Handling Continuous Attributes in Ant Colony Classification Algorithms

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

***Abstract*— Most real-world classification problems involve continuous (real-valued) attributes, as well as, nominal (discrete) attributes. The majority of Ant Colony Optimisation (ACO) classification algorithms have the limitation of only being able to cope with nominal attributes directly. Extending the approach for coping with continuous attributes presented by cAnt-Miner (Ant-Miner coping with continuous attributes), in this paper we propose two new methods for handling continuous attributes in ACO classification algorithms. The first method allows a more flexible representation of continuous attributes' intervals. The second method explores the problem of attribute interaction, which originates from the way that continuous attributes are handled in cAnt-Miner, in order to implement an improved pheromone updating method. Empirical evaluation on eight publicly available data sets shows that the proposed methods facilitate the discovery of more accurate classification models.**

I. Introduction

THE classification task in data mining aims at predicting the value of a given goal attribute for an example, based on the values of a set of predictor attributes for that example [1]. Since real-world classification problems are generally described by nominal (discrete) and continuous (real-valued) attributes, classification algorithms are required to be able to cope with both nominal and continuous attributes in order to build a classification model. However, most Ant Colony Optimisation (ACO) [2] classification algorithms have the limitation of being able to cope with only nominal attributes. Continuous attributes, if present, need to be transformed into nominal attributes, by creating discrete intervals in a preprocessing step. There are potentially two drawbacks by not coping with continuous attributes directly. Firstly, there is a need for a discretisation procedure in a preprocessing step. Secondly, less information is available to the classification algorithm, since the discretisation procedure creates a fixed number of discrete intervals for each continuous attribute.

In this paper, we focus on extending the ideas of cAnt-Miner [3] in coping with continuous attributes. cAnt-Miner pioneered in coping with both types (nominal and continuous) attributes directly, taking full advantage of all continuous attributes' information and not requiring a discretisation procedure in a preprocessing step. We propose two new methods for handling continuous attributes in ACO classification algorithms. The first method gives the ability to use continuous attributes intervals with lower and upper bound values (i.e. $v_{lower} \leq attribute < v_{upper}$). The second method explores the attribute interaction introduced by the way that continuous attributes are handled internally, which is not taken into account by cAnt-Miner. We present empirical evaluation results for validating the proposed methods.

The remainder of this paper is organised as follows. Section II presents a brief overview of Ant-Miner [4], the first ACO classification algorithm, and cAnt-Miner. Section III discusses the proposed two new methods for handling continuous attributes. The empirical evaluation results are presented in Section IV. Finally, Section V draws the conclusions of the paper and present future research directions.

The authors are with the Computing Laboratory, University of Kent, Canterbury, Kent, United Kingdom (email: {febo2, A.A.Freitas, C.G.Johnson}@kent.ac.uk).

II. Background

Ant Colony Optimisation (ACO) systems simulate the behaviour of real ants using a colony of artificial ants, which cooperate in finding good solutions to optimization problems. Each artificial ant, representing a simple agent in the system, builds candidate solutions to the problem at hand and communicates indirectly with other artificial ants by means of pheromone levels. At the same time that ants perform a global search for new solutions, the search is guided to better regions of the search space based on the quality of solutions found so far. The system converges to good solutions as a result of the collaborative interaction among the ants. The interactive process of building candidate solutions and updating pheromone values allows an ACO algorithm to converge to optimal or near-optimal solutions.

In the context of discovering classification rules in data mining, ACO algorithms have been successfully applied to several different classification problems [5]. In this section, we present an overview of Ant-Miner, the first implementation of an ACO algorithm for the classification task of data mining [4], and cAnt-Miner [3], the first (to the best of our knowledge) ACO classification algorithm able to cope with continuous attributes without requiring a discretisation procedure in a preprocessing step.

A. Ant-Miner Overview

Ant-Miner aims at extracting *IF-THEN* classification rules of the form *IF* $(term_1)$ *AND* $(term_2)$ *AND ... AND* $(term_n)$ *THEN* $(class)$ from data. Each term in the rule is a triple $(attribute, operator, value)$, where *operator* represents a relational operator and *value* represents a value of the domain of *attribute* (e.g. $sex = male$). The *IF* part corresponds to the rule's antecedent and the *THEN* part corresponds to the rule's consequent, which represents the class to be predicted by the rule. An example that satisfies the rule's antecedent

will be assigned the class predicted by the rule. As Ant-Miner only works with nominal (categorical or discrete) attributes, the only valid relational operator is "=" (equality operator). Continuous attributes need to be discretised in a preprocessing step.

