Neural Comput & Applic (2012) 21:219–235
DOI 10.1007/s00521-010-0490-5

SWARM INTELLIGENCE

A correlation-based ant miner for classification rule discovery

Abdul Rauf Baig · Waseem Shahzad

Received: 13 May 2009 / Accepted: 10 November 2010 / Published online: 8 December 2010

Abstract In recent years, a few sequential covering algorithms for classification rule discovery based on the ant colony optimization meta-heuristic (ACO) have been proposed. This paper proposes a new ACO-based classification algorithm called AntMiner-C. Its main feature is a heuristic function based on the correlation among the attributes. Other highlights include the manner in which class labels are assigned to the rules prior to their discovery, a strategy for dynamically stopping the addition of terms in a rule's antecedent part, and a strategy for pruning redundant rules from the rule set. We study the performance of our proposed approach for twelve commonly used data sets and compare it with the original AntMiner algorithm, decision tree builder C4.5, Ripper, logistic regression technique, and a SVM. Experimental results show that the accuracy rate obtained by AntMiner-C is better than that of the compared algorithms. However, the average number of rules and average terms per rule are higher.

Keywords Ant colony optimization (ACO) · Classification rules · Data mining · Swarm intelligence

1 Introduction

In this paper, our focus is on the discovery of rules for the classification task using supervised training data. Many classification algorithms already exist, such as decision trees, neural networks, k-nearest neighbor classifiers, and support vector machines. Some of them (e.g., neural networks and support vector machines) are incomprehensible and opaque to humans, while others are comprehensible (e.g., decision tress). In many applications, both comprehensibility and accuracy are required.

Swarm intelligence [1–3], which deals with the collective behavior of small and simple entities, has been used in many application domains. Ant colony optimization (ACO) [4–6] is one of the most famous meta-heuristic under the umbrella of swarm intelligence. Since its inception it has been used to solve many problems including those related to data mining [7–10]. In this paper, we propose an ACO-based algorithm for the discovery of classification rules. The proposed algorithm, called AntMiner-C, has both properties of accuracy and comprehensibility.

The overall approach of the AntMiner-C algorithm is sequential covering. It starts with a full training set, creates a best rule that covers a subset of the training data, adds the best rule in the discovered rules list, and removes the training samples that are correctly classified by the best rule. This process continues until none or only a few training samples are left. The proposed algorithm has some unique features that allow it to achieve a higher accuracy rate when compared with some previously proposed ACO-based approaches for the same task.

The remainder of this paper is organized as follows. In Sect. 2, we explain the basic concepts of ACO and give an overview of existing ACO algorithms for classification rules discovery. Section 3 describes our proposed approach. In Sect. 4, we present and discuss our experimental results on some publicly available data sets. Finally, Sect. 5 concludes the paper and gives future directions of research.

A. R. Baig (✉) · W. Shahzad
Computer Science Department, National University of Computer and Emerging Sciences, Sector H-11/4, Islamabad, Pakistan
e-mail: Rauf.Baig@nu.edu.pk

W. Shahzad
e-mail: Waseem.Shahzad@nu.edu.pk

2 ACO and its application to classification rule discovery

Ant colony optimization is a branch of swarm intelligence (and, in general, a genre of population-based heuristic algorithms [11]) inspired by the food foraging behavior of biologic ants. Ants pass on the information about the trail they are following by spreading a chemical substance called pheromone in the environment. Ants communicate with each other by means of pheromone. Other ants that arrive in the vicinity are more likely to take the path with higher concentration of pheromone than the paths with lower concentrations. In other words, the desirability of possible paths is proportional to their pheromone concentrations. The pheromone evaporates with time, and if new pheromone is not added, the older one dwindles away.

This indirect form of information passing via the environment helps the ants to find the shortest path to the food source. If two paths between a food source and the ant nest are initially discovered by some ants, then the longer of the two paths will soon become unattractive to subsequent ants, because the ants following it will take longer to go to the food source and return back, and hence, the pheromone concentration on that path will not increase as rapidly as on the shorter path. If an established path is blocked, some ants will first go to the left and some to the right with equal probability. However, an ant taking the shorter of the two paths will return earlier than the ant taking the longer path, and hence, the pheromone on the shorter path will be enhanced, and subsequent ants will have a higher probability of taking the efficient path. Soon a new shorter path that bypasses the blockage will be established. The more ants follow a given trail, the more attractive that trail becomes and is followed by other ants.

This phenomenon has been modeled in the ACO metaheuristic. An artificial ant constructs a solution to the problem by adding solution components one by one. After a solution is constructed, its quality (i.e. fitness) is determined and the components of the solution are assigned pheromone concentrations proportional to this quality. Subsequently, other ants construct their solutions one by one, and they are guided by the pheromone concentrations in their search for components to be added in their solutions. The components with higher pheromone concentrations are thus identified as contributing to a good solution and repeatedly appear in the solutions. Usually, after sufficient iterations, the ants converge on a good, if not the optimal, solution.

For the application of ACO to a problem, we have the following requirements.

- The ability to represent a complete solution as a combination of different components.
- A method to determine the fitness or quality of the solution.
- A heuristic measure for the solution's components (this is helpful, if available, but not necessary).

Since its inception, ACO has been applied to solve many problems [2, 4]. It is naturally suited to discrete optimization problems, such as quadratic assignment, job scheduling, subset problems, network routing, vehicle routing, load dispatch in power systems, bioinformatics, and, of course, data mining, which is the subject of this paper.

2.1 ACO for the discovery of classification rules

The ACO has been applied for the discovery of classification rules. The first ACO-based algorithm for classification rule discovery, called AntMiner, was proposed by Parpinelli et al. [12]. An ant constructs a rule by starting with an empty rule and incrementally adding one condition at a time. The selection of a condition to be added is probabilistic and based on two factors: a heuristic quality of the condition and the amount of pheromone deposited on it by the previous ants. The authors use the information gain as the heuristic value of a condition. After the antecedent part of a rule has been constructed, the consequent of the rule is assigned by a majority vote of the training samples covered by the rule. The constructed rule is then pruned to remove the irrelevant terms and to improve its accuracy. The quality of the constructed rule is determined, and pheromone values are updated on the trail followed by the ant (conditions selected by the ant) proportional to the quality of rule. After all ants have constructed their rules, the best quality rule is selected and added to a discovered rule list. The training samples correctly classified by that rule are removed from the training set. This process is continued until the number of uncovered samples becomes less than a user-specified value. The final product is an ordered rule list that is used to classify the test data.

Versions the AntMiner algorithm are proposed by Liu et al. and called AntMiner2 [13] and AntMiner3 [14, 15]. In AntMiner2, the authors propose density estimation as a heuristic function instead of information gain used by AntMiner. They show that this simpler heuristic value does the same job as well as the complex one, and hence, AntMiner2 is computationally less expensive than the original AntMiner but has comparable performance. In AntMiner3, the authors use a different pheromone update method. They update and evaporate the pheromone of only those conditions that occur in the rule and do not evaporate the pheromones of unused conditions. In this way, exploration is encouraged.

Martens et al. [16] propose a Max–Min Ant System based algorithm (AntMiner+) that differs from the previously proposed AntMiners in several aspects. They define a different search environment for the ants, choose class label of rules prior to their construction, incorporate the ability to form interval rules, and make the selection of two important ACO parameters alpha and beta part of the search process. Only the iteration-best ant is allowed to update the pheromone, the range of the pheromone trail is limited within an interval, and a new rule quality measure is used.

Other works on AntMiner include [17] in which an algorithm for discovering unordered rule sets has been presented. In [18], PSO algorithm is used for continuous-valued attributes and ACO for nominal-valued ones, and these two algorithms are jointly used to construct rules. The issue of continuous attributes has also been dealt in [19] and [20]. An AntMiner version for multi-label classification problem can be found in [21]. Relevant background information regarding AntMiners can be found in [22].

Our proposed algorithm is different in many ways from the previous AntMiner algorithms. The main novelty of our approach is a new class dependant heuristic function based on correlation which reduces the search space considerably and yields better results. Other differences are highlighted along with the new algorithm's description in the next section.

3 Correlation-based AntMiner (AntMiner-C)

In this section, we describe our ACO-based classification rules mining algorithm called AntMiner-C. We begin with a general description of the algorithm and then discuss the heuristic function, pheromone update, rule pruning, early stopping, etc.

3.1 General description

The core of the algorithm is the incremental construction of a classification rule of the type

IF $<\text{term}_1$ AND term_2 AND $\ldots >$ THEN $<\text{class}>$

by an ant. Each term is an attribute-value pair related by an operator. In our current experiments, we use "="as the only possible relational operator. An example term is "Color = red". The attribute's name is "color", and "red" is one of its possible values. Since we use only "=", any continuous (real-valued) attributes present in the data have to be discretized in a preprocessing step.

For the ants to search for a solution, we need to define the space in which the search is to be conducted. This space is defined by the data set used. The dimensions (or coordinates) of the search space are the attributes of the data set. The class attribute is included in the search space but not handled in the same way as other attributes. The possible values of an attribute constitute the range of values for the corresponding dimension in the search space. For example, a dimension called 'Color' may have three possible values {red, green, blue}. The task of the ant is to visit a dimension and choose one of its possible values to form an antecedent condition of the rule (e.g., Color = red). The total number of terms for a data set is equal to

$$Total\_terms = \sum_{n=1}^{a} b_n \quad (1)$$

where a is the total number of attributes (excluding the class attribute) and b_n is the number of possible values that can be taken on by an attribute A_n. When a dimension has been visited, it cannot be visited again by an ant, because we do not allow the conditions of the type "Color = red OR Color = green". The search space is such that an ant may pick a term from any dimension, and there is no ordering in which the dimensions can be visited. An example search space represented as a graph is shown in Fig. 1.

