that are correctly classified. The coverage gives an indication of the overall importance of the specific rule by measuring the number of correctly classified remaining data points over the total number of remaining data points. More formally, the pheromone amount to add to the path of the iteration best ant is given by the benefit of the path of the iteration best ant, as indicated by (8), with $rule_{ant}$, the rule antecedent (if part) being a conjunction of terms corresponding to the path chosen by the ant, $rule^c_{ant}$ the conjunction of $rule_{ant}$ with the class chosen by the ant, and Cov a binary variable expressing whether a data point is already covered by one of the extracted rules ($Cov = 1$) or not ($Cov = 0$). The number of remaining data points can, therefore, be expressed as $|Cov = 0|$

$$Q^+ = \underbrace{\frac{|rule^c_{ant}|}{|rule_{ant}|}}_{\text{confidence}} + \underbrace{\frac{|rule^c_{ant}|}{|Cov = 0|}}_{\text{coverage}}. \quad (8)$$

Weight Parameters: An issue typically encountered when applying ACO algorithms is the need to instantiate several system parameters, such as the number of ants, the evaporation factor, and the weight parameters α and β. In AntMiner, AntMiner2, and AntMiner3, the weight values are set to 1. AntMiner+, however, allows other values to be chosen and actually lets the ants themselves choose suitable values. This is done by introducing two new vertex groups in the construction graph: one for each weight parameter. We limit the values for the weight parameters to integers between 1 and 3, which typically provided values around $\alpha = 2$ and $\beta = 1$. Since we do not have a specific preference, the heuristic values for these vertices are all set to 1, so the search is only influenced by the pheromone levels.

The final construction graph, with the inclusion of weight parameters α and β, and the class variable V_0 is provided in Fig. 4.

D. Pruning

Rule pruning is a technique used to improve the generalization behavior of a rule-based classifier by removing irrelevant terms from a rule [32]. When all the ants have created their rule,

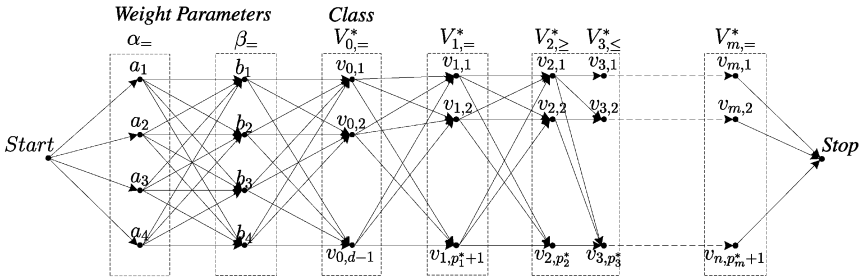

Fig. 4. Multiclass construction graph G^+ of AntMiner+ with the inclusion of weight parameters.

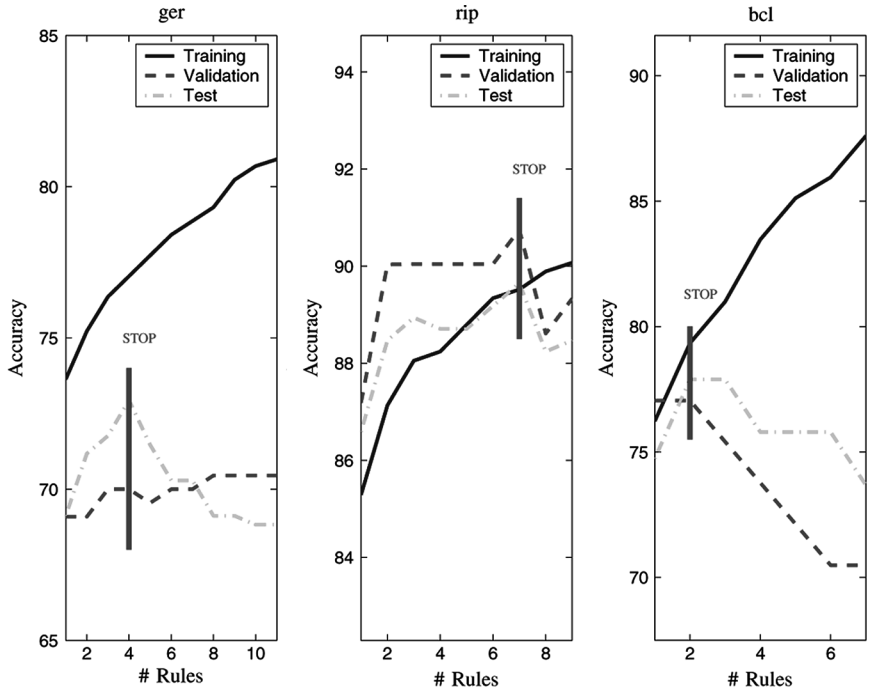

Fig. 5. Illustration of early stopping principle.

the rule of the best ant is selected for pruning. Suppose this rule has t terms, then all possible rules with one term less, thus $t-1$ rules are created and the corresponding confidence is calculated. The rule that experiences the highest confidence increase is selected for further pruning (thus considering rules with $t-2$ terms). This process is repeated until none of the rules with one term less have a confidence that is higher or the same. When a rule is pruned, a better rule is obtained, both in terms of accuracy (as only confidence is considered, and not the coverage) and comprehensibility (the rule is less complex since it consists of one term less). The path corresponding to this pruned path will be updated next, as we have described in Table I.

E. Early Stopping

To accomplish good generalization properties, and thus avoid the classifier from fitting the noise in the training data, a separate validation set can be monitored during the training session. When the error measure on the validation set starts to increase, training is stopped, thus effectively preventing the rule base from fitting the noise in the training data. Typical evolutions of the accuracy on the training, validation, and test set are shown in Fig. 5 for a selection of data sets. This stopping technique is known as *early stopping* and is used in AntMiner+. Note that this causes loss of data that cannot be used for constructing the rules and, hence, this method is less appropriate for small data sets. Alternatively, for such small data sets, the stopping criterion can be defined by a user-defined minimum for the percentage of data points covered by the rule list, or a maximum number of rules. As stated in the next section, we have opted for the first alternative.

Implementing the early stopping criterion involves the separation of the data set in three parts: a training set used by AntMiner+ to infer rules from, a validation set to implement the early stopping rule, and a test set to calculate an unbiased performance estimate. As a rule of thumb, 2/3 of the complete data set is typically used for training and validation and the remaining 1/3 for testing. Of the data points set aside for training and validation, 2/3 is training data and 1/3 validation data.

Our experiments will demonstrate that this stopping criterion results in an improved average accuracy over using a user-specified minimum threshold for data coverage.