A high level pseudo-code of Ant-Miner is presented in Algorithm 1 [4]. In summary, Ant-Miner works as follows. It starts with an empty rule list and iteratively (*while* loop) adds one rule at a time to that list while the number of uncovered training examples is greater than a user-specified maximum value. In order to construct rules, ants start with an empty rule (no terms in its antecedent) and add one term at a time to their rule antecedent (*repeat-until* loop). Terms are probabilistically chosen to be added to current partial rules based on the values of the amount of pheromone (τ) and a problem-dependent heuristic information (η) associated with terms (vertices in the construction graph). A pheromone value and a heuristic value are associated with each possible term – i.e. each possible triple ($attribute$, $operator$, $value$). As usual in ACO, heuristic values are fixed (based on an information theoretical measure of the predictive power of the term), while pheromone values are iteratively updated based on the quality of the rules built by ants. Ants keep adding a term to their partial rule until any term added to their rule's antecedent would make their rule cover less training examples than a user-specified threshold (in order to avoid too specific and unreliable rules), or all attributes have already been used. The latter rule construction stopping criterion is necessary because an attribute can only occur once in the antecedent of a rule, in order to avoid inconsistencies such as $<sex = male$ AND $sex = female>$. Once the rule construction process has finished, the rule constructed by an ant is pruned to remove irrelevant terms from the rule antecedent. Then, the consequent of a rule is chosen to be the class value most frequent among the set of training examples covered by the rule in question. Finally, pheromone trails are updated using the best rule, based on a quality measure Q, created by ants. The process of constructing a rule is repeated until a user-specified number of iterations has been reached, or the best rule of the current iteration is exactly the same as the best rule constructed by a predefined number of previous iterations, which works as a rule convergence test. The best rule found along this iterative process is added to the rule list and the covered training examples (training examples that satisfy the antecedent of the best rule) are removed from the training set.

Algorithm 1: High level pseudo-code of Ant-Miner.

```
begin Ant-Miner
    tr_set ← all training examples;
    rule_list ← ∅;
    while |tr_set| > MaxUncoveredExamples do
        τ ← initializes pheromones;
        rule_best ← ∅;
        repeat
            CreateRules();
            ComputeConsequents();
            PruneRules();
            current_best ← BestRule();
            UpdatePheromones(τ, current_best);
            if Q(current_best) > Q(rule_best) then
                rule_best ← current_best;
            end
            i ← i + 1;
        until i ≥ MaxIterations OR Convergence() ;
        rule_list ← rule_list + rule_best;
        tr_set ← tr_set \ CoveredExamples(rule_best);
    end
end
```

B. cAnt-Miner Overview

In order to overcome Ant-Miner's limitation of only coping with nominal attributes, Otero et al. [3] have proposed an Ant-Miner extension — named cAnt-Miner (Ant-Miner coping with continuous attributes) — which can dynamically create thresholds on continuous attributes' domain values during the rule construction process. Since cAnt-Miner has the ability of coping with continuous attributes "on-the-fly", continuous attributes do not need to be discretised in a preprocessing step. cAnt-Miner extended Ant-Miner in several ways, as follows.

Firstly, cAnt-Miner includes vertices to represent continuous attributes in the construction graph. For each nominal attribute x_i and value v_{ij} (where x_i is the i-th nominal attribute and v_{ij} is the j-th value belonging to the domain of x_i), a vertex ($x_i = v_{ij}$) is added to the construction graph, as in Ant-Miner. Furthermore, for each *continuous* attribute y_i, a vertex (y_i) is added to the construction graph, unlike in Ant-Miner. Note that continuous attributes vertices do not represent a valid term, since they do not have a relational operator and value associated in the construction graph, in contrast to nominal attributes. The relational operator and a threshold value will be determined when an ant selects a continuous attribute vertex as the next term to be added to the rule (an example of continuous attribute term is: '*age* > 21'). This makes the choice of a relational operator and value tailored to the current candidate rule being constructed, rather than chosen in a static preprocessing step.