The same search space has been utilized by AntMiner [12]. The search environment utilized by AntMiner+ [16] is different in several ways: each of the ants is required to choose one of the classes (majority class is excluded from

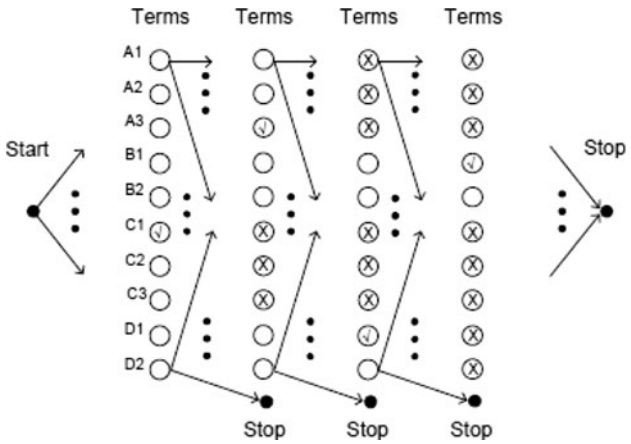

Fig. 1 An example problem's search space represented as a graph. There are four attributes *A*, *B*, *C*, and *D* having 3, 2, 3, and 2 possible values, respectively. An ant starts from the "Start" vertex and constructs a rule by adding conditions (attribute-value pairs or terms) for the antecedent part. After a term has been selected, all the other terms from the same attribute become prohibited for the ant. Suppose the ant chooses *C1* as its first term, it cannot select any more terms from attribute *C*. Further, suppose that the ant subsequently chooses *A3*, *D1*, and *B1*. The rule construction process is stopped when the ant reaches the "Stop" vertex. The consequent part of the rule (class label) is known to the ant prior to the rule construction. It can stop prematurely if the addition of the latest term makes the rule to cover only those training samples that have the chosen class label

this set) before constructing its rule, the selection of alpha and beta parameters are made part of the search process, the dimensions (i.e., attributes) are visited in a fixed order but an ant may choose to ignore a dimension, and the discretization of continuous variables is replaced by a mechanism that enables ants to include interval conditions in their rules.

A general description of the AntMiner-C algorithm is shown in Fig. 2. Its basic structure is that of AntMiner [12]. It is a sequential covering algorithm. It discovers a rule and the training samples correctly covered by this rule (i.e., samples that satisfy the rule antecedent and have the class predicted by the rule consequent) are removed from the training set. The algorithm discovers another rule using the reduced training set, and after its discovery, the training set is further reduced by removing the training samples correctly covered by the newly discovered rule. This process continues until the training set is almost empty, and a user-defined number is used as a condition for terminating the discovery of new rules. If the number of remaining uncovered samples in the training set is lower than or equal to this number, then the algorithm stops.

One rule is discovered after one execution of the outer WHILE loop. First of all, a class label is chosen from the set of class labels present in the uncovered samples of training set. This class label remains constant for a batch of ants of the REPEAT-UNTIL loop. Each iteration of the REPEAT-UNTIL loop sends an ant to construct a rule. Each ant starts with an empty list of conditions and constructs the antecedent part of the rule by adding one term at a time. Every ant constructs its rule for the particular class chosen before the beginning of the REPEAT-UNTIL loop. The quality of the rule constructed by an ant is determined and pheromones are updated, and then another ant is sent to construct another rule. Several rules are constructed through an execution of the REPEAT-UNTIL loop. The best one among them, after pruning, is added to the list of discovered rules.

The algorithm terminates when the outer WHILE loop exit criterion is met. The output of the algorithm is an ordered set of rules. This set can then be used to classify unseen data. The main features of the algorithm are explained in detail in the following sub-sections.

As mentioned before, an ant constructs the antecedent part of the rule by adding one term at a time. The choice of adding a term in the current partial rule is based on the pheromone value and heuristic value associated with the term and is explained in the next sub-section.

3.2 Rule construction

Rule construction, which is an important part of the algorithm, is described below.

3.2.1 Class commitment

Our heuristic function (described later) is dependent upon the class label of samples, and hence, we need to decide beforehand the class of the rule being constructed. For each iteration of the WHILE loop in the algorithm, a class is chosen probabilistically, by roulette wheel selection, on the basis of the weights of the classes present in the yet

Fig. 2 The AntMiner-C algorithm

```
Size(TrainingSet) = {all training samples};
DiscoveredRuleList = {};            /* rule list is initialized with an empty list */
WHILE (Size(TrainingSet) > Max_uncovered_samples)
        t = 1;                      /* counter for ants */
        rcc = 1;                    /* counter for rule convergence test */
        Select class label;
        Initialize pheromone values on all edges;
        Calculate heuristic values for all edges;
        REPEAT
                Send an Ant_t which constructs a classification rule R_t for the selected class;
                Assess the quality of the rule and update the pheromone of all trails;
                IF (R_t is equal to R_t-1)      /* update convergence test */
                        THEN rcc = rcc + 1;
                        ELSE rcc = 1;
                END IF
                t = t + 1;
        UNTIL (t ≥ No_of_ants) OR (rcc ≥ No_rules_converg)
        Choose the best rule R_best among all rules R_t constructed by all the ants;
        Prune the best rule R_best;
        Add the pruned best rule R_best to DiscoveredRuleList;
        Remove the training samples correctly classified by the pruned best rule R_best;
END WHILE
Add a default rule in the DiscoveredRuleList:
Prune the Discovered RuleList; (Optional)
```

uncovered data samples. The weight of a class is the ratio of its uncovered data samples to the total number of uncovered data samples. If 40% of the yet uncovered data samples are of class "A", then there is a 40% chance of its selection. The class is chosen once and becomes fixed for all the ant runs made in the REPEAT-UNTIL loop.

Our method of class commitment is different from those used by AntMiner and AntMiner+. In AntMiner, the consequent part of the rule is assigned after rule construction and is according to the majority class among the cases covered by the rule. In AntMiner+, each ant selects the consequent of the rule prior to antecedent construction. The consequent selection is from a set of class labels which does not contain the class label of samples in majority. In our case, even though the class label is selected prior to the rule construction, it is not selected by the individual ants and is fixed for all the ant runs of a REPEAT-UNTIL loop. Also, the set of class labels does not exclude the class label of samples in majority.

3.2.2 Pheromone initialization

At the beginning of each iteration of the WHILE loop, the pheromone values on the edges between all terms are initialized. The initial pheromone between two terms $term_i$ and $term_j$ belonging to two different attributes is

$$\tau_{ij}(t=1) = \frac{1}{\sum_{n=1}^{a} x_n b_n} \tag{2}$$

where a is the total number of attributes (excluding the class attribute), b_n is the number of possible values that can be taken on by an attribute A_n and x_n is set to 0 if the attribute A_n is that to which $term_i$ belongs, otherwise it is set to 1. The pheromones on edges between the terms of same attribute are initialized to zero. Since all the pheromone values for competing terms are the same, the first ant has no historical information to guide its search. This method of pheromone initialization has been used by the AntMiner. AntMiner+ utilizes MAX–MIN Ant System and all the pheromone values are set equal to a value $\tau_{\max}$.

3.2.3 Term selection

An ant incrementally adds terms in the antecedent part of the rule that it is constructing. The selection of the next term is subject to the restriction that a term from its attribute A_n should not be already present in the current partial rule. In other words, once a term has been included in the rule then no other term from that attribute can be considered. Subject to this restriction, the probability of selection of a term for addition in the current partial rule is given by the equation:

$$P_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(s)}{\sum_{j=1}^{Total\_terms} x_j\{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(s)\}} \tag{3}$$

where $\tau_{ij}(t)$ is the amount of pheromone associated with the edge between $term_i$ and $term_j$ for the current ant (the pheromone value is updated after passage of each ant), $\eta_{ij}(s)$ is the value of the heuristic function for the current iteration s of the WHILE loop (constant for a batch of ants), x_j is a binary variable that is set to 1 if $term_j$ is selectable, otherwise it is 0. Selectable terms are those that belong to attributes which have not become prohibited due to the selection of a term belonging to them. The denominator is used to normalize the numerator value for all the possible choices.

The parameters α and β are used to control the relative importance of the pheromones and heuristic values in the probability determination of next movement of the ant. We use $\alpha = \beta = 1$, which means that we give equal importance to the pheromone and heuristic values. However, we note that different values may be used (e.g., $\alpha = 2$, $\beta = 1$ or $\alpha = 1$, $\beta = 3$), and we have carried out a limited experiment in this regard (Sect. 4.7).

Equation 3 is a classical equation and used in Ant System, MAX–MIN Ant System, Ant Colony System (where the state transition is also dependant on one other equation), AntMiner (with $\alpha = \beta = 1$), and AntMiner+.