TABLE II
OVERVIEW OF ANTMINER VERSIONS

	AntMiner	**AntMiner2**	**AntMiner3**	**AntMiner+**
η_{ij}	$\frac{log_2(k)-Info(T_{ij})}{\sum_{i=1}^{n} x_i \sum_{j=1}^{p_i}(log_2(k)-Info(T_{ij}))}$	$\frac{\lvert T_{ij} \ \& \ CLASS{=}maj.\ class(T_{ij})\rvert}{\lvert T_{ij}\rvert}$	$\frac{\lvert T_{ij} \ \& \ CLASS{=}maj.\ class(T_{ij})\rvert}{\lvert T_{ij}\rvert}$	$\frac{\lvert T_{ij} \ \& \ CLASS{=}class_{ant}\rvert}{\lvert T_{ij}\rvert}$
$\tau_{ij}(t{=}0)$	$\frac{1}{\sum_{i=1}^{n} p_i}$	$\frac{1}{\sum_{i=1}^{n} p_i}$	$\frac{1}{\sum_{i=1}^{n} p_i}$	τ_{max}
τ *update rule*	$\frac{\tau_{ij}(t)+\tau_{ij}(t)\cdot Q}{\sum_{\forall ij \in rule} \tau_{ij}}$	$\frac{\tau_{ij}(t)+\tau_{ij}(t)\cdot Q}{\sum_{\forall ij \in rule} \tau_{ij}}$	$(1-\rho')\cdot\tau_{ij}(t)+(1-\frac{1}{1+Q})\cdot\tau_{ij}(t)$	$\rho\cdot\tau_{(v_{i,j},v_{i+1,k})}(t)+\frac{Q^{+}_{best}}{10}$
$P_{ij}(t)$	$\frac{\tau_{ij}(t)\cdot\eta_{ij}}{\sum_{k=1}^{n} x_k \sum_{l=1}^{p_k}(\tau_{kl}(t)\cdot\eta_{kl})}$	$\frac{\tau_{ij}(t)\cdot\eta_{ij}}{\sum_{k=1}^{n} x_k \sum_{l=1}^{p_k}(\tau_{kl}(t)\cdot\eta_{kl})}$	$\frac{\tau_{ij}(t)\cdot\eta_{ij}}{\sum_{k=1}^{n} x_k \sum_{l=1}^{p_k}(\tau_{kl}(t)\cdot\eta_{kl})}$ if $q_1 \leq 0.4$, with $q_{1,2}$ random $\in [0,1]$ loop if $q_2 \leq \sum_{j \in J_i} P_{ij}$ then choose term$_{ij}$ end loop else choose term with max P_{ij}	$\frac{[\tau_{(v_{i-1,k},v_{i,j})}(t)]^{\alpha}\cdot[\eta_{v_{i,j}}(t)]^{\beta}}{\sum_{l=1}^{p_i}[\tau_{(v_{i-1,k},v_{i,l})}(t)]^{\alpha}\cdot[\eta_{v_{i,l}}(t)]^{\beta}}$
Pruning	Yes (based on Q)	Yes (based on Q)	Yes (based on Q)	Yes (based on $confidence$)

F. Curse of Dimensionality

The curse of dimensionality is a phenomenon where many types of data analysis become significantly harder as the data dimensionality increases [32], and arises from the fact that a large dimensional space with relatively few data points is sparse. For classification, this implies that insufficient data points are available to create a reliable classification model. In our case, none of the ants are able to extract a rule that covers at least one data point when dealing with such a high-dimensional data set.

To deal with this curse of dimensionality, we have two options: either lower the number of values per variable, or lower the number of variables. The first option is dealt with using discretization, the latter variable selection option with a χ^2-based filter [33] (see Appendix II). For AntMiner+, experimental investigation suggests to limit the number of variables to 20, with maximum 15 values each. These guidelines are set for data sets with about 1000 observations; for larger data sets these guidelines can of course be relaxed.

G. Overview AntMiner Versions

A complete overview of the differences between the proposed AntMiner versions is shown in Table II.[3] The table presents the used heuristic function η_{ij}, the initial pheromone value $\tau_{ij}(t=0)$, and the pheromone update rule, as well as how these are combined to determine the edge probability P_{ij}, and finally if pruning occurs.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

AntMiner+ has been applied to a wide range of publicly available data sets, corresponding to both binary and multiclass classification problems [34]. Training, validation, and test set are determined in the manner discussed before, although the early stopping rule (and thus validation set) is only used for data sets with more than 250 data points. The stopping criterion for data sets with less data points is determined by a user-defined minimum for the percentage of data points covered by the rule list, which we have set to 1%. To eliminate any chance of having unusually good or bad training and test sets, ten runs are conducted where the data is randomized before the training, validation, and test set are chosen.

The only parameters that need to be set are the number of ants and the evaporation factor ρ. The higher these values, better the results that can be expected, since more ants will automatically imply that more candidate rules are generated and increasing evaporation factor ρ will result in a slower convergence process. However, from a certain threshold on, a flat-maximum effect is reached: increasing the parameter(s) only results in more execution time and no significant increase in accuracy. The execution time scales linearly with the number of ants and exponentially with the evaporation factor ρ. Fig. 6 shows the influence of these parameters on both accuracy [left side of Fig. 6(a) and (b)] and execution time [right side of Fig. 6(a) and (b)] for the tic-tac-toe and wine data sets. The two-dimensional plots show the surface lines for a constant number of ants and constant evaporation factor ρ, with a wider line used for the experiments with 1000 ants and varying ρ, and the experiments with ρ set to 0.85 and varying number of ants. Although the shape of the plots varies across the different data sets, the choice of 1000 ants and an evaporation factor ρ of 0.85 (indicated with the white dot) provides an accuracy in the maximum plateau, while maintaining a reasonable execution time, and are therefore the parameter settings chosen for our experiments.

To compare the results of AntMiner+, a benchmarking study is performed that includes the previous ACO-based classification techniques and commonly used state-of-the-art classification techniques. Both AntMiner, as the adapted AntMiner2 and AntMiner3 versions, are implemented by the authors in Matlab®. C4.5 is the popular decision tree builder

[3]Note that *majority class* is abbreviated to *maj. class*.