Secondly, in order to compute the heuristic information for continuous attributes, cAnt-Miner incorporates a dynamic entropy-based discretisation procedure. In Ant-Miner, the heuristic value of each nominal vertex ($x_i = v_{ij}$) involves a measure of entropy associated with the partition of examples which have the specific v_{ij} value for the attribute x_i. The entropy measure, which is derived from information theory and is often used in data mining, quantifies the impurity of a collection of examples. Since continuous attribute vertices (y_i) do not represent a partition of examples as nominal attribute vertices, a threshold value v need to be selected in order to dynamically partition the set of examples into two intervals: $y_i < v$ and $y_i \geq v$. The best threshold value

v is the value v that minimizes the entropy of the partition, computed as

$$entropy(y_i, v) = \frac{|S_{y_i<v}|}{|S|} \cdot entropy(S_{y_i<v}) + \frac{|S_{y_i \geq v}|}{|S|} \cdot entropy(S_{y_i \geq v}), \quad (1)$$

where $|S_{y_i<v}|$ is the total number of examples in the partition $y_i < v$ (partition of training examples where the attribute y_i has a value less than v), $|S_{y_i \geq v}|$ is the total number of examples in the partition $y_i \geq v$ (partition of training examples where the attribute y_i has a value greater or equal to v) and $|S|$ is the total number of training examples. The values of $entropy(S_{y_i<v})$ and $entropy(S_{y_i \geq v})$ are computed as

$$entropy(T) = \sum_{c=1}^{k} -p(c\,|\,T) \cdot \log_2 p(c\,|\,T), \quad (2)$$

where $p(c\,|\,T)$ is the proportion of examples in T that have class c and k is the number of classes. After selection of the best threshold value v using Equation (1), the measure of entropy used to calculate the heuristic value of the continuous attribute vertex (y_i) corresponds to the minimum entropy value between the two generated intervals ($y_i < v$) and ($y_i \geq v$), according to Equation (2).

Thirdly, when a continuous attribute vertex (y_i) is selected by an ant to be added to its current partial rule, a relational operator and a value is computed using a similar procedure as for the heuristic information. The best threshold value v is selected using Equation (1), subject to the restriction of considering only examples covered by the current partial rule in the evaluation of threshold values. Then, the relational operator ('$<$' or '$\geq$') associated with the interval with the lowest entropy value is selected and a term in the form (y_i, *operator*, v) added to the ant's current partial rule (e.g. *age* < 18).

Fourthly, the pheromone updating procedure has been extended to cope with continuous attribute vertices. In the case of continuous attributes, pheromone values are associated with continuous attribute vertices not considering the operator and threshold value, that is, there is a single entry in the pheromone matrix for each continuous attribute, in contrast to multiple entry for nominal attributes — nominal attributes have an entry for every (x_i, v_{ij}) pair.

III. New Methods for Handling Continuous Attributes in Ant Colony Classification Algorithms

Extending on the ideas of cAnt-Miner, this paper presents two new methods for handling continuous attributes in Ant Colony Optimisation (ACO) classification algorithms. The first method enables the creation of intervals with both lower and upper bound values for continuous attributes, in the form of $v_{lower} \leq attribute < v_{upper}$, by incorporating a new dynamic discretisation procedure based on the Minimum Description Length (MDL) principle [6] in the rule construction process. The second method explores the fact that the discretisation of continuous attributes occurs when an ant selects a continuous attribute vertex to be added to its current partial rule. Since only examples covered by the current partial rule are considered for the threshold calculation, the previously selected vertices play an important role in the creation of discrete intervals. The order dependency between vertices in a rule characterizes an attribute interaction, which is not taken into account by cAnt-Miner.

A. MDL-based Discretisation

Fayyad and Irani [6] presented an MDL-based approach where multiple discrete intervals can be extracted by applying a binary discretisation procedure recursively, selecting the best threshold value at each iteration, and using the minimal description length principle as a stopping criterion to determine whether more threshold values should be introduced. The motivation for multiple interval discretisation lay in the fact that the 'interesting' value range may be an internal interval (e.g. $18 \leq age < 21$), which can not be easily generated by a binary-interval-at-a-time discretisation procedure. The MDL-based approach generally leads to coarse intervals in cases where the examples are homogeneously distributed (distributed in a few different class values) and to fine intervals in cases of more uniform distributions.

Following Fayyad and Irani, we have incorporated a MDL-based decision criterion to decide whether or not to split a given interval further in cAnt-Miner. The basic idea is to apply cAnt-Miner's entropy-based discretisation method recursively, relying on the MDL criterion to accept or reject a threshold value. In this way, instead of generating only intervals in the form $y_i < v$ and $y_i \geq v$, internal intervals in the form $v_{lower} \leq y_i < v_{upper}$ can be created (where v, v_{lower} and v_{upper} are values in the domain of the continuous attribute y_i).