3.2.4 Heuristic function

The heuristic value of an edge gives an indication of its usefulness and thus provides a basis to guide the search. We use a heuristic function based on the correlation of the most recently chosen term with other candidate terms in order to guide the selection of next term. The heuristic function is:

$$\eta_{ij} = \frac{|term_i, term_j, class_k|}{|term_i, class_k|} \cdot \frac{|term_j, class_k|}{|term_j|}. \tag{4}$$

The most recently chosen term is $term_i$, and the term being considered for selection is $term_j$. The number of uncovered training samples having $term_i$ and $term_j$ and which belong to the committed class label k of the rule is given by $|term_i, term_j, class_k|$. This number is divided by the number of uncovered training samples, which have $term_i$ and which belong to $class_k$, to find the correlation between the terms $term_i$ and $term_j$. The value of the heuristic function is zero for edges between terms of the same attribute.

The correlation is multiplied by the importance of $term_j$ in determining the $class_k$. The factor $|term_j, class_k|$ is the number of uncovered training samples having $term_j$ and

which belong to $class_k$, and the factor $|term_j|$ is the number of uncovered training samples containing $term_j$, irrespective of their class label. Since the class label of the rule is committed before the construction of the rule antecedents, our heuristic function is dependent on the class chosen for a batch of ants.

The heuristic function quantifies the relationship of the term to be added with the most recently added term and also takes into consideration the overall discriminatory capability of the term to be added. An example is shown in Fig. 3. Suppose the specified class label is $class_k$. An ant has recently added $term_i$ in its rule. It is now looking to add another term from the set of available terms. We take the case of two competing terms ($term_{j1}$ and $term_{j2}$) that can be easily generalized to more terms. We want the chosen term to maximize correct coverage of the rule as much as possible. This is encouraged by the first part of the heuristic function. The heuristic value on edge between $term_i$ and $term_{j1}$ is $|term_i, term_{j1}, class_k|/|term_i, class_k|$ and that on the edge between $term_i$ and $term_{j2}$ is $|term_i, term_{j2}, class_k|/|term_i, class_k|$. We also want to encourage the inclusion of a term that has better potential for inter class discrimination. This is made possible by the second portion of the heuristic function. The two competing terms will have the values $|term_{j1}, class_k|/|term_{j1}|$ and $|term_{j2}, class_k|/|term_{j2}|$. The two ratios quantifying correct coverage and interclass discrimination are multiplied to give the heuristic value of the edges between terms. The heuristic values calculated according to Eq. 4 are normalized before usage. The heuristic values on edges originating from a $term_i$ are normalized by the summation of all heuristic values on the edges between $term_i$ and other terms.

The correlation-based heuristic function has the potential to be effective in large dimensional search spaces. The heuristic values are calculated only once at the beginning of the REPEAT-UNTIL loop. It assigns a zero value to the combination of those terms that do not occur together for a given class label, thus efficiently restricting the search space for the ants. In contrast, each ant of AntMiner continues its attempts to add terms in its rule until it is sure that there is no term whose addition would not violate the condition of minimum number of covered samples. In AntMiner+, every ant has to traverse the whole search space and there are no such short cuts.

3.2.5 Heuristic function for the 1st layer of terms

The heuristic values on the edges between the Start node and the first layer terms are calculated on the basis of the following Laplace-corrected confidence [9] of a term:

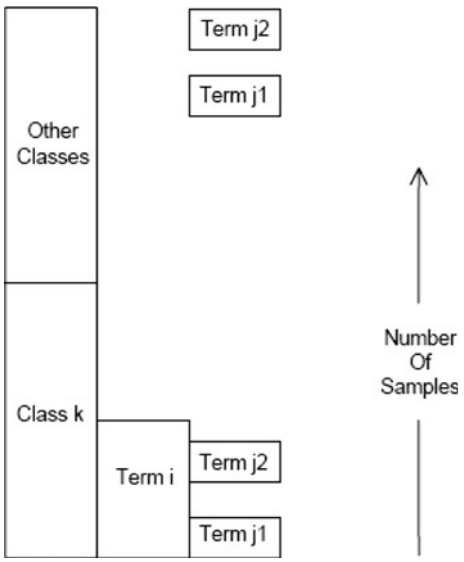

Fig. 3 Suppose the specified class label is $class_k$, and an ant has recently added $term_i$ in its rule. It is now looking to add another term from the set of available terms. For two competing terms ($term_{j1}$ and $term_{j2}$), we want the chosen term to maximize correct coverage of the rule as much as possible, and we also want to encourage the inclusion of a term that has better potential for inter class discrimination. These two considerations are incorporated in the heuristic function

$$\eta_{Start,j} = \frac{|term_j, class_k| + 1}{|term_j| + m} \tag{5}$$

where m is the number of classes present in the data set. This heuristic function has the advantage of penalizing the terms that would lead to very specific rules and thus helps to avoid over-fitting. For example, if a term occurs in just one training sample and its class is the chosen class, then its confidence is 1 without the Laplace correction. With Laplace correction, its confidence is 0.66 if there are two classes in the data. Before usage, the values obtained by Eq. 5 are normalized by the summation of all such values between the Start node and the first layer of terms.

Equations 4 and 5 have not been used before in any AntMiner. All previous AntMiner versions have calculated the heuristic value of a candidate $term_j$ without considering any other previously chosen term. AntMiner utilizes a heuristic function based on the entropy of the terms and their normalized information gain.

$$\frac{\log_2 m - H(W|term_j)}{\sum_{competing\_terms}(\log_2 m - H(W|term_j))} \tag{6}$$

where W is the class attribute whose entropy H given $term_j$ is defined as

$$H(W|term_j) = -\sum_{k=1}^{m} (P(k|term_j)\ \log_2 P(k|term_j)) \quad (7)$$

where k is one of the values of W, and $P(k|term_j)$ is the probability of having class label k for instances containing $term_j$. AntMiner2 and AntMiner3 use

$$\frac{|term_j, majority\_class(term_j)|}{|term_j|} \quad (8)$$

and AntMiner+ uses

$$\frac{|term_j, class\_chosen\_by\_ant|}{|term_j|}. \quad (9)$$

3.2.6 Rule termination

An ant continues to add terms in the rule it is constructing and stops only when all the samples covered by the rule have homogenous class label (which is committed prior to the REPEAT-UNTIL loop). The other possibility of termination is that there are no more terms to add. Due to these termination criteria, it might happen that a constructed rule may cover only one training sample.

Our rule construction stoppage criterion of homogenous class label is different from that of AntMiner and AntMiner+. In AntMiner, the rule construction is stopped if only those terms are left unused whose addition will make the rule cover a number of training samples smaller than a user-defined value called minimum samples per rule. In AntMiner+, one term from each attribute is added. However, each attribute has a don't care option that allows for its non-utilization in the rule.

3.3 Rule quality and pheromone update

3.3.1 Quality of a rule

When an ant has completed the construction of a rule, its quality is calculated. The quality, Q, of a rule is computed by using confidence and coverage of the rule and is given by:

$$Q = \frac{TP}{Covered} + \frac{TP}{N} \quad (10)$$

where TP is the number of samples covered by the rule that have the same class label as that of the rule's consequent, $Covered$ is the total number of samples covered by the rule, and N is the number of samples in the training set yet uncovered by any rule in the discovered rule set. The second portion is added to encourage the construction of rules with wider coverage. Equation 10 has been used in AntMiner+, whereas AntMiner uses sensitivity multiplied by specificity as the quality measure.

(a)

	A1	A2	A3	B1	B2	C1	C2	C3	D1	D2
S	τ_{01}	τ_{02}	τ_{03}	τ_{04}	τ_{05}	τ_{06}	τ_{07}	τ_{08}	τ_{09}	τ_{010}

(b)

	A1	A2	A3	B1	B2	C1	C2	C3	D1	D2
A1	0	0	0	τ_{14}	τ_{15}	τ_{16}	τ_{17}	τ_{18}	τ_{19}	τ_{110}
A2	0	0	0	τ_{24}	τ_{25}	τ_{26}	τ_{27}	τ_{28}	τ_{29}	τ_{210}
A3	0	0	0	τ_{34}	τ_{35}	τ_{36}	τ_{37}	τ_{38}	τ_{39}	τ_{310}
B1	τ_{41}	τ_{42}	τ_{43}	0	0	τ_{46}	τ_{47}	τ_{48}	τ_{49}	τ_{410}
B2	τ_{51}	τ_{52}	τ_{53}	0	0	τ_{36}	τ_{37}	τ_{38}	τ_{39}	τ_{310}
C1	τ_{61}	τ_{62}	τ_{63}	τ_{64}	τ_{65}	0	0	0	τ_{69}	τ_{610}
C2	τ_{71}	τ_{72}	τ_{73}	τ_{74}	τ_{75}	0	0	0	τ_{79}	τ_{710}
C3	τ_{81}	τ_{82}	τ_{83}	τ_{84}	τ_{85}	0	0	0	τ_{89}	τ_{810}
D1	τ_{91}	τ_{92}	τ_{93}	τ_{94}	τ_{95}	τ_{96}	τ_{97}	τ_{98}	0	0
D2	τ_{101}	τ_{102}	τ_{103}	τ_{104}	τ_{105}	τ_{106}	τ_{107}	τ_{108}	0	0

Fig. 4 The pheromone values for the example problem of Fig. 1. The elements of (**a**) are the pheromone values on edges from start node to nodes of the first layer of terms. The elements of (**b**) are pheromone values on edges between terms in any two consecutive layers. For example, the first row elements are the pheromone values for edges from the term *A1* to all other terms in the next layer. These values are the same for edges between 1st and 2nd layer, 2nd and 3rd layer, and so on. If an ant chooses *C1*, *A3*, *D1*, and *B1* terms in its rule then the elements τ_{06}, τ_{63}, τ_{39}, and τ_{94} are updated according to Eq. 11. Elements of rows *S*, *C1*, *A3*, and *D1* are then normalized. The remaining rows remain unchanged

3.3.2 Pheromone update

An example pheromone matrix is shown in Fig. 4. The pheromone matrix is asymmetric and captures the fact that pheromone values on edges originating from a term to other terms are kept the same in all the layers of the search space. In other words, the same matrix is valid for any two consecutive layers.