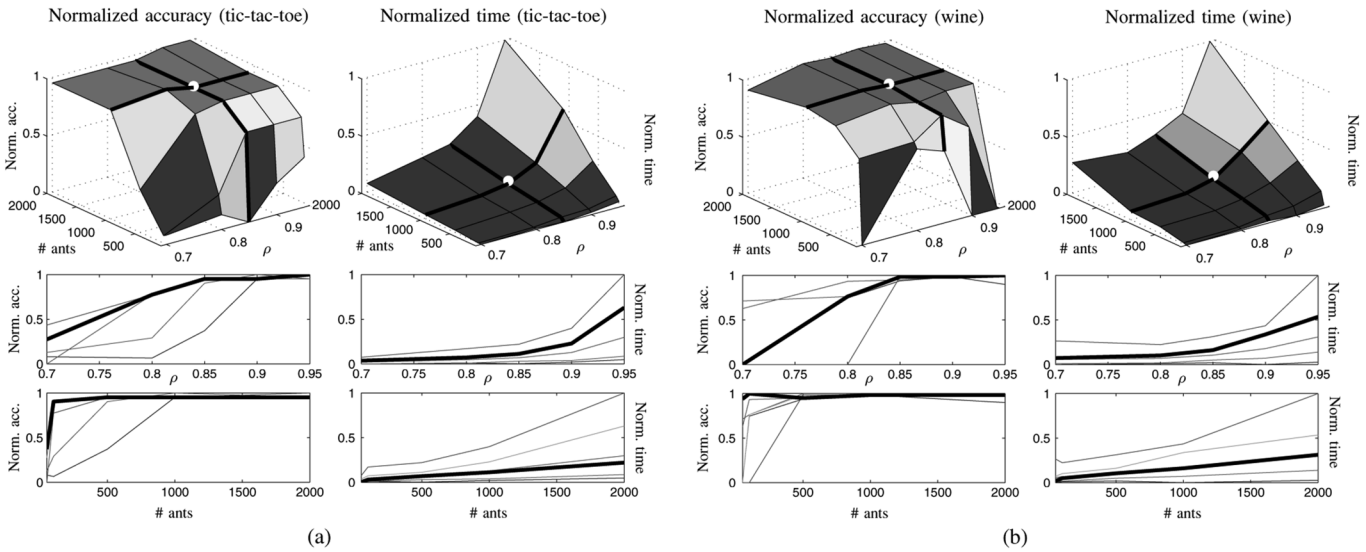

Fig. 6. Influence of the number of ants and evaporation factor ρ on accuracy and execution time for (a) tic-tac-toe and (b) wine data sets. The choice of 1000 ants and an evaporation factor of 0.85 is denoted with a white dot.

[35] where each leaf assigns class labels to observations. Each of these leaves can be represented by a rule and, therefore, C4.5 also builds comprehensible classifiers. Since C4.5 discovers unordered rule sets—whereas the AntMiner versions discover ordered rule lists—the RIPPER technique is also included in our experiments, since it also builds rule lists and, hence, has the same representation bias as the AntMiner versions. K-nearest neighbor classifiers (kNN) classify a data instance by considering only the k most similar data points in the training set. This technique is called lazy, since it does not involve creating a classification model, but rather defers the decisions on how to classify new data points beyond the training session. For both C4.5 and kNN, we used the Weka workbench [36], with the use of the standard pruning mechanism of C4.5, as implemented by the Weka workbench. Also, included is the commonly used logistic regression (logit) classifier and the nonlinear SVM classifier [37], with the nominal variables encoded with weights of evidence [33]. We report the results of the SVM with radial basis function kernel and hyperparameters set by a gridsearch procedure [38]. Since SVM and logit are binary classifiers, a set of classifiers of size equal to the number of classes are built using a *one versus all* scheme [39] for the multiclass data.

B. Datasets

AntMiner+ has been applied to a wide range of data sets. As we are interested in studying the algorithm's performance and also in situations which require comprehensibility of the generated classification models, some of the data sets are related to the credit scoring and medical diagnosis domains. Unless mentioned otherwise, all data sets are publicly available and can be retrieved from the UCI data repository [34].

Two credit scoring data sets are included in our experiments. The first one is the *australian* (aus) credit approval data set, which concerns credit card applications. The second credit scoring data set is the *german* (ger) data set, for which an extracted rule list is shown in Table III.

TABLE III
EXAMPLE RULE LIST ON GERMAN CREDIT SCORING DATA SET

if (Checking Account $<$ 200 DM **and** Duration $>$ 15 m **and**
Credit History $=$ no credits taken **and** Savings Account $<$ 1000 DM)
then class = bad
else if (Purpose $=$ new car/repairs/education/others **and**
Credit History $=$ no credits taken/all credits paid back duly at this bank **and**
Savings Account $<$ 1000 DM)
then class = bad
else if (Checking Account $<$ 0 DM **and**
Purpose $=$ furniture/domestic appliances/business **and**
Credit History $=$ no credits taken/all credits paid back duly at this bank **and**
Savings Account $<$ 500 DM)
then class = bad
else if (Checking Account $<$ 0 DM **and** Duration $>$ 15 m **and**
Credit History $=$ delay in paying off in the past **and**
Savings Account $<$ 500 DM)
then class = bad
else class = good

The breast cancer diagnosis data sets are *breast cancer Ljubljana* (bcl) and *breast cancer Wisconsin* (bcw). For these data sets, the task consists of classifying breast masses as being either benign or malignant. The *contraceptive method choice* (cmc) data set involves the prediction of the current contraceptive method choice (no use, long-term methods, or short-term methods) of a married woman based on her demographic and socioeconomic characteristics. An example rule list for this data set is shown in Table IV.

Several toy problems are included as well. The *iris* data set is a commonly used data set in the pattern recognition literature, where each of the three possible classes refer to a type of iris plant. The *tic-tac-toe* (ttt) data set encodes the complete set of possible board configurations at the end of tic-tac-toe games, where the target concept is "win for X." *Ripley*'s data set (rip) has two variables and two classes, which allows for visualization of the extracted rules. The classes are drawn from two normal distributions with a high degree of overlap [40]. We used a training set of size 250 and a test set of 1000 data

TABLE IV
EXAMPLE RULE LIST ON CONTRACEPTIVE METHOD CHOICE DATA SET

if (23 $\leq$ Age $\leq$ 35 **and** Education = low/rather low **and**
Husband Education = rather low/rather high **and** 1 $\leq$ Children $\leq$ 4 **and**
Work = not working **and** Husband Occupation = 3 **and**
Standard-of-Living = rather low/rather high)
then class = short term
else if (23 $\leq$ Age $\leq$ 35 **and** Education = rather low **and**
Husband Education = rather high/high **and** 1 $\leq$ Children $\leq$ 4 **and**
Religion = Islam **and** Husband Occupation = 2 **and**
Standard-of-Living = rather high **and** Media Exposure = good)
then class = short term
else if (36 $\leq$ Age $\leq$ 42 **and** Education = high **and**
2 $\leq$ Children $\leq$ 4 **and** Religion = non-Islam **and** Husband Occupation = 3 **and**
Standard-of-Living = high **and** Media Exposure = good)
then class = long term
else class = no-use

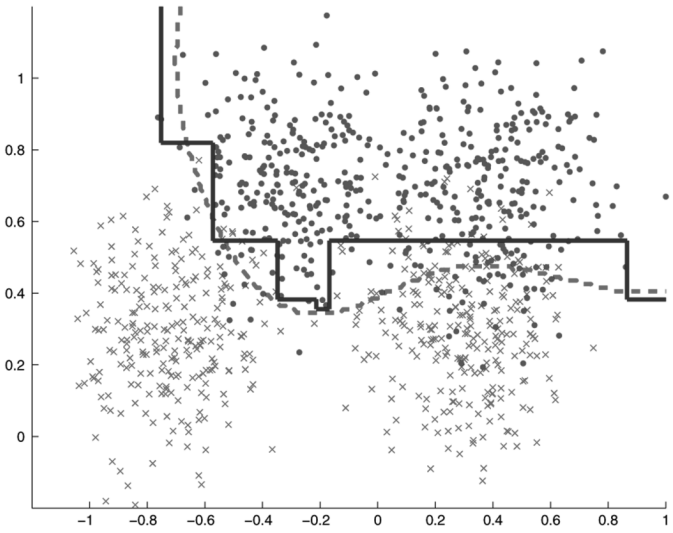

Fig. 7. AntMiner+ rules (—) and SVM decision boundary (--) for Ripley's data set.

points. Fig. 7 shows the decision boundary defined by the extracted AntMiner+ rules (accuracy 90.8%), as well as a SVM model (accuracy 91.4%).