The MDL-based discretisation method is divided in two steps, as follows. In the first step, the best threshold value v for a continuous attribute vertex (y_i) is selected as in the original cAnt-Miner — Equation (1). In the second step, the MDL decision criterion for accepting or rejecting a threshold value v is computed as

$$Gain(y_i, v; S) > \frac{log_2(|S| - 1)}{|S|} + \frac{\Delta(y_i, v; S)}{|S|}, \quad (3)$$

$$Gain(y_i, v; S) = entropy(S) - \frac{|S_{y_i<v}|}{|S|} \cdot entropy(S_{y_i<v}) - \frac{|S_{y_i \geq v}|}{|S|} \cdot entropy(S_{y_i \geq v}), \quad (4)$$

$$\Delta(y_i, v; S) = log_2(3^k - 2) - [k \cdot entropy(S) - k_{y_i<v} \cdot entropy(S_{y_i<v}) - k_{y_i \geq v} \cdot entropy(S_{y_i \geq v})], \quad (5)$$

where k, $k_{y_i<v}$ and $k_{y_i\geq v}$ are the number of different class values in S, $S_{y_i<v}$ and $S_{y_i\geq v}$, respectively. If the MDL criterion defined in Equation (3) is satisfied, the threshold value v for the continuous attribute y_i is accepted; otherwise it is rejected. Note that the entropy measures of S, $S_{y_i<v}$ and $S_{y_i\geq v}$ required to evaluate the threshold value v against the MDL criterion are already computed by the first step (threshold selection). Therefore, there is no increase in computational time to compute the MDL criterion. Finally, if the threshold value v is accepted, the discretisation procedure is repeated individually for the partitions $S_{y_i<v}$ and $S_{y_i\geq v}$.

At the end of the MDL discretisation procedure, we can have potentially multiple threshold values. In order to select the best threshold value(s), the list of threshold values is sorted and the entropy value for each discrete interval is calculated. Then, the interval with the lowest entropy value is selected (based on the fact that lower entropy values represent more "pure" partitions where most of the examples belong to a single class). If an internal interval is selected (an interval between two threshold values), a term in the form $v_j \leq y_i < v_{j+1}$ is generated; otherwise, a term in the form $y_i < v_j$ or $y_i \geq v_j$ is generated (where j is the j-th threshold value selected). Fig. 1 illustrates the intervals that could have been created by selecting two threshold values for a continuous attribute *age*.

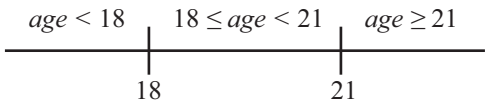

Fig. 1. Illustration of discrete intervals that that could have been created by selecting two threshold values for a continuous attribute *age*. At the end of the MDL discretisation procedure, the interval associated with the lowest entropy value is selected.

B. Encoding Attribute Interaction as Pheromone Levels: Associating Pheromones with Edges

In the original version of Ant-Miner, pheromone values are associated with vertices in the construction graph, where each vertex represents a term $x_i = v_{ij}$ (e.g. $sex = male$). Hence, both ants with rule antecedents $<sex = male$ AND $smoke = no>$ and $<smoke = no$ AND $sex = male>$ will deposit the same amount of pheromone on vertices '$sex = male$' and '$smoke = no$'.

cAnt-Miner employs the same pheromone update procedure, even though the order of attributes (vertices) could affect the threshold values of continuous attributes. Note that this attribute interaction dependency is not observed in the original Ant-Miner, since it only supports nominal attribute vertices which are represented by $<attribute = value>$ pairs and their order is irrelevant. For instance, a partial rule with a term $<sex = male>$ followed by a term $<smoke = no>$ has no difference is comparison with a partial rule with a term $<smoke = no>$ followed by a term $<sex = male>$. On the other hand, a partial rule with a term $<smoke = no>$ followed by a term representing the continuous attribute *age* is different in comparison with a partial rule with a term representing a continuous attribute *age* followed by a term $<smoke = no>$. In the first case, only examples covered by the partial rule $<smoke = no>$ are used to compute a threshold value for the continuous attribute *age* (e.g. $age < 10$). In the latter case, all examples of the training set are used to compute a threshold value for the continuous attribute *age* (e.g. $age \geq 25$) since the continuous attribute *age* is the first attribute to be added to the rule. Although the discretisation procedure is deterministic, there is no guarantee that the same values will be chosen since the examples used to compute the threshold values are different. This might affect the way that ants explore the construction graph, since the pheromone levels do not accurately reflect paths explored by previous ants.