The pheromone values are updated so that the next ant can make use of this information in its search. The amount of pheromone on the edges between consecutive terms occurring in the rule is updated according to the equation:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \left(1 - \frac{1}{1+Q}\right)\tau_{ij}(t) \quad (11)$$

where $\tau_{ij}(t)$ is the pheromone value encountered by Ant_t (the t ant of the REPEAT-UNTIL loop) between $term_i$ and $term_j$. The pheromone evaporation rate is represented by ρ, and Q is the quality of the rule constructed by Ant_t.

Equation 11 is according to the Ant System and has been previously used in AntMiner3, but with a different equation for determining Q. The equation updates pheromones by first evaporating a percentage of the previously occurring pheromone and then adding a percentage of the pheromone dependant on the quality of the recently discovered rule. If the rule is good, then the pheromone added

is greater than the pheromone evaporated and the terms become more attractive for subsequent ants. The evaporation in the equation improves exploration, otherwise in the presence of a static heuristic function the ants tend to converge quickly to the terms selected by the first few ants.

Note that our pheromone matrix is asymmetric, whereas the matrix used in original AntMiner is symmetric. The matrix becomes asymmetric because for a constructed rule with $term_i$ occurring immediately before $term_j$, the pheromone on edge between $term_i$ and $term_j$ is updated, but the pheromone on edge between $term_j$ and $term_i$ is not updated. This is done to encourage exploration and discourage early convergence.

The next step is to normalize the pheromone values. Each pheromone value is normalized by dividing it by the summation of pheromone values on the edges of all its competing terms. In the pheromone matrix (Fig. 4), normalization of the elements is done by dividing them with the summation of values of the row to which they belong. Note that for those rows for which no change in the values has occurred for the ant run under consideration, the normalization process yields the same values as before. Referring to Fig. 1, normalization of pheromone value of an edge originating from a term is done by dividing it by the summation of all pheromone values of the edges originating from that term. This process changes the amount of pheromone associated with those terms that do not occur in the most recently constructed rule but are the competitors of the selected term. If the quality of rule has been good and there has been a pheromone increase in the terms used in the rule, then the competing terms become less attractive for subsequent ants. The reverse is true if the rule found is not of good quality. The normalization process is an indirect way of simulating evaporation of pheromones. Note that in original AntMiner, every element of the pheromone matrix is normalized by dividing it with the sum of all elements. This is unnecessary and tends to discourage exploration and favors early convergence.

3.4 Termination of REPEAT-UNTIL loop

The REPEAT-UNTIL loop is used to construct as many rules as the user-defined number of ants. After the construction of each rule, its quality is determined and the pheromones on the trail are updated accordingly. The pheromone values guide the construction of next rule. An early termination of this loop is possible if the last few ants have constructed the same rule. This implies that the pheromone values on a trail have become very high and convergence has been achieved. Any further rule construction will most probably yield the same rule again. Hence, the loop is terminated prematurely. For this purpose, each constructed rule is compared with the last rule and a counter is incremented if both the rules are the same. If the value of this counter becomes equal to a user defined parameter called "No_rules_converge", then the loop is terminated. In our experiments, we use a value of 10 for this parameter.

This method of early termination of REPEAT-UNTIL loop is also used by AntMiner. In AntMiner+, there is no provision for early termination of this loop and it terminates when the pheromone values on one path converge to τ_{max} and all other paths have τ_{min}, as required by the MAX–MIN Ant System.

3.5 Pruning of best rule

Rule pruning is the process of finding and removing any irrelevant terms that might have been included in the constructed rule. Rule pruning has two advantages. First, it increases the generalization of the rule thus potentially increasing its predictive accuracy. Second, a shorter rule is usually simpler and more comprehensible.

In AntMiner-C, rule construction by an ant stops when all the samples covered by the rule have homogenous class label (which is committed prior to the REPEAT-UNTIL loop). The other possibility of termination is that there are no more terms to add. Those rules whose construction is stopped due to homogeneity of class label of samples covered are already compact, and any attempt at pruning is bound to cause non-homogeneity. However, those rules that comprise of one term from all the attributes have the possibility of improvement with pruning of terms. Based on an experiment (Sect. 4.5), we do not prune all found rules but apply the pruning procedure only to the best rule discovered during an iteration of the WHILE loop. This means that pheromone updates are done with unpruned rules.

The rule pruning procedure starts with the full rule. It temporarily removes the first term and determines the quality of the resulting rule. It then replaces the term back and temporarily removes the second term and again calculates the quality of the resulting rule. This process continues until all the terms present in the rule are dealt with. After this assessment, if there is no term whose removal improves or maintains the quality, then the original rule is retained. However, if there is a term whose removal improves (or maintains) the quality of the rule, then that term is permanently removed. If there are two or more such terms, then the term whose removal most improves the quality of the rule is removed. The shortened rule is again subjected to the procedure of rule pruning. The process continues until any further shortening is impossible (removal of any term present in the rule leads to decrease in its quality) or if there is only one remaining term in the rule.

Rule pruning is also used by AntMiner and AntMiner+. However, the rule quality criterion used in AntMiner is different. Furthermore, in AntMiner, each discovered rule is subjected to rule pruning (prior to pheromone update), whereas we prune only the best rule among the rules found during the execution of the REPEAT-UNTIL loop of the algorithm, and our pheromone update is prior to and independent of rule pruning. AntMiner+ also prunes the best rule. However, it is pertinent to note that AntMiner+ has several iterations of the REPEAT-UNTIL loop, and in each iteration, there are a 1,000 ant runs. The best rule from these 1,000 rules is pruned. In other words, AntMiner+ prunes several rules for each rule inserted in the rule set, whereas we prune only one rule.

3.6 Final rule set

The best rule is placed in the discovered rule set after pruning, and the training samples correctly covered by the rule are flagged (i.e., removed) and have no role in the discovery of other rules. The algorithm checks whether the uncovered training samples are still greater than the value specified for "Max_uncovered_samples". If that is the case, a new iteration of the WHILE loop starts for discovery of the next rule. If not, a final default rule is added to the rule set, and the rule set may optionally be pruned of redundant rules (see Sect. 4.6 for more details). The rule set may then be used for classifying unseen samples. These aspects are discussed here.

3.6.1 *Early stoppage of algorithm*

The algorithm can be continued until the training data set is empty. However, this leads to rules which usually cover one or two training samples and thus over-fit the data. Such rules usually do not have generalization capabilities. Some of the methods to avoid this phenomenon are the following:

- Use a separate validation set to monitor the training. The algorithm can be stopped when the accuracy on the validation set starts to dip.
- Specify a value for the number of training samples present in the data set. If, after an iteration of the REPEAT-UNTIL loop, the remaining samples in the training set are equal to or below this specified value, then stop the algorithm. This threshold value can also be defined as a percentage of the samples present in the initial training set.
- Specify a threshold on the maximum number of rules to be found.

The validation set is not appropriate for small data sets because we have to divide the total data samples into training, validation, and test sets.

We have opted for the second option and use a fixed threshold for all the data sets. AntMiner and AntMiner+ both employ early stopping. We use the same option for early stopping as is done in AntMiner while AntMiner+ uses the validation set technique for large data sets (greater than 250 samples) and a threshold of 1% remaining samples for small data sets.

3.6.2 *Default rule*

A final default rule is added at the bottom of the rule set. This rule is without any conditions and has a consequent part only. The assigned class label for this rule is the majority class label of the remaining uncovered samples of the training set. The default rule is used by all AntMiners, and our method is same as that used by AntMiner. AntMiner+, however, assigns the default rule class label on the basis of majority class of the complete set of training samples.

3.6.3 *Pruning of rule set*

Some of the rules may be redundant in the final rule set. The training samples correctly classified by a rule may all be correctly classified by one or more rules occurring later on in the rule set. In such cases, the earlier rules are redundant, and their removal will not decrease the accuracy of the rule set. Furthermore, the removal of redundant rules increases the comprehensibility of the rule set. We have developed a procedure that attempts to reduce the quantity of rules without compromising on the accuracy obtained.

The rule set pruning procedure is applied to the final rule set that includes the default rule. Each rule is a candidate for removal. The procedure checks the first rule, and if removing it from the rule list does not decrease the accuracy on the training data, then it is permanently removed. After dealing with the first rule, the second rule is checked, and it is either retained or removed. Each rule is subjected to the same procedure on its turn.

Our experiments (Sect. 4.6) show that the technique is effective in reducing the number of rules. The remaining rules also have a tendency to have lesser terms/rule on the average. However, the strategy is greedy, and although it makes the final classifier relatively fast, but it tends to decrease predictive accuracy for some data sets. However, increased accuracy on other data sets make us believe that sometimes the pruning results in a rule set with superior generalization capability. Hence, pending further investigation, we have made this procedure optional.