The *teacher assistant evaluation* (tae) data set consists of evaluations of teaching assistant performance at the University of Wisconsin–Madison. The scores were divided into three roughly equal-sized categories to form the class variable. Each example in the *balance* (bal) data set is classified as having the balance scale tip to the right, tip to the left, or be balanced. The *car* evaluation data set evaluates cars in four classes according to the price and technical properties; and finally, the *wine* data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars.

C. Results

The results of our experiments on the binary classification problems are summarized in Table V, and the results for the multiclass classification problems are shown in Table VI. For each data set, the number of data instances (inst) and attributes (attr), as well as the average accuracy and, if applicable, the number of generated rules (#R) are displayed. The complexity of the rules is given by the average number of terms per rule (#T/R). The best average test set performance over the ten randomizations is underlined and denoted in bold face for each data set. We then use a paired t-test to test the performance differences. Performances that are not significantly different at the 5% level from the top performance with respect to a one-tailed paired t-test are tabulated in bold face. Statistically significant underperformances at the 1% level are emphasized in italics. Performances significantly different at the 5% level but not at the 1% level are reported in normal script. Since the observations of the randomizations are not independent, we remark that this standard t-test is used as a common heuristic to test the performance differences [41].

Table VII shows the results averaged over all the data sets, with the best performances shown in bold. Also provided is the average test set accuracy, the average number of rules and terms per rule, and the computational time required to obtain the results. Furthermore, we have assigned ranks to the techniques according to their accuracy on each data set, and the average ranking over the different data sets is also included in the table.

D. Discussion

The first observation that can be made is that our implementation of the previous AntMiner versions obtains very similar results, both in terms of accuracy as in terms of comprehensibility, with differences that are never statistically significant at the 5% level.

Considering the average accuracy and average ranking over all binary classification problems, AntMiner+ performs best with an average ranking of 3.2, even though it uses less training data than the other techniques to allow for a validation set, and thus early stopping. C4.5 and RIPPER do not perform significantly worse, except for the tic-tac-toe data set. The results obtained by 1NN are never the most accurate and are significantly worse than the best technique at a 1% level on 1/2 of the data sets. Logistic regression and SVM consistently perform well, with an average ranking of 3.5 and 3.8, respectively. Only on the tic-tac-toe data set, where the data set can be described by simple rules, do these mathematical classifiers perform significantly worse than AntMiner+.

When reviewing the results for the multiclass classification problems, SVM comes out as the most accurate technique, closely followed by AntMiner+ and logistic regression. 1NN comes in fourth, but it still performs significantly worse at a 1% level on 1/2 of the data sets. Since each multiclass problem requires a set of SVM or logit classifiers to be built in a *one versus all* setup, an ensemble of models is created, which can explain the added accuracy of these classifiers compared with the single model multiclass classification results.

Looking at the overall average accuracy and ranking in Table VII, SVM performs best, as could be expected from this nonlinear classification technique. Still the SVM models perform significantly worse on the bcl and tae data sets. This finding is typical for data mining, as no technique performs best on all data sets [5]. AntMiner+ achieves the second best average accuracy, whereas logistic regression achieves the second best average ranking. Considering accuracy and ranking only, AntMiner+ provides the best results of all included rule-based classifiers.

As motivated earlier in this paper, accuracy is not the only performance indicator: comprehensibility should be considered

TABLE V
AVERAGE OUT-OF-SAMPLE PERFORMANCE FOR BINARY CLASSIFICATION PROBLEMS

	aus				ger				bcw				bcl				rip				ttt			
	inst		attr		inst		attr		inst		attr		inst		attr		inst		attr		inst		attr	
	690		15		1000		19		683		9		277		9		1250		2		958		9	
Technique	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R
AntMiner+	**84.05**	0.46	2.3	1.5	*71.88*	0.37	5.7	4.0	**96.40**	0.54	3.4	4.3	**77.47**	2.44	3.4	3.2	**88.87**	1.72	5.2	1.9	**99.75**	0.77	9.0	3.0
AntMiner	**84.09**	1.65	6.5	2.3	*71.08*	3.01	9.2	1.2	*91.14*	2.44	9.7	1.1	**76.45**	4.63	6.7	1.6	*86.04*	1.68	19.5	1.0	*75.03*	2.26	4.2	1.1
AntMiner2	**84.30**	0.78	6.1	1.8	*70.66*	3.71	9.3	1.2	*91.54*	2.74	9.4	1.1	**75.91**	5.72	7.0	1.6	*85.66*	1.16	20.7	1.0	*71.13*	6.44	6.2	1.3
AntMiner3	**83.61**	2.86	7.1	2.4	*71.29*	2.89	9.2	1.2	*90.92*	2.95	9.4	1.1	**78.39**	3.64	7.0	1.6	*86.09*	1.36	20.9	1.0	*68.94*	4.15	7.3	1.3
RIPPER	**84.52**	1.47	5.3	2.3	*73.18*	1.63	3.2	4.8	**95.35**	2.51	4.6	2.4	75.22	3.72	2.3	2.0	**89.57**	1.88	5.1	2.5	*97.99*	0.45	10.1	3.3
C4.5	**84.82**	0.83	15.8		**74.20**	1.90	14.8		94.69	2.16	9.3		*74.56*	2.36	7.3		**88.87**	1.38	7.5		*83.79*	2.75	78.0	
1NN	*80.83*	2.17			**74.05**	1.96			**96.40**	1.49			*72.18*	2.47			**89.23**	1.36			*98.50*	0.75		
logit	**84.83**	2.51			**75.24**	0.81			**96.53**	1.00			**76.56**	3.95			*87.46*	0.93			*65.57*	2.23		
SVM	**85.22**	2.12			**73.68**	3.00			**92.81**	5.75			*75.27*	3.12			**89.57**	1.69			91.06	10.86		