A straightforward implementation to preserve the order is to associate pheromone on the edges instead of vertices. For instance, when updating pheromones levels for a rule $<smoke = no$ AND $age < 10>$, instead of depositing the pheromone on the vertices '$sex = male$' and 'age', the pheromone is deposited on the edge that connects the vertex '$sex = male$' to the vertex 'age'. Consequently, when updating pheromone levels for a rule $<age \geq 25$ AND $smoke = no>$, the pheromone is deposited on the edge that connects the vertex 'age' to the vertex '$smoke = no$'. Note that even though the construction graph is conceptually defined as a fully connected (bidirectional edges) graph, in practice there is one edge for each direction between each pair of vertices. To be able to associate pheromone levels to the first vertex of a rule, a dummy vertex '*start*' is added and unidirectionally connected to all vertices in the construction graph. This vertex represents the starting point for creating trails and its purpose is to associate pheromone values on the edge of the first attribute vertex of an ant's trail.

In order to calculate the probability of selecting a vertex to be added to the current partial rule, the pheromone value associated with the edge between the last vertex of the rule (or the dummy '*start*' vertex when the rule is empty) and the candidate vertex is used in combination with the heuristic value associated with the candidate vertex in the same manner as in Ant-Miner and cAnt-Miner. The main advantage of this method can be seen when dealing with continuous attributes. Since the dynamic discretisation procedure of continuous attributes is deterministic and based on the current covered examples, as discussed previously, using pheromone levels to represent the order in which ants select vertices to compose candidate rules indirectly preserves the threshold values of continuous attributes.

The idea of associating pheromone values with the edges of the construction graph have been previously explored in AntMiner+ by Martens et al. [7], although it was used in a very different context. The construction graph in AntMiner+ is defined as a direct acyclic graph (DAG). Therefore, the attribute order is explicitly determined by the construction graph DAG-structure, rather than by pheromone levels on

the edges between vertices as proposed in this work, which has the undesirable bias of restricting ants to create trails following a specific order of attributes.

As discussed in subsection II-A, after an ant completes the rule's construction process, the created rule undergoes a pruning procedure which aims at removing irrelevant terms that might have been included in the rule. The original pruning procedure employed by Ant-Miner and cAnt-Miner consists of removing one term at a time while this procedure improves the quality of the rule. Therefore, at each iteration of the pruning procedure, n (where n is the number of terms in the current rule) candidate rules with $n-1$ terms are evaluated and the rule with highest quality is selected for further pruning. This procedure is repeated until no term can be removed which improves the current rule quality or the current rule has only one term left. Clearly, the original pruning procedure does not take into account the order in which terms appear in a rule. For instance, a rule $<sex = male$ AND $age \geq 25$ AND $smoke = yes>$ can be pruned to $<age \geq 25$ AND $smoke = yes>$, which will update the pheromone levels considering 'age' as the first vertex of the rule followed by the vertex '$smoke = yes$'. This is not consistent with how the threshold of the attribute age was calculated and there is no guarantee that, if an ant selects the vertex 'age' as the first term of its rule, it will have the same threshold value ('25' in this case).

To avoid such inconsistencies, we proposed a new pruning procedure (dubbed 'threshold-aware' pruning) sensible to the order of attributes (vertices). Since the order of terms in a rule is consistent with continuous attributes threshold values, a simple implementation of a threshold-aware pruning is to remove the last term that was added to the rule in order to simplify the rule. The removal process is repeated until the rule quality decreases when the last term is removed or the rule has only one term left. Note that, in this procedure, the continuous attribute threshold values do not have to be re-calculated, since terms are removed in the reverse order that they were added to the rule. Also, only one candidate rule has to be evaluated at each iteration of the pruning procedure, resulting in a more efficient pruning procedure when compared to the original one employed by Ant-Miner and cAnt-Miner.