3.6.4 Use of discovered rule set for classifying unseen samples

A new test sample unseen during training is classified by applying the rules in order of their discovery. The first rule whose antecedents match the new sample is fired, and the class predicted by the rule's consequent is assigned to the sample. If none of the discovered rules are fired, then the final default rule gets activated.

3.7 Differences with other AntMiners

AntMiner-C has several similarities to previously proposed AntMiners, yet it is also different in many ways. The main differences are presented here.

3.7.1 Differences with AntMiner, AntMiner2, and AntMiner3

AntMiner-C has the following:

- prior selection of class label,
- new heuristic measure, Eq. 4,
- pruning of best rule only,
- rule construction termination on the basis of class homogeneity of samples covered by that rule,
- an asymmetric pheromone matrix,
- a different pheromone update equation, Eq. 11 (an exception is AntMiner3 that has the same update equation),
- a different method of pheromone normalization, and
- a different equation for assessing the quality of rule found and for rule pruning, Eq. 10.

In addition to these differences, AntMiner3 utilizes Eq. 3 in conjunction with a transition rule for the selection of next term that is different than ours.

3.7.2 Differences with AntMiner+

AntMiner-C has the following:

- a different search space,
- a different method of class selection that is done only once and subsequently the class remains fixed for all ant runs made for the corresponding antecedent part of the rule,
- does not exclude the majority class from the class selection choices,
- no provision of selection of alpha and beta parameters,
- no mechanism by which ants can include interval conditions in their rules, and continuous variables have to be discredited as a preprocessing step,
- state transition (next term selection) and pheromone update according to Ant System,
- a new heuristic measure, Eq. 4,
- a criterion for early stopping of rule construction,
- a provision for early stopping of algorithm according to a user-defined maximum number of uncovered training samples, and
- default rule class label assignment according to the majority class of remaining uncovered samples of the training set.

Furthermore, AntMiner-C has only one complete execution of the REPEAT-UNTIL loop for the extraction of a rule. Also, each iteration of the loop consists of only one ant run. For example, we use 1,000 iterations in our experiments and also find that, on the average, only 18% of these iterations are executed due to the early termination of rule construction (Sect. 4.2). The best solution is pruned after exiting from the REPEAT-UNTIL loop. In AntMiner+, the REPEAT-UNTIL loop is executed until the pheromone values on one path converge to τ_{max} and for all other paths are τ_{min}. There are multiple ants per iteration of the loop (1,000 ants are used in the experiments reported in [16]). The best solution from each iteration is pruned. In other words, several solutions are pruned before exiting from the REPEAT-UNTIL loop.

4 Experiments and analysis

In this section, we report our experimental setup and the results obtained.

4.1 Experimental setup and results

In our experiments, we use twelve data sets obtained from the UCI repository [23]. The main characteristics of the data sets are summarized in Table 1. The data sets in this suite have reasonable variety in terms of number of attributes, instances, and classes and are commonly used in evaluating algorithms.

AntMiner-C algorithm works with categorical attributes, and continuous attributes need to be discretized in a preprocessing step. We use unsupervised discretization filter of Weka-3.4 machine learning tool [9] for discretizing continuous attributes. This filter first computes the intervals of continuous attributes from the training data set and then uses these intervals to discretize them.

We compare the results of our algorithm with those for AntMiner, C4.5 [24, 25], Ripper, logistic regression, and support vector machines (SVM) [26]. AntMiner has been implemented by us in Matlab. For other compared algorithms, we use the Weka machine learning tool [9]. Our performance metrics for the comparison of rule sets obtained by competing algorithms are: predictive

Table 1 Characteristics of data sets used in experiments

Dataset	Attributes	Instances	Classes
Wisconsin breast cancer	9	683	2
Wine	13	178	3
Credit (Australia)	15	690	2
Credit (Germany)	19	1,000	2
Car	6	1,728	4
Tic-tac-toe	9	958	2
Iris	4	150	3
Balance scale	4	625	3
Teacher assistant evaluation (TAE)	6	151	3
Glass	9	214	7
Heart	13	270	2
Hepatitis	19	155	2

Table 2 Parameters used in experiments

Parameter	Value
Number of ants	1,000
Max. uncovered samples	10
Evaporation rate	0.15
No. of rules converged	10
Alpha	1
Beta	1

accuracy, number of rules, and number of terms per rule.

The experiments are performed using a ten fold cross-validation procedure. A data set is divided into ten equally sized, mutually exclusive subsets. Each of the subset is used once for testing while the other nine are used for training. The results of the ten runs are then averaged, and this average is reported as final result.

We have implemented the AntMiner-C algorithm in Matlab 7.0. The experiments are run on a machine that has 1.75 GHz dual processors with 1 GB RAM. The time duration of experiments of different data sets depends upon the number of attributes and the number of instances in the data set. For example, the duration of experiment of one fold is almost 3 min for the iris data set that has 150 instances and 4 attributes. For the dermatology data set, which has 34 attributes and 366 instances, the duration of experiment of one fold is almost 15 min.

The AntMiner-C has six user-defined parameters: number of ants, maximum uncovered samples, evaporation rate, convergence counter threshold, alpha, and beta. The values of these parameters are given in Table 2. The option of pruning of rule set is not used in any experiment except that of Sect. 4.5. The same parameters have been retained while obtaining results for AntMiner. These values have been chosen because they seem reasonable and have been used by some other AntMiner versions reported in literature [12–17]. In Sect. 4.7 we report the results of some limited experiments that give an indication of the influence of α, β, and ρ on the performance of AntMiner-C. We found that the predictive accuracy results are highest when $\rho = 0.15$, $\alpha = 1$, and $\beta = 3$.

The predictive accuracies, average number of rules per discovered rule set, and average number of terms per rule are shown in Tables 3 and 4. All results are obtained using ten fold cross validation.

The results indicate that the AntMiner-C achieves higher accuracy rate than the five compared algorithms for most of the data sets. However, the number of rules and the number of terms per rule generated by our proposed technique are mostly higher. The reason is that we allow the generation of rules with low coverage.

4.2 Convergence speed

In Table 2, we have specified the value of the 'Number of Ants' parameter as 1,000, that is, a maximum of 1,000 rules can be constructed out of which the best one is chosen to be placed in the discovered rule set. In reality, on the average, very less ants are used because the REPEAT-UNTIL loop gets terminated if convergence is achieved (all of the recently discovered rules are the duplicates of each other; Sect. 3.4). For this purpose, the threshold parameter 'Number of rules converged' (10 in our experiments) is used. The average of the actual number of ants used per execution of REPEAT-UNTIL loop is reported in Table 5. Convergence speed is an important aspect, particularly, for large- and high-dimensional data sets.

4.3 Class choice prior or after rule construction

In this experiment, we deviate from our algorithm by first constructing the rule antecedent and then assigning the rule consequent. The rule consequent assigned is the majority class label of the samples covered by the rule. Since our heuristic function (Eq. 4) requires prior commitment of class label, we have to modify it for this particular experiment. The heuristic function used is

$$\eta_{ij} = \frac{|term_i, term_j|}{|term_i|}. \tag{12}$$

The heuristic function used for the first layer of terms is

$$\eta_j = \sum_{k=1}^{m} -\frac{|term_j, class_k|}{|term_j|} \log_2 \left(\frac{|term_j, class_k|}{|term_j|} \right). \tag{13}$$

Experimental results are shown in Table 6. The predictive accuracy is lower when compared to our

Table 3 Average predictive accuracies obtained using 10-fold cross validation

Datasets	AntMiner-C	AntMiner	C4.5	Ripper	Logistic reg.	SVM
BC-W	**97.54 ± 0.98**	94.64 ± 2.74	94.84 ± 2.62	95.57 ± 2.17	96.56 ± 1.21	96.70 ± 0.69
Wine	98.24 ± 2.84	90.0 ± 9.22	96.60 ± 3.93	94.90 ± 5.54	96.60 ± 4.03	**98.30 ± 2.74**
Credit (Australia)	**89.42 ± 4.21**	86.09 ± 4.69	81.99 ± 7.78	86.07 ± 2.27	85.77 ± 4.75	85.17 ± 2.06
Credit (Germany)	73.64 ± 2.67	71.62 ± 2.71	70.73 ± 6.71	70.56 ± 5.96	**75.82 ± 4.24**	75.11 ± 3.63
Car	**98.02 ± 0.96**	82.38 ± 2.42	96.0 ± 2.13	89.17 ± 2.52	93.22 ± 2.10	93.74 ± 2.65
Tic-tac-toe	**100 ± 0.0**	74.95 ± 4.26	94.03 ± 2.44	97.57 ± 1.44	98.23 ± 0.50	98.33 ± 0.53
Iris	**97.33 ± 4.66**	95.33 ± 4.50	94.0 ± 6.63	94.76 ± 5.26	**97.33 ± 5.62**	96.67 ± 3.52
Balance scale	86.61 ± 6.18	75.32 ± 8.86	83.02 ± 3.24	80.93 ± 3.35	**88.30 ± 2.69**	87.98 ± 1.80
TAE	**77.33 ± 10.5**	50.67 ± 6.11	51.33 ± 9.45	44.67 ± 10.35	53.33 ± 11.33	58.67 ± 10.98
Glass	**74.29 ± 6.43**	53.33 ± 4.38	68.90 ± 8.98	70.48 ± 8.19	63.65 ± 6.72	57.70 ± 8.10
Heart	77.78 ± 5.79	**80.74 ± 4.94**	78.43 ± 6.26	73.59 ± 9.57	77.0 ± 5.05	80.32 ± 6.25
Hepatitis	**87.33 ± 9.66**	80.67 ± 8.67	68.25 ± 11.63	73.46 ± 8.21	64.25 ± 8.87	75.37 ± 8.62