TABLE VI
AVERAGE OUT-OF-SAMPLE PERFORMANCE FOR MULTICLASS CLASSIFICATION PROBLEMS

	cmc				iris				tae				bal				car				wine			
	inst		attr		inst		attr		inst		attr		inst		attr		inst		attr		inst		attr	
	1473		9		150		4		151		5		625		4		1728		6		178		13	
Technique	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R	Acc	σ_{Acc}	#R	#T/R
AntMiner+	*45.93*	0.63	4.5	5.7	**94.51**	0.83	4.2	1.2	**56.73**	3.17	32.9	1.6	*79.81*	1.29	9.2	3.1	*92.01*	1.72	26.6	5.5	**94.59**	1.74	4.5	2.5
AntMiner	*42.32*	2.63	14.3	1.8	*76.60*	3.89	5.6	1.0	*40.39*	7.17	8.9	1.2	*68.09*	3.46	13.7	1.0	*77.38*	2.02	13.5	1.3	*84.50*	5.67	5.6	1.0
AntMiner2	*41.49*	2.95	15.5	1.7	*81.80*	3.22	5.4	1.0	*43.73*	5.55	8.3	1.3	*66.94*	5.17	13.8	1.0	*77.93*	3.70	14.3	1.3	*85.33*	6.42	5.6	1.0
AntMiner3	*40.85*	2.47	15.3	1.7	*77.00*	3.80	5.5	1.0	*40.39*	6.68	8.3	1.2	*65.02*	5.87	13.8	1.0	*77.50*	4.16	15.0	1.3	*83.50*	5.00	5.5	1.1
RIPPER	**48.94**	2.83	4.8	3.2	**93.00**	1.94	3.2	1.2	*35.20*	6.41	5.1	1.3	*79.38*	3.86	9.1	3.2	*94.01*	1.95	17.2	5.5	90.68	4.88	4.9	2.1
C4.5	46.60	2.15	79.4		**93.80**	2.20	3.3		*47.20*	5.67	64.8		*77.11*	4.69	28.6		*96.61*	0.65	44.3		*89.83*	4.99	6.4	
1NN	*42.16*	1.28			*91.00*	2.16			50.20	7.86			*81.83*	2.86			*92.69*	1.01			**95.43**	1.79		
logit	**47.52**	1.80			**93.80**	2.90			**51.96**	6.74			*86.75*	1.66			*80.52*	2.37			**94.33**	2.63		
SVM	**48.55**	1.63			**94.40**	2.63			*48.43*	4.81			**91.58**	1.72			**97.71**	0.97			**94.83**	3.46		

TABLE VII
OVERALL AVERAGE OUT-OF-SAMPLE PERFORMANCES

Technique	Accuracy	Ranking	#R	#T/R	Time
AntMiner+	81.92	3.5	9.1	3.3	1776
AntMiner	72.76	7.1	9.8	1.3	390
AntMiner2	73.03	7.0	10.1	**1.3**	422
AntMiner3	71.96	7.5	10.4	1.3	867
RIPPER	79.75	4.3	**3.2**	2.8	0.3
C4.5	79.34	4.3	30.0		0.1
1NN	80.37	4.3			**0.0**
logit	78.42	3.3			0.3
SVM	**81.93**	**2.9**			941

as well. The nearest neighbor technique is lazy in the sense that there is no actual classifier. Comprehensibility of such decisions, based on the similarity with training data, is limited. Although the SVM models perform consistently well, the nonlinear, black-box nature of the generated classifiers makes them rather incomprehensible to humans. Logistic regression achieves good results as well but is troubled with similar opacity issues. The form of the SVM and logistic regression classifiers, respectively, are described by (9) and (10) and clearly indicate the opacity problem of these models

$$y_{SVM}(\mathbf{x}) = sign\left[\sum_{i=1}^{N} \alpha_i y_i \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_i\|_2^2}{\sigma^2}\right\} + b\right] \tag{9}$$

$$y_{logit}(\mathbf{x}) = 1/\left(1+\exp\left\{-\left(w_0+\mathbf{w}^T\mathbf{x}\right)\right\}\right). \tag{10}$$

The only techniques that deal with the comprehensibility aspect are the rule-based classifiers C4.5, RIPPER, and the AntMiner versions. RIPPER obtains the lowest number of rules, with an average of only 3.2 rules. Although the previous AntMiner versions extract more rules than RIPPER, they require less terms per rule than RIPPER. AntMiner+ extracts more rules and obtains more terms per rule when compared with RIPPER. C4.5 performs worst on this performance measure, with an average of 30 rules. An issue faced by C4.5 that can explain this higher number of rules is its greedy character, since every split made in the decision tree is irreversibly present in all leaves underneath.

When comparing the AntMiner versions, AntMiner+ achieves an average accuracy that is significantly better than the previous AntMiner versions on most of the data sets, and needs less rules to achieve this accuracy gain. On the other hand, the average number of terms per AntMiner+ rule is more than double the number of terms per rule for the other AntMiner versions.

The tradeoff between accuracy and comprehensibility is clearly present in our results: the most accurate results are obtained by incomprehensible nonlinear SVM models, whereas the most accurate rule-based classifier, AntMiner+, needs more rules and terms per rule to outperform the second best rule-based classifier, RIPPER.

The better results of AntMiner+ can be attributed to the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System which has a proven record of providing high performing solutions, but also to our construction graph which allows for an effective decision making process for the ants and the ability to include interval rules. Since the ants choose the rule consequent (the class) before the antecedent, the ants can direct their search towards the intended class. The inclusion of dummy variables and the distinction between ordinal and nominal variables do not only contribute to the comprehensibility, but also to the accuracy achieved, as demonstrated in the following section.

Some disadvantages still need to be addressed. AntMiner+ requires more computational time than the other techniques

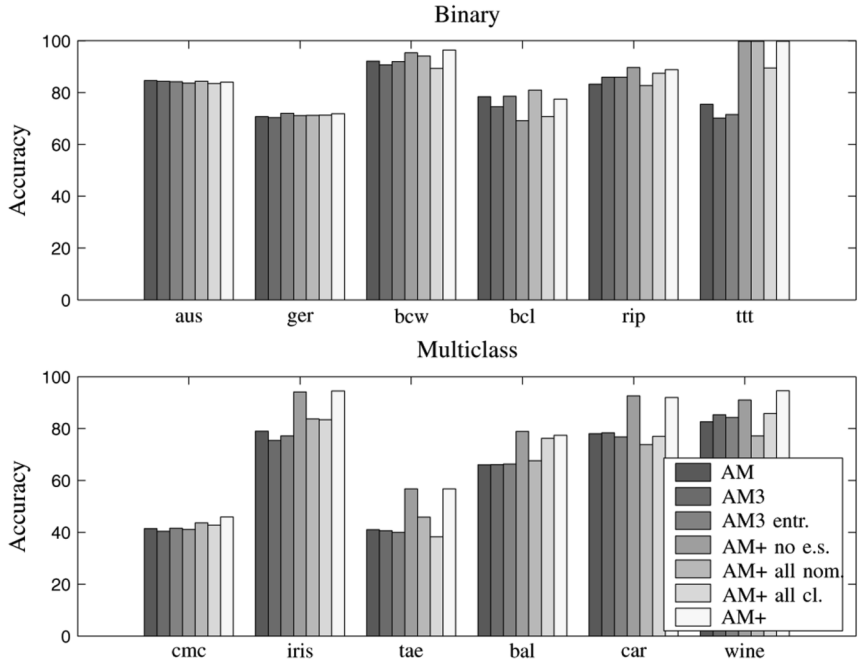

Fig. 8. Average accuracies for each data set over ten runs with the different AntMiner versions.

to achieve its results and, as all AntMiner versions, is only able to deal with categorical variables, while continuous variables must be discretisized. Parallelization of the inherently distributed AntMiner+ system could decrease the computation time needed, although one also needs to be aware of the minor importance of computational time for classification tasks such as credit scoring and medical diagnosis.