IV. Computational Results

The proposed cAnt-Miner extensions were evaluated using eight publicly available data sets from the UCI Irvine machine learning repository [8]. We selected data sets that contain at least one continuous attribute, since our goal is to evaluate the different proposed variations of cAnt-Miner, using the original Ant-Miner as a baseline method. We also include the results for J48 (Weka [9] implementation of the well-known C4.5 decision tree algorithm [10]) using the same data sets. We evaluated the proposed cAnt-Miner extensions individually, and also the combination of both, generating three variations of cAnt-Miner. Table I summarizes the different classifiers used in our experiments.

TABLE I
SUMMARY OF THE CLASSIFIER METHODS USED IN OUR EXPERIMENTS.

Classifier	Description
Ant-Miner	original Ant-Miner version
cAnt-Miner	original cAnt-Miner version
cAnt-Miner-MDL	MDL-based cAnt-Miner
cAnt-Miner2	cAnt-Miner using pheromones on the edges
cAnt-Miner2-MDL	MDL-based cAnt-Miner2
J48	Weka's C4.5 implementation

TABLE II
SUMMARY OF THE DATA SETS USED IN OUR EXPERIMENTS.

Data set	Attributes		Classes	Size	
	Nominal	Continuous		Original	Discrete
wdbc	0	30	2	569	366
crx	9	6	2	690	639
hepatitis	13	6	2	155	116
glass	0	9	7	213	119
ionosphere	0	34	2	350	292
wine	0	13	3	178	126
australian	8	6	2	690	637
heart	6	7	2	270	232

The experiments were conducted using a 10-fold cross-validation procedure [9]. For stochastic classifiers, i.e. Ant-Miner and cAnt-Miner variations, we run the classifier 10 times — using a different random seed to initialise the search each time — for each cross-validation fold. In the case of the deterministic J48 classifier, it is run just once for each cross-validation fold. Since the original version of Ant-Miner does not cope with continuous attributes directly, the data sets were discretised in a preprocessing step. For each cross-validation fold, we separately discretised (using the C4.5-Disc discretisation method [11]) the training set and the created discrete intervals were used to discretise the test set. This separation is necessary because, if we had discretised the entire dataset before creating the cross-validation folds, the discretisation method would had access to the test data. This would have compromised the reliability of the experiments. Moreover, we also removed the duplicated examples (examples with the same values for all attributes) from the resulting discrete dataset to avoid the possibility that a test set contains an example that is the same as a training example.

Table II presents the summary of the data sets used in our experiments. In Table II, the first column gives the data set name, the second and third columns give the number of nominal and continuous attributes respectively, the fourth column gives the number of classes, the fifth column gives the number of examples in the original data set and the sixth column gives the number of examples in the discrete data set (after the removal of duplicated examples). Recall that only the original Ant-Miner used the discrete data sets.

Table III shows the predictive accuracy achieved by each

TABLE III

PREDICTIVE ACCURACY (*mean* ± *standard deviation*) AFTER THE 10-FOLD CROSS-VALIDATION PROCEDURE.

Data set	Ant-Miner	cAnt-Miner	cAnt-Miner-MDL	cAnt-Miner2	cAnt-Miner2-MDL	J48
wdbc	90.39 ± 1.42	**93.88 ± 0.70**	93.30 ± 0.90	**93.94 ± 0.62**	**93.64 ± 0.65**	92.63 ± 1.04
crx	83.21 ± 0.91	85.12 ± 0.92	85.11 ± 0.96	85.79 ± 0.97	**85.90 ± 0.91**	85.29 ± 0.89
hepatitis	70.17 ± 4.76	**84.92 ± 4.23**	**84.74 ± 3.86**	**84.74 ± 3.38**	**82.53 ± 3.66**	82.15 ± 4.27
glass	48.08 ± 2.22	**64.60 ± 2.31**	**62.85 ± 1.95**	**66.72 ± 1.88**	**66.81 ± 2.07**	**62.49 ± 4.45**
ionosphere	88.45 ± 1.79	86.60 ± 1.10	87.17 ± 1.32	86.14 ± 1.31	87.60 ± 1.23	88.57 ± 1.65
wine	83.23 ± 2.06	**89.25 ± 1.16**	**88.73 ± 1.28**	**92.27 ± 1.10**	**89.50 ± 1.43**	**93.30 ± 1.61**
australian	83.41 ± 1.09	84.87 ± 1.02	84.51 ± 1.04	84.48 ± 1.12	84.45 ± 1.30	85.07 ± 1.06
heart	74.77 ± 3.16	74.67 ± 2.26	75.67 ± 1.75	76.89 ± 1.92	77.37 ± 2.18	78.15 ± 2.67

classifier on the data sets used in our experiments. Each entry in the table corresponds to the average value of accuracy obtained using the 10-fold cross-validation procedure, followed by the standard deviation. In addition, an entry is shown in bold if, for the corresponding data set, the accuracy obtained by the corresponding classifier was significantly greater than the accuracy achieved by Ant-Miner for that data set — according to a two-tailed Student's t-test with 95% confidence.