The best values of each row are shown in bold

Table 4 Average number of rules per discovered rule set and average number of terms per rule

	No. of rules/rule set				Terms/rule			
	AM-C	AntMiner	C4.5	Ripper	AM-C	AntMiner	C4.5	Ripper
BC-W	18.60	11.0	10.50	**5.10**	1.45	**1.02**	2.32	1.79
Wine	4.10	5.50	5.30	**3.90**	1.65	**1.04**	1.41	1.62
Credit (Australia)	7.80	**3.90**	74.80	4.60	1.58	**1.0**	3.22	1.81
Credit (Germany)	10.0	8.50	73.60	**4.20**	1.71	**1.13**	3.21	2.36
Car	57.60	**11.40**	80.26	41.10	2.49	**1.03**	2.59	4.01
Tic-tac-toe	14.80	**6.60**	38.60	10.30	2.50	**1.09**	2.64	2.82
Iris	10.90	9.20	5.50	**3.90**	1.05	**1.0**	1.22	1.03
Balance scale	98.30	17.70	40.10	**11.10**	2.51	**1.0**	2.85	2.91
TAE	44.50	20.90	18.30	**3.90**	1.48	**1.0**	2.69	1.64
Glass	41.60	15.50	15.40	**7.20**	2.0	**1.01**	2.83	2.33
Heart	9.20	**5.60**	12.60	**5.60**	1.83	**1.08**	1.73	1.86
Hepatitis	7.30	**3.90**	11.60	4.60	2.55	1.11	1.70	**1.0**

The results are obtained using 10-fold cross validation. The best values of each row are shown in bold

Table 5 Average number of ant runs per execution of REPEAT-UNTIL loop

Dataset	Avg. ants/ execution	Dataset	Avg. ants/ execution
Breast cancer w	166	Iris	77
Wine	80	Balance scale	148
Credit (Australia)	175	TAE	108
Credit (Germany)	349	Glass	137
Car	146	Heart	216
Tic-tac-toe	308	Hepatitis	259

original algorithm. In our opinion, the prior commitment of class label focuses the search (all the ants in an iteration are searching for the best version of the same rule) resulting in more appropriate rules.

4.4 Termination of rule construction

In our algorithm, terms are added to a rule until all the samples covered by it have the same class label or until there are no more terms to be added. There is no restriction that a rule should cover a minimum number of samples. A constructed rule might cover only one sample. In this section, we report the results of an experiment in which we impose a restriction that a rule being created should cover at least a minimum number of samples. We set this threshold as 10, a value used in [12] for the same purpose. A term is added to a rule only if the rule still covers at least 10 samples after its addition. As a result, all the constructed rules cover a minimum of ten samples. The results of this experiment are shown in Table 7. The predictive accuracy decreases for most of the data sets. The reason is that when we restrict a rule in this way, we may miss a very effective

Table 6 Results obtained by first constructing rule antecedent and then choosing class label

Dataset	Posterior class commitment			Prior class commitment		
	Accuracy	#R	#T/R	Accuracy	#R	#T/R
Breast cancer w	95.36 ± 1.65	**14.8**	**1.12**	**97.54 ± 0.98**	18.60	1.45
Wine	95.88 ± 4.84	4.2	**1.39**	**98.24 ± 2.84**	**4.10**	1.65
Credit (Australia)	85.80 ± 4.58	**7.67**	**1**	**89.42 ± 4.21**	7.80	1.58
Credit (Germany)	70.53 ± 3.40	**9.2**	**1.34**	**73.64 ± 2.67**	10.0	1.71
Car	95.58 ± 1.20	**46.2**	**2.05**	**98.02 ± 0.96**	57.60	2.49
Tic-tac-toe	89.79 ± 2.98	**12.8**	**1.77**	**100 ± 0.0**	14.80	2.50
Iris	96.00 ± 4.66	11	**1**	**97.33 ± 4.66**	**10.90**	1.05
Balance scale	83.23 ± 4.18	**56.4**	**1.59**	**86.61 ± 6.18**	98.30	2.51
TAE	73.33 ± 7.70	**44.2**	**1.01**	**77.33 ± 10.5**	44.50	1.48
Glass	65.71 ± 8.92	**36.90**	**1.09**	**74.29 ± 6.43**	41.60	2.0
Heart	74.44 ± 7.70	**8.70**	**1.01**	**77.78 ± 5.79**	9.20	1.83
Hepatitis	85.33 ± 4.30	**4.30**	**1.05**	**87.33 ± 9.66**	7.30	2.55

The best values of each row are shown in bold

Table 7 Results obtained by imposing restriction that a constructed rule must cover a minimum of 10 samples

Dataset	Condition of minimum rule coverage			No minimum rule coverage		
	Accuracy	#R	#T/R	Accuracy	#R	#T/R
Breast cancer w	96.64 ± 2.06	**15.33**	**1.14**	**97.54 ± 0.98**	18.60	1.45
Wine	**98.24 ± 2.84**	4.70	**1.49**	**98.24 ± 2.84**	**4.10**	1.65
Credit (Australia)	85.94 ± 2.56	**4.20**	**1.31**	**89.42 ± 4.21**	7.80	1.58
Credit (Germany)	69.65 ± 3.67	**7.46**	**1.43**	**73.64 ± 2.67**	10.0	1.71
Car	96.57 ± 1.04	**52.90**	2.57	**98.02 ± 0.96**	57.60	**2.49**
Tic-tac-toe	91.89 ± 4.41	**11.90**	**2.22**	**100 ± 0.0**	14.80	2.50
Iris	96.00 ± 5.62	**9.60**	1.08	**97.33 ± 4.66**	10.90	**1.05**
Balance scale	80.48 ± 4.33	**45.40**	**1.65**	**86.61 ± 6.18**	98.30	2.51
TAE	74.00 ± 11.90	**28.90**	**1.09**	**77.33 ± 10.5**	44.50	1.48
Glass	58.10 ± 11.40	**17.70**	**1.93**	**74.29 ± 6.43**	41.60	2.0
Heart	**80.74 ± 6.49**	**6.60**	**1.48**	77.78 ± 5.79	9.20	1.83
Hepatitis	85.33 ± 11.67	**3.30**	**1.72**	**87.33 ± 9.66**	7.30	2.55

The best values of each row are shown in bold

Table 8 Result of pruning each constructed rule and pruning only the best rule

Dataset	Pruning all generated rules			Pruning only the best rule		
	Accuracy	#R	#T/R	Accuracy	#R	#T/R
Breast cancer w	94.06 ± 3.95	**13.10**	**1.21**	**97.54 ± 0.98**	18.60	1.45
Wine	94.71 ± 5.15	4.40	**1.55**	**98.24 ± 2.84**	**4.10**	1.65
Credit (Australia)	84.93 ± 4.54	**4.0**	**1.43**	**89.42 ± 4.21**	7.80	1.58
Credit (Germany)	73.54 ± 3.15	**4.40**	**1.29**	**73.64 ± 2.67**	10.0	1.71
Car	96.51 ± 2.33	**46.80**	2.52	**98.02 ± 0.96**	57.60	**2.49**
Tic-tac-toe	86.21 ± 5.07	**10.40**	**1.87**	**100 ± 0.0**	14.80	2.50
Iris	95.33 ± 4.50	**8.90**	**1.04**	**97.33 ± 4.66**	10.90	1.05
Balance scale	**87.26 ± 4.65**	**96.20**	**2.46**	86.61 ± 6.18	98.30	2.51
TAE	72.0 ± 14.67	45.70	1.52	**77.33 ± 10.5**	**44.50**	**1.48**
Glass	**78.10 ± 12.13**	**37.10**	**1.98**	74.29 ± 6.43	41.60	2.0
Heart	**80.0 ± 6.81**	**5.0**	**1.43**	77.78 ± 5.79	9.20	1.83
Hepatitis	81.33 ± 10.33	**4.20**	**2.09**	**87.33 ± 9.66**	7.30	2.55

The best values of each row are shown in bold

Table 9 Comparison of results with and without pruning of redundant rules from the rule set

Dataset	Pruning of rule set			No pruning of rule set		
	Accuracy	#R	#T/R	Accuracy	#R	#T/R
Breast cancer w	97.39 ± 1.78	**13.20**	**1.26**	**97.54 ± 0.98**	18.60	1.45
Wine	**98.24 ± 2.84**	**3.80**	**1.55**	**98.24 ± 2.84**	4.10	1.65
Credit (Australia)	86.67 ± 3.97	**3.10**	**1.45**	**89.42 ± 4.21**	7.80	1.58
Credit (Germany)	72.12 ± 3.76	**8.60**	2.06	**73.64 ± 2.67**	10.0	**1.71**
Car	97.85 ± 1.16	**48.20**	**2.48**	**98.02 ± 0.96**	57.60	2.49
Tic-tac-toe	**100 ± 0.0**	**8.90**	2.61	**100 ± 0.0**	14.80	**2.50**
Iris	**98.00 ± 3.22**	**6.50**	**1.03**	97.33 ± 4.66	10.90	1.05
Balance scale	**87.58 ± 5.49**	**58.10**	**2.26**	86.61 ± 6.18	98.30	2.51
TAE	**78.00 ± 13.68**	**30.90**	1.55	77.33 ± 10.5	44.50	**1.48**
Glass	**74.76 ± 6.75**	**37.20**	**1.99**	74.29 ± 6.43	41.60	2.0
Heart	77.78 ± 6.49	**6.90**	1.98	**77.78 ± 5.79**	9.20	**1.83**
Hepatitis	**88.0 ± 10.11**	**6.90**	**2.52**	87.33 ± 9.66	7.30	2.55

The best values of each row are shown in bold

term in the rule that may increase the accuracy of the rule. Another problem with this approach is that there are different data sets and each has different number of dimensions and number of samples. It is difficult to decide the minimum number of samples for a given data set. Our experiment, though limited, provides some evidence that the strategy of rule construction by adding terms without any restriction is effective. However, we note that an advantage of the restriction is that we have less number of rules.