E. Experimental Investigation of Key Concepts

To explore the significance of the differences between AntMiner+ and previous AntMiner versions, we have studied several modifications of AntMiner+, of which the average results over ten runs for each data set are given in Fig. 8 and the overall averages in Fig. 9. Although the studied modifications might seem straightforward, one has to consider that many of the features are entangled with one another. It is the combined working of all the features of our approach, such as the construction graph, $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, heuristic and pheromone function, etc., that yield the reported results. For example, because of our directed acyclic construction graph, where no ant can stop before having passed all vertex groups, it is more likely for an ant to generate a rule with lower coverage than in the original AntMiner construction graph. By using the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, however, this effect is overcome and we actually obtain better results. In view of that, the accuracy drop accomplished by omitting some features of AntMiner+ cannot always be expected to be obtained when adding these to other techniques.

The considered modifications are described next.

1) $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System: Demonstrating the favorable effect of the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System in AntMiner+ needs no modification, as it is revealed by Fig. 6. As the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System only adds pheromone to the path of the best ant, using fewer ants results in a lowering accuracy.

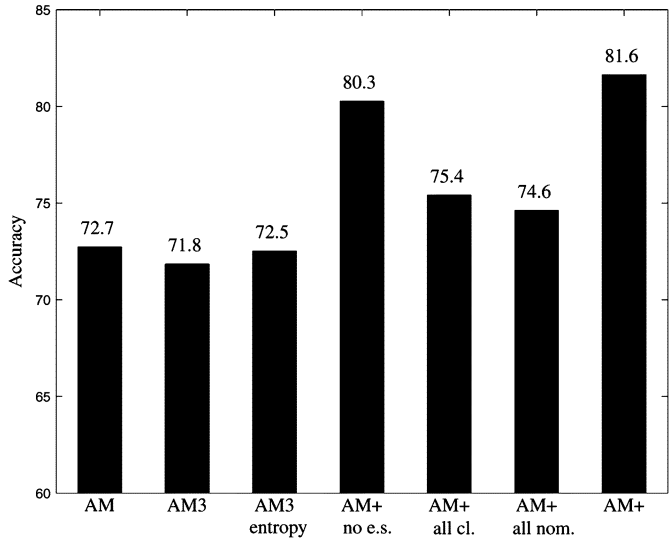

Fig. 9. Average accuracies over all data sets with the different AntMiner versions.

Adding pheromone to the path of each ant, as is done in the traditional Ant System, will therefore yield worse results.

2) AntMiner3 With the Entropy-Based Heuristic: The first reported results are the accuracy for AntMiner (AM) and AntMiner3 (AM3). To determine the importance of the more complex heuristic function used in AntMiner, we applied the entropy-based heuristic function to AntMiner3, providing the third results (AM3 *entr.*). Although the difference in accuracy is not significant, still an overall average improvement is reported. This observation, together with the positive results of AntMiner+ which uses more accurate heuristic values and the reported typical values of $\alpha = 2$ and $\beta = 1$, indicate the importance of providing a good heuristic function, as suggested before for other applications [8].

3) AntMiner+ Without Early Stopping: We replaced the early stopping criterion with a similar stopping criterion as the one used in previous AntMiner versions, i.e., a user-specified minimum threshold for coverage (AM+ *no e.s.*). As the sizes of the data sets vary greatly, instead of using a threshold for the number of uncovered data instances, we use a threshold for the percentage of uncovered data instances. Determining such a threshold that can be applied to all data sets is not an easy task as a proper value depends upon the noise present in the data: for some data sets (such as for bcl), the best accuracies are obtained when stopping if less than 25% of the data set remains uncovered, whereas for other data sets (such as for rip and iris), best accuracies are obtained when stopping at 5%, and even other data sets obtain best results at 1% (among others for car and wine). The 1% threshold is chosen, since for most data sets this is optimal, and since AntMiner+ stops anyway when no ant finds a rule that covers at least one data instance, so the algorithm can stop even though more than 1% of the data remains uncovered. The results show a rather slight decrease in average accuracy of 1.3%, in favor of the early stopping criterion.

4) AntMiner+ With the Inclusion of All Classes: A more significant change is observed for AntMiner+ when including all the classes in the class vertex (AM+ *all cl.*), meaning no class is set apart to act as default class in the final else clause. As for the previous AntMiner versions, rules are extracted for all possible classes, with the final class being the majority class of the remaining, uncovered training instances. When no class is set apart, the remaining data instances will be spread rather uniformly over all the classes, whereas setting a class apart will yield a clear majority of data entries with that class. So, assigning all remaining data instances to the majority class of the remaining instances will naturally yield a higher error when no class is set apart.

5) All Variables Declared Nominal: The final change considered is the declaration of all variables as nominal, as shown by our fifth adaptation (*AM+ all nom.*). Being unable to include interval rules results in an accuracy drop from 81.6% to 75.4%. As we can also distinguish between nominal and ordinal variables in C4.5 and RIPPER, we applied the same modification to those techniques, i.e., declaring all the variables as nominal. This resulted in an average accuracy decrease for C4.5 and RIPPER of 2.2% and 2.9%, respectively. These results clearly indicate the beneficial value of being able to include interval rules, in general, and its key role in AntMiner+.

VI. Conclusion and Future Research

In this paper, we have discussed the use of ACO for classification. By providing an appropriate environment, the ants choose their paths and implicitly construct a rule. One of the strengths of the ant-based approaches is that the results are comprehensible, as they are in a rule-based format. Such rule lists provide insight into the decision making, which is a key requirement in domains such as credit scoring and medical diagnosis.

The proposed AntMiner+ technique can handle both binary and multiclass classification problems and generates rule lists consisting of propositional and interval rules. Furthermore, the problem commonly encountered in ACO of setting system parameters α and β is dealt with in an automated, dynamic manner. Another advantage of ACO that comes out more clearly in our approach is the possibility to handle distributed environments. Since the AntMiner+ construction graph is defined as a sequence of vertex groups (of which the order is of no relevance), we are able to mine distribute databases.

TABLE VIII
Example Rule List on Breast Cancer Ljubljana Data Set With Inconsistency Underlined

if (menopause = ge40 **and** nodes-involved $\geq$ 3 **and** irradiat = no)
then class = recurrence
else if (tumor-size $\leq$ 44 **and** nodes-involved $\geq$= 3 **and**
node-caps = yes **and** degree-malignant $\geq$ 3)
then class = recurrence
else class = no-recurrence

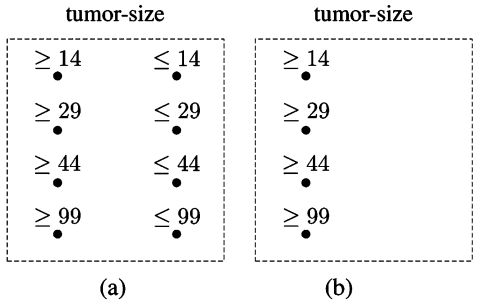

Fig. 10. Vertex groups for variable *tumor-size* (a) without and (b) with environment adjustment to reflect domain knowledge.