The results on Table III show that all variations of cAnt-Miner are able to obtain significant improvements over Ant-Miner (the baseline classifier in our experiments) in at least three out of eight data sets, namely hepatitis, glass and wine. The use of the MDL-based discretisation isolated was not effective in the wdbc data set, where cAnt-Miner-MDL obtained a higher accuracy value that was not statistically significant over Ant-Miner. The remaining three variations of cAnt-Miner obtained significant improvements over Ant-Miner in the wdbc data set. Furthermore, the combination of both proposed extensions enabled cAnt-Miner2-MDL to obtain significant improvements over Ant-Miner in the crx data set. On the other hand, J48 significantly outperformed Ant-Miner only in two data sets, namely glass and wine.

The results obtained in the experiments can be summarizes as follows. Overall, cAnt-Miner2-MDL is the most accurate classifier method compared to the baseline Ant-Miner classifier. In five of eight data sets, cAnt-Miner2-MDL significantly outperformed Ant-Miner. In the remaining three data sets, there were no significant differences between cAnt-Miner2-MDL and Ant-Miner. Although there are no significant differences between cAnt-Miner variations and J48, it should be noted that J48 was able to significantly outperform the baseline Ant-Miner only in two out of eight data sets.

V. CONCLUSIONS

In this paper, we have presented two new methods concerning the handling of continuous attributes in ACO classification algorithms. Following the ideas of cAnt-Miner (Ant-Miner coping with continuous attributes), a new discretisation procedure based on the MDL principle was incorporated in the rule construction process, allowing the creation of discrete intervals using lower and upper bound values (i.e. $v_{lower} \leq attribute < v_{upper}$). Furthermore, it was proposed to deposit the pheromone on edges instead of vertices of the construction graph in order to deal with attribute interaction introduced by the way that the rule construction process copes with continuous attributes in cAnt-Miner.

We have performed experiments comparing the performance of Ant-Miner, cAnt-Miner, three variations of cAnt-Miner which incorporate the proposed methods, and J48 (Weka's C4.5 implementation). Using Ant-Miner as a baseline classifier, all cAnt-Miner variations (including the original cAnt-Miner) significantly outperformed Ant-Miner in at least three out of eight data sets. The incorporation of both proposed methods into cAnt-Miner, dubbed cAnt-Miner2-MDL, significantly outperformed Ant-Miner in five out of eight data sets. There were no significant differences between cAnt-Miner variations and J48. On the other hand, J48 significantly outperformed Ant-Miner only in two data sets.

As future research direction, it would be interesting to investigate the performance of different discretisation methods in the rule construction process. Also, since depositing pheromone on the edges leads to a modified pruning procedure, evaluating other kinds of pheromone updating strategies is also a direction worth further exploration.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support from an European Union's INTERREG project (Ref. No. 162/025/361). Fernando Otero also acknowledges further financial support from the Computing Laboratory, University of Kent.

REFERENCES

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

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

[3] F. Otero, A. Freitas, and C.G.Johnson, "cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes," in *Ant Colony Optimization and Swarm Intelligence (Proc. ANTS 2008), LNCS 5217*. Springer, 2008, pp. 48–59.

[4] R. Parpinelli, H. Lopes, and A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321–332, 2002.
[5] A. Freitas, R. Parpinelli, and H. Lopes, "Ant colony algorithms for data classification," To apper in *Encyclopedia of Info. Sci. & Tech*. 2nd Ed, 2008.
[6] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Thirteenth International Joint Conference on Artifical Inteligence*. Morgan Kaufmann, 1993, pp. 1022–1027.
[7] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651–665, 2007.
[8] A. Asuncion and D. Newman, "UCI Machine Learning Repository," http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.
[9] H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.
[10] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
[11] R. Kohavi and M. Sahami, "Error-based and entropy-based discretization of continuous features," in *Proceedings of the 2nd International Conference Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 114–119.