4.5 Rule pruning: all rules versus only the best rule

In this experiment, we prune each rule constructed by the ants prior to pheromone update; all other parameters are kept the same. The results of this experiment are shown in Table 8. The predictive accuracy is less for nine out of twelve sets, but the number of rules generated is less. The number of terms per rule is almost the same.

The reason for the dip in accuracy is that there are many rules whose construction is stopped due to homogeneity of class label of samples covered. These rules are already compact and any attempt at pruning causes non-homogeneity. The criterion for assessing quality of a rule (Eq. 10) has two components: confidence and coverage. While attempting to prune those rules that cover the samples of homogenous class labels the confidence decreases but that may be compensated by an increase in coverage. A rule may thus get pruned on the basis of Eq. 10 with lower confidence but better coverage than before. Such pruned rules cause pheromone increase in terms that may be misleading for future ants. Also, more importantly, the final best rule selected from the pruned rules may not have the same discriminatory capability as the final rule selected from unpruned rules.

4.6 Rule set pruning

This experiment is for testing the effect of rule set pruning (Sect. 3.6). The algorithm is run with and without the procedure, and the results are shown in Table 9. From the table, we can see that the accuracy sometimes improves and sometimes decreases. However, the number of rules consistently decreases if pruning is used. The remaining rules are simpler, and the average number of terms/rule also decreases in most of the cases.

4.7 Parameter settings

AntMiner-C has following user-defined parameters.

- number of ants,
- number of rules converged (for early termination of REPEAT-UNTIL loop),
- number of uncovered samples (for termination of WHILE loop),
- the powers of pheromone and heuristic values (α, β) in the probabilistic selection Eq. 3, and
- the pheromone evaporation rate (used in Eq. 11).

According to our experiments (Sect. 4.2), the number of 1,000 ants is an adequate value because it is used with the early convergence option of the REPEAT-UNTIL loop. Furthermore, the number of rules converged is an indication of pheromone saturation and does not seem to be a sensitive parameter as long as it is not a very small number. Likewise, the number of uncovered samples used for termination of WHILE loop is used to avoid rules with very low coverage and does not seem sensitive if it is restricted to a small value. The relationship between α, β, and ρ is complex and needs to be analyzed experimentally.

Table 10 One fold accuracy results for different combinations of ρ, α, and β

ρ, α, β	BCW	Wine	Cr-A	Cr-G	Car	T-t-t	Iris	Bal. S	TAE	Glass	Heart	Hep.
0, 3, 1	95.71	94.44	85.51	71.0	96.26	96.88	93.33	85.71	62.50	68.18	81.48	81.25
0.05, 3, 1	97.14	94.44	86.96	74.0	98.26	98.96	93.33	85.71	71.25	72.73	88.89	85.0
0.10, 3, 1	95.71	88.88	75.36	72.0	98.26	85.42	93.33	87.30	62.50	72.73	77.78	81.25
0.15, 3, 1	97.57	**100.0**	84.05	70.0	95.95	97.92	86.67	88.89	75.0	72.73	88.89	87.50
0.20, 3, 1	94.29	**100.0**	86.96	75.0	97.57	89.58	93.33	85.71	68.75	68.18	85.19	**93.75**
0, 2, 1	97.57	**100.0**	85.51	76.0	96.26	**100.0**	93.33	88.89	68.75	63.64	85.19	75.0
0.05, 2, 1	95.71	**100.0**	84.06	74.0	97.11	**100.0**	93.33	87.30	75.0	72.73	77.78	81.25
0.10, 2, 1	95.71	94.44	85.51	71.0	98.26	95.83	86.67	84.13	75.0	63.64	81.48	75.0
0.15, 2, 1	95.41	**100.0**	86.96	73.0	97.69	**100.0**	93.33	84.13	81.25	72.73	88.89	**93.75**
0.20, 2, 1	97.14	94.44	85.51	75.0	98.27	98.96	86.67	82.54	68.75	63.64	81.48	**93.75**
0, 1, 1	94.29	94.44	88.41	75.0	97.11	**100.0**	93.33	82.54	62.50	72.73	85.19	81.25
0.05, 1, 1	97.57	**100.0**	86.96	75.0	98.26	**100.0**	86.67	90.48	68.75	77.27	77.68	**93.75**
0.10, 1, 1	95.41	**100.0**	88.41	**78.0**	97.11	**100.0**	93.33	84.13	75.0	72.73	85.19	**93.75**
0.15, 1, 1	97.14	**100.0**	86.96	74.0	97.69	**100.0**	93.33	87.30	81.25	63.64	85.19	87.50
0.20, 1, 1	95.71	**100.0**	88.41	69.0	98.26	100.0	86.67	82.54	68.75	63.64	88.89	87.50
x, 0, 1	94.29	94.44	85.51	72.0	96.26	98.96	93.33	88.89	71.25	68.18	85.19	85.50
0, 1, 2	97.57	94.44	85.71	72.0	96.26	**100.0**	86.67	88.89	75.0	63.64	81.48	87.50
0.05, 1, 2	95.71	**100.0**	84.06	76.0	98.26	98.96	93.33	87.30	75.0	72.73	88.89	**93.75**
0.10, 1, 2	**98.57**	**100.0**	88.43	**78.0**	97.69	**100.0**	86.67	80.95	68.75	77.27	88.89	87.50
0.15, 1, 2	**98.57**	**100.0**	88.41	75.0	**98.84**	**100.0**	93.33	88.89	75.0	72.73	85.19	**93.75**
0.20, 1, 2	97.14	**100.0**	85.51	72.0	97.69	100.0	86.67	87.30	75.0	63.64	88.89	87.50
0, 1, 3	95.71	**100.0**	88.41	74.0	**98.84**	**100.0**	93.33	88.89	75.0	63.64	85.19	**93.75**
0.05, 1, 3	**98.57**	**100.0**	88.41	74.0	95.95	**100.0**	86.67	90.48	81.25	63.64	85.19	87.50
0.10, 1, 3	**98.57**	**100.0**	88.41	76.0	**98.84**	**100.0**	**100.0**	90.48	81.25	72.73	**92.59**	87.50
0.15, 1, 3	**98.57**	**100.0**	**91.30**	**78.0**	**98.84**	**100.0**	**100.0**	**92.06**	**87.50**	77.27	**92.59**	**93.75**
0.20, 1, 3	97.14	**100.0**	86.96	70.0	**98.84**	**100.0**	93.33	85.71	75.0	**81.83**	85.19	81.25
0, 1, 4	97.14	94.44	88.41	74.0	98.26	98.96	86.67	82.54	75.0	77.27	**92.59**	**93.75**
0.05, 1, 4	95.71	**100.0**	84.06	**78.0**	**98.84**	98.96	86.67	87.30	68.75	59.09	85.19	87.50
0.10, 1, 4	97.41	**100.0**	75.36	71.0	97.69	**100.0**	**100.0**	87.30	71.25	63.64	88.89	87.50
0.15, 1, 4	97.41	**100.0**	85.51	73.0	97.69	**100.0**	93.33	87.30	**87.50**	77.27	74.07	75.0
0.20, 1, 4	**98.57**	**100.0**	75.36	70.0	98.27	**100.0**	86.67	82.54	81.25	59.09	81.48	87.50

The best values of each column are shown in bold

In order to get some indication of the influence of α, β, and ρ on the performance of AntMiner-C, different combinations of these parameters are used, and the accuracy results are reported in Table 10. The results of Table 10 are obtained by running AntMiner-C for one fold only. The results are highest when $\rho = 0.15$, $\alpha = 1$, and $\beta = 3$.

4.8 Number of probability calculations

An important aspect of AntMiner-C is its potential for extracting rules from high-dimensional data sets in reasonable time. The nature of the heuristic function (Eq. 4) combined with the fact that the class is selected prior to the rule construction reduces the search space considerably. It assigns the value of zero to all terms that do not occur together for a given class in the training samples of the data set. The heuristic matrix is calculated once and remains fixed for one complete execution of the REPEAT-UNTIL loop.