Our experiments show only little differences in the results obtained by the previous AntMiner versions, while the AntMiner+ technique achieves a significantly higher accuracy. Experimental investigation revealed the importance of the interval rules and accurate class-specific heuristic values for this performance gain. When compared with state-of-the-art classification techniques, AntMiner+ ranks at the absolute top when considering both accuracy and comprehensibility.

An issue faced by any rule-based classifier is that, although the classifier is comprehensible, it is not necessarily in line with existing domain knowledge [7]. It may well occur that data instances, that are very evident to classify by the domain expert, do not appear frequently enough in the data in order to be appropriately modeled by the rule induction technique. Hence, to be sure that the rules are intuitive and logical, expert knowledge needs to be incorporated. An example rule-set of such an unintuitive rule list, generated by AntMiner+, is provided in Table VIII. The underlined term is contradictory to medical knowledge suggesting that increasing tumor sizes result in higher probability of recurrence. As shown in Fig. 10, such domain knowledge can be included in AntMiner+ by changing the environment.[4] Since the ants extract rules for the recurrence class only, we can remove the second vertex group corresponding to the upper bound on the variable. Doing so ensures that the rules comply with the constraint required for the tumor size variable. Applying such constraints on relevant data sets to obtain accurate, comprehensible, and intuitive rule lists is surely an interesting topic for future research.

[4]The variable tumor-size actually has 12 possible values, which we limit to 4 in Fig. 10 to avoid overloading the figure.

TABLE IX
NOTATIONS

avg	Average number of values per variable
Cov	Binary variable expressing whether a data point is already covered by one of the extracted rules or not
d	Number of classes
m	Number of vertex groups in the AntMiner+ construction graph
n	Number of variables
T_{ij}	The set of remaining (not yet covered by the rule list) data instances having $V_i = Value_{ij}$
$\|x\|$	The size of the data set defined by x
p_i	Number of values for variable V_i
$P_{i,j}$	Probability for an ant to go to vertex $v_{i,j}$
Q	sensitivity · specificity
Q^+	confidence + coverage
$rule_{ant}$	The rule antecedent comprising of a conjunction of terms corresponding to the path chosen by the ant
$rule^c_{ant}$	The conjunction of $rule_{ant}$ with the class chosen by the ant, thus ($rule_{ant}$ & $CLASS = class_{ant}$)
$v_{i,j}$	Vertex for which variable V_i is equal to (or larger/smaller than) its j^{th} value
x_i	Set to 1 if variable V_i is not yet used by current ant, 0 otherwise
η	Heuristic value
$\eta_{v_{i,k}}$	Heuristic value for vertex $v_{i,k}$
τ	Pheromone level
$\tau_{(v_{i,j},v_{i+1,k})}$	Pheromone level on the edge going from vertex $v_{i,j}$ to vertex $v_{i+1,k}$

Finally, additional comprehensibility to the current AntMiner+ implementation can be provided by visualizing the extracted rule lists with decision tables [6] and decision diagrams [42], allowing for truly easy and user-friendly consultation in every day practices.

APPENDIX I
NOTATIONS

An overview of the notations are found in Table IX.

APPENDIX II
χ^2-BASED FILTER FOR VARIABLE SELECTION

In order to reduce the number of variables, a filter can be used that provides an indication of the predictiveness of each of the variables. By choosing those variables that are the most predictive, a reduced set of variables is obtained.

The χ^2 statistic can be used as a filter in the following manner [33]. We measure the observed frequencies of all possible combinations of values for class and variable, as shown in Table X. Based on this, we calculate the theoretical frequencies shown in Table XI, assuming complete independence between the variable and the class. The hypothesis of equal odds provides a χ^2 test statistic; the higher this value, the more the assumption of independence is invalid, and thus the more predictive the variable is. A detailed and theoretical framework is provided next.

TABLE X
OBSERVED FREQUENCIES

	$Class_1$	$Class_2$	$Class_3$	...	$Class_q$
$Value_1$	x_{11}	x_{12}	x_{13}		x_{1q}
$Value_2$	x_{21}	x_{22}	x_{23}		x_{2q}
...					
$Value_{p_i}$	x_{p_i1}	x_{p_i2}	x_{p_i3}		x_{p_iq}

TABLE XI
THEORETICAL FREQUENCIES

	$Class_1$	$Class_2$	$Class_3$	...	$Class_q$
$Value_1$	y_{11}	y_{12}	y_{13}		y_{1q}
$Value_2$	y_{21}	y_{22}	y_{23}		y_{2q}
...					
$Value_p$	y_{p1}	y_{p2}	y_{p3}		y_{pq}

From the observed frequencies x_{ij}, as defined by (11), the theoretical frequencies under the independence assumption y_{ij} can be constructed, as shown by (12) with N the total number of observations

$$x_{ij} = P(Var = Value_i \ \& \ Class = Class_j) \tag{11}$$

$$y_{ij} = P(Var = Value_i) \cdot P(Class = Class_j) = \frac{\sum_{k=1}^{q} x_{ik}}{N} \cdot \frac{\sum_{l=1}^{p_i} x_{lj}}{N} \cdot N. \tag{12}$$

Under independence, (13) would hold. The more x_{ij} deviates from y_{ij}, the less the assumption of independence is valid and, hence, the more predictive the variable is

$$x_{ij} = P(Var = Value_i \ \& \ Class = Class_j) = P(Var = Value_i) \cdot P(Class = Class_j) = y_{ij}. \tag{13}$$

The χ^2 test characteristic, as defined by (14) can be seen as a measure for the predictiveness: the higher its value, the more predictive the variable. This characteristic can be used as a filter by ranking all the variables with respect to their p-value and selecting the most predictive ones

$$\chi^2 = \sum_{i=1}^{p} \sum_{j=1}^{q} \frac{(y_{ij} - x_{ij})^2}{x_{ij}}. \tag{14}$$

ACKNOWLEDGMENT

The authors extend their gratitude to the (associate) editor and the anonymous reviewers, as their many constructive and detailed remarks certainly contributed much to the quality of this paper.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2000.