The most costly computation in the algorithm, which is done repeatedly, is the probability equation (Eq. 3). The counter of its utilization gives a direct indication of the amount of search space that an ant has had to consider. We present some probability equation usage counts in Fig. 5 that provide an indication of the reduction in search space of AntMiner-C and also highlight its performance improvement over AntMiner. The sets used in this limited experiment are Iris, Wine, and Dermatology with 4, 13, and 33 attributes, respectively. The experiment was run for one fold only. The ratios of overall probability calls of

Fig. 5 Probability equation usage counts for Iris, Wine, and Dermatology with 4, 13, and 33 attributes, respectively. The experiment was run for one fold only. The results provide an indication of the reduction in search space of AntMiner-C and also highlight its performance improvement over AntMiner

Iris Dataset (4 attributes, 150 instances, 3 classes)

AntMiner

Rule #1:	Ants Used = 49,	Total probability equation calls = 1255
Rule #2:	Ants Used = 58,	Total probability equation calls = 1674
Rule #3:	Ants Used = 58,	Total probability equation calls = 1549
Rule #4:	Ants Used = 40,	Total probability equation calls = 1150
Rule #5:	Ants Used = 43,	Total probability equation calls = 1288
Rule #6:	Ants Used = 24,	Total probability equation calls = 755
Rule #7:	Ants Used = 11,	Total probability equation calls = 372

Accuracy: 86.66, Total number of rules found = 7, Number of terms/rule: 1, Grand total of probability equation calls = 9193

AntMiner-C

Rule #1:	Ants Used = 51,	Total probability equation calls = 73
Rule #2:	Ants Used = 66,	Total probability equation calls = 111
Rule #3:	Ants Used = 33,	Total probability equation calls = 41
Rule #4:	Ants Used = 71,	Total probability equation calls = 107
Rule #5:	Ants Used = 65,	Total probability equation calls = 83
Rule #6:	Ants Used = 49,	Total probability equation calls = 58
Rule #7:	Ants Used = 72,	Total probability equation calls = 142
Rule #8:	Ants Used = 71,	Total probability equation calls = 168
Rule #9:	Ants Used = 40,	Total probability equation calls = 47

Accuracy: 93.33, Total number of rules found = 9, No of terms / rule: 1, Grand total of probability equation calls = 830

Wine Dataset (13 attributes, 178 instances, 3 classes)

AntMiner

Rule #1:	Ants Used = 38,	Total probability equation calls = 1023
Rule #2:	Ants Used = 17,	Total probability equation calls = 477
Rule #3:	Ants Used = 230,	Total probability equation calls = 5646

Accuracy: 94.44, Total number of rules found = 3, Number of terms/rule: 2, Grand total of probability equation calls = 7146

AntMiner-C

Rule #1:	Ants Used = 74,	Total probability equation calls = 178
Rule #2:	Ants Used = 31,	Total probability equation calls = 51
Rule #3:	Ants Used = 120,	Total probability equation calls = 587
Rule #4:	Ants Used = 137,	Total probability equation calls = 362

Accuracy: 100, Total number of rules found = 4, No of terms / rule: 1.25, Grand total of probability equation calls = 1178

Dermatology Dataset (33 attributes, 366 instances, 6 classes)

AntMiner

Rule #1:	Ants Used = 25,	Total probability equation calls = 2813
Rule #2:	Ants Used = 26,	Total probability equation calls = 3083
Rule #3:	Ants Used = 504,	Total probability equation calls = 58656
Rule #4:	Ants Used = 1000,	Total probability equation calls = 117240
Rule #5:	Ants Used = 131,	Total probability equation calls = 15700
Rule #6:	Ants Used = 21,	Total probability equation calls = 2542
Rule #7:	Ants Used = 11,	Total probability equation calls = 1347

Accuracy: 80.48, Total number of rules found = 7, Number of terms/rule: 1.88, Grand total of probability equation calls = 201,381

AntMiner-C

Rule #1:	Ants Used = 134,	Total probability equation calls = 377
Rule #2:	Ants Used = 47,	Total probability equation calls = 156
Rule #3:	Ants Used = 80,	Total probability equation calls = 121
Rule #4:	Ants Used = 467,	Total probability equation calls = 4268
Rule #5:	Ants Used = 104,	Total probability equation calls = 259
Rule #6:	Ants Used = 45,	Total probability equation calls = 82
Rule #7:	Ants Used = 68,	Total probability equation calls = 121
Rule #8:	Ants Used = 65,	Total probability equation calls = 130
Rule #9:	Ants Used = 550,	Total probability equation calls = 4348
Rule #10:	Ants Used = 162,	Total probability equation calls = 684
Rule #11:	Ants Used = 188,	Total probability equation calls = 927
Rule #12:	Ants Used = 123,	Total probability equation calls = 336
Rule #13:	Ants Used = 42,	Total probability equation calls = 61
Rule #14:	Ants Used = 87,	Total probability equation calls = 167
Rule #15:	Ants Used = 90,	Total probability equation calls = 165
Rule #16:	Ants Used = 50,	Total probability equation calls = 146
Rule #17:	Ants Used = 654,	Total probability equation calls = 5317
Rule #18:	Ants Used = 609,	Total probability equation calls = 6431

Accuracy: 97.29, Total number of rules found = 18, Number of terms/rule: 1.89, Grand total of probability equation calls = 24,096

AntMiner and AntMiner-C for the three data sets are 11.07, 6.06, and 8.35, respectively.

5 Conclusions and future work

Previously, ACO has been used to discover classification rules from data sets. This paper proposes a new algorithm whose main highlight is the use of a heuristic function based on the correlation between the recently added term and the term to be added in the rule. We check the performance of the algorithm on a suite of twelve data sets. The experimental results show that the proposed approach achieves higher accuracy rate and has fast convergence speed.

For future work, the optimal values of parameters can be investigated more thoroughly and some general guidelines may be proposed by analyzing the experimental results. Another direction of research can be to use a heuristic function that considers the correlation between all of the selected terms and the next term to be added. For this purpose, a new search space will have to be proposed.

Finally, we would like to refer to the knowledge fusion problem that deals with the cohesion of extracted knowledge with the domain knowledge provided by experts. The rules extracted by the algorithm for a given data set may not completely adhere to the available domain knowledge. There might be missing, redundant, and unjustifiable (misleading) rules. In [27], the AntMiner+ is extended to incorporate hard constraints of the domain knowledge by modifying the search space and the soft constraints by influencing the heuristic values. The same modifications are possible in AntMiner-C and seem to be an interesting direction for future study.

Acknowledgments The authors would like to thank the reviewers of this paper for their valuable comments.

References

1. Engelbrecht AP (2007) Computational intelligence, an introduction, 2nd edn. Wiley, Hoboken
2. Engelbrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, Hoboken
3. Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann Academic Press, Massachusetts
4. Dorigo M, Stützle T (2004) Ant colony optimization. MIT Press, Cambridge, MA
5. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 26:1
6. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the travelling salesman problem. IEEE Trans Evol Comput 1:1
7. Abraham A, Grosan C, Ramos V (2006) Swarm intelligence in data mining, studies in computational intelligence, vol 34. Springer, Berlin
8. Han J, Kamber M (2006) Data mining: concepts and techniques, 2nd edn. Morgan Kaufmann Publishers, Massachusetts
9. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, Massachusetts
10. Duda RO, Hart PE, Stork DG (2000) Pattern classification. Wiley, Hoboken
11. Eiben AE, Smith JE (2007) Introduction to evolutionary computing. Natural computing series, 2nd edn. Springer, Berlin
12. Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. IEEE Trans Evol Comput 6(4):321–332
13. Liu B, Abbass HA, McKay B (2002) Density-based heuristic for rule discovery with ant-miner. In: Proceedings 6th Australasia-Japan joint workshop on intelligent and evolutionary systems. Canberra, Australia, pp 180–184
14. Liu B, Abbass HA, McKay B (2003) Classification rule discovery with ant colony optimization. In: Proceedings IEEE/WIC international conference on intelligent agent technology, pp 83–88
15. Liu B, Abbass HA, McKay B (2004) Classification rule discovery with ant colony optimization. IEEE Comput Intell Bull 3:1
16. Martens D, de Backer M, Haesen R, Vanthienen J, Snoeck M, Baesens B (2007) Classification with ant colony optimization. IEEE Trans Evol Comput 11:5
17. Smaldon J, Freitas AA (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
18. Holden N, Freitas AA (2007) A hybrid PSO/ACO algorithm for classification. In: Proceedings GECCO-2007 workshop on particle swarms: the second decade. ACM Press, New York
19. Swaminathan S (2006) Rule induction using ant colony optimization for mixed variable attributes. MSc Thesis, Texas Tech University, Lubbock
20. Otero F, Freitas A, Johnson C (2008) cAnt-miner: an ant colony classification algorithm to cope with continuous attributes. In: Ant colony optimization and swarm intelligence (ANTS 2008), LNCS 5217. Springer, Berlin
21. Chan A, Freitas A (2006) A new ant colony algorithm for multi-label classification with applications in bioinformatics. In: Proceedings genetic and evolutionary computation conference (GECCO-2006)
22. Boryczka U, Kozak J (2009) New Algorithms for generation decision trees—Ant-miner and its modifications. In: Foundations of computational intelligence, vol 6. Book series—Studies in computational intelligences, vol 206. Springer, New York
23. Hettich S, Bay SD (1996) The UCI KDD archive. Department of Information Computer Science, University California, Irvine, CA, [Online]. Available: http://kdd.ics.uci.edu
24. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, Massachusetts
25. Quinlan JR (1987) Generating production rules from decision trees. In: Proceedings international joint conferences on artificial intelligence. San Francisco, USA
26. Vapnik VN (1995) The nature of statistical learning theory. Springer, New York
27. Martens D, de Backer M, Haesen R, Baesens B, Mues C, Vanthienen J (2006) Ant-based approach to the knowledge fusion problem. In: Ant colony optimization and swarm intelligence (ANTS 2006), LNCS 4150. Springer, Berlin