[2] D. J. Hand and S. D. Jacka, *Discrimination and Classification*. New York: Wiley, 1981.
[3] D. J. Hand, "Pattern detection and discovery," in *Pattern Detection and Discovery*, ser. Lecture Notes in Computer Science, D. J. Hand, N. Adams, and R. Bolton, Eds. Berlin, Germany: Springer-Verslag, 2002, vol. 2447, pp. 1–12.
[4] B. Baesens, "Developing intelligent systems for credit scoring using machine learning techniques," Ph.D. dissertation, K.U. Leuven, Leuven, Belgium, 2003.
[5] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. A. K. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," *J. Oper. Res. Soc.*, vol. 54, no. 6, pp. 627–635, 2003.
[6] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen, "Using neural network rule extraction and decision tables for credit-risk evaluation," *Manage. Sci.*, vol. 49, no. 3, pp. 312–329, 2003.
[7] M. Pazzani, S. Mani, and W. Shankle, "Acceptance by medical experts of rules generated by machine learning," *Methods of Inf. Med.*, vol. 40, no. 5, pp. 380–385, 2001.
[8] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.
[9] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 11–32.
[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford Univ. Press, 1999.
[11] M. Dorigo, V. Maniezzo, and A. Colorni, Positive feedback as a search strategy Dipartimento di Elettronica e Informatica, Politecnico di Milano, Milano, Italy, Tech. Rep. 91016, 1991.
[12] ——, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. Part B*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
[13] T. Stützle and H. H. Hoos, Improving the ant-system: A detailed report on the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system FG Intellektik, TU Darmstadt, Germany, Tech. Rep. AIDA 96-12, 1996.
[14] T. Stützle and H. H. Hoos, "$\mathcal{MAX}$-$\mathcal{MIN}$ ant system," *Future Generation Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.
[15] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[16] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system: A computational study," *Central Eur. J. Oper. Res. Econ.*, vol. 7, no. 1, pp. 25–38, 1999.
[17] A. Wade and S. Salhi, "An ant system algorithm for the mixed vehicle routing problem with backhauls," in *Metaheuristics: Computer Decision-Making*. Norwell, MA: Kluwer, 2004, pp. 699–719.
[18] B. Bullnheimer, R. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. Osman, and C. Roucairol, Eds. Norwell, MA: Kluwer, 1999, pp. 285–296.
[19] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Combinatorial Optim.*, vol. 10, no. 4, pp. 327–343, 2005.
[20] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job-shop scheduling," *J. Oper. Res., Stat., Comput. Sci.*, vol. 34, no. 1, pp. 39–53, 1994.
[21] C. Blum, "Beam-ACO—hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Comput. Oper. Res.*, vol. 32, no. 6, pp. 1565–1591, 2005.
[22] K. Socha, J. Knowles, and M. Sampels, "A $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for the university timetabling problem," in *Proc. 3rd Int. Workshop on Ant Algorithms*, M. Dorigo, G. Di Caro, and M. Sampels, Eds., Sep. 2002, vol. 2463, pp. 1–13, Lecture Notes in Computer Science.
[23] L. M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," in *Proc. 12th Int. Conf. Mach. Learn.*, A. Prieditis and S. Russell, Eds., Palo Alto, CA, 1995, pp. 252–260.
[24] G. Di Caro and M. Dorigo, "Antnet: Distributed stigmergetic control for communications networks," in *J. Artif. Intell. Res.*, 1998, vol. 9, pp. 317–365.
[25] A. Abraham and V. Ramos, "Web usage mining using artificial ant colony clustering," in *The Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press, 2003, pp. 1384–1391.
[26] J. Handl, J. Knowles, and M. Dorigo, "Ant-based clustering and topographic mapping," *Artif. Life*, vol. 12, no. 1, pp. 35–61, 2006.
[27] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "An ant colony based system for data mining: Applications to medical data," in *Proc. Genetic and Evol. Comput. Conf.*, 2001, pp. 791–797.
[28] B. Liu, H. A. Abbass, and B. McKay, "Density-based heuristic for rule discovery with ant-miner," in *Proc. 6th Australasia-Japan Joint Workshop on Intell. Evol. Syst.*, Canberra, Australia, 2002, pp. 180–184.
[29] ——, "Classification rule discovery with ant colony optimization," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.*, 2003, pp. 83–88.
[30] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 321–332, Aug. 2002.
[31] N. Christofides, *Graph Theory. An Algorithmic Approach*. New York: Academic Press, 1975.
[32] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA: Addison Wesley, 2005.
[33] L. Thomas, D. Edelman, and J. Crook, Eds., *Credit Scoring and Its Applications*. Philadelphia, PA: SIAM, 2002.
[34] S. Hettich and S. D. Bay, *The UCI KDD Archive*. Irvine, CA: Dept. Inf. Comput. Sci., Univ. California, 1996 [Online]. Available: http://kdd.ics.uci.edu,
[35] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[36] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Mateo, CA: Morgan Kaufmann, 2000.
[37] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[38] T. Van Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle, "Benchmarking least squares support vector machine classifiers," *Mach. Learn.*, vol. 54, no. 1, pp. 5–32, 2004.
[39] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[40] B. D. Ripley, "Neural networks and related methods for classification," *J. Royal Stat. Soc. B*, vol. 56, pp. 409–456, 1994.
[41] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.
[42] C. Mues, B. Baesens, C. M. Files, and J. Vanthienen, "Decision diagrams in machine learning: An empirical study on real-life credit-risk data," *Expert Syst. With Appl.*, vol. 27, no. 2, pp. 257–264, 2004.

David Martens received the M.S. degree in civil engineering from the Department of Computer Science, Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium, in 2003, and the Master of Business Administration degree from Reims Management School, Reims, France, in 2005. Since 2005, he has been working towards the Ph.D. degree at the Department of Decision Sciences and Information Management, K.U. Leuven.

His research is mainly focused on the development of comprehensible data mining techniques, with main application of the building of Basel II-compliant credit scoring systems.

Manu De Backer receivedthe M.S. degree in applied economic sciences from the Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium, in 2001. Currently, he is working towards the Ph.D. degree at the Department of Decision Sciences and Information Management, K.U. Leuven.

His research interests include business process modeling and verification, object-oriented conceptual modeling, and data mining.

Raf Haesen (S'05) received the M.S. degree in civil engineering from the Department Computer Science, Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium. He is currently working towards the Ph.D. degree at the KBC-Vlekho–K.U. Leuven Research Center.

His research interests are object-oriented conceptual modeling, service and component-based architecture, model driven development, and data mining.

Jan Vanthienen received the M.Sc. and Ph.D. degrees in applied economic sciences from the Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium.

He is a Professor at the Department of Decision Sciences and Information Management, K.U. Leuven. His main research themes are business rules and information management.

Monique Snoeck received the Ph.D. degree in computer science from the Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium.

She is a Full Professor in the Management Information Systems Group, Department of Applied Economic Sciences, K.U. Leuven. Her research focuses on object oriented conceptual modeling, software architecture, and software quality, with a thesis that lays the formal foundations of the object-oriented business modelling method MERODE.

Bart Baesens received the M.Sc. and Ph.D. degrees in applied economic sciences from the Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium, in 1998 and 2003, respectively.

He is currently working as an Assistant Professor at K.U. Leuven and as a Lecturer at the University of Southampton, Southampton, U.K. His research interests include classification, rule extraction, neural networks, support vector machines, data mining, and credit scoring.