Clustering-based Bayesian Multi-net Classifier Construction with Ant Colony Optimization

Khalid M. Salama
School of Computing
University of Kent
Canterbury, CT2 7NF, UK
Email: kms39@kent.ac.uk

Alex A. Freitas
School of Computing
University of Kent
Canterbury, CT2 7NF, UK
Email: A.A.Freitas@kent.ac.uk

***Abstract*—Bayesian Multi-nets (BMNs) are a special kind of Bayesian network (BN) classifiers that consist of several local networks, typically, one for each predictable class, to model an asymmetric set of variable dependencies given each class value. Alternatively, multi-nets can be learnt upon arbitrary partitions of a dataset, in which each partition holds more consistent variable dependencies given the data subset in the partition. This paper proposes two contributions to the approach that clusters the dataset into separate data subsets to build asymmetric local BN classifiers, one for each subset. First, we extend the K-modes algorithm, previously used by the Case-Based Bayesian Network Classifiers (CBBN) approach to create clusters before learning the BN classifiers. Second, we introduce the Ant-Clust-B algorithm that employs Ant Colony Optimization (ACO) to learn clustering-based BMNs. Ant-Clust-B uses ACO in the clustering step before learning the local BN classifiers. Empirical results are obtained from experiments on 18 UCI datasets.**

I. Introduction

Ant Colony Optimization (ACO) [1] is a meta-heuristic for solving combinatorial optimization problems, inspired by observations of the behavior of ant colonies in nature. ACO has been successfully employed in several research areas related to our current work, classification [2], [3], [4], [5], clustering [6], [7], [8], and learning general-purpose Bayesian Networks (BNs) [9], [10], [11], [12]. Recently, the authors have introduced ABC-Miner [13], the first ACO-based algorithm to build Bayesian network classifiers, which has shown better performance compared to some greedy and deterministic BN algorithms. Thus, we carry on developing ACO-based algorithms in the Bayesian classification area. Classification is a central problem in data mining and machine learning where the system builds, from labelled data instances, a model (classifier) that predicts the class of unlabelled instances [14]. There are many types of classification methods [14], but in this work we focus on building BN classifiers.

A BN classifier is a special kind of probabilistic networks that aims to predict the class of a data instance by computing the posterior probability of each available class value, given the values of the predictor attributes of the instance, and then labeling the instance with the class having the highest posterior probability. Naïve-Bayes, as discussed in [15], is the simplest kind of BN classifier and it assumes the attributes are independent given the class label. Although it obtained good predictive performance in several domains [15], extensions were developed to improve Naïve-Bayes [16], [17], since the independency assumption amongst attributes is often not realistic.

In a BN classifier, a single network structure is built to model the variable dependencies in the whole dataset. By contrast, a Bayesian Multi-net (BMN) classifier consists of several local networks, one for each subset of the dataset, to model an asymmetric set of variable dependencies for each data subset. Typically, data subsets are obtained by partitioning the dataset based on the class values. Alternatively, arbitrary partitions can be discovered, in which each partition holds more consistent variable dependencies given the data subset in the partition. Consequently, more effective local BN classifiers are built for each data subset.

In this paper, we propose two new contributions to the approach that clusters the dataset into separate data subsets to build asymmetric local Bayesian network classifiers, one for each subset. First, we extend the K-modes clustering algorithm, previously used in Case-Based Bayesian Network Classifiers (CBBN) approach [18], to create data clusters before learning the BN classifiers. The modifications are applied on the clustering technique to cope well with the classification objective of the algorithm by using different instance similarity and cluster-belongingness measures. Second, we introduce the new Ant-Clust-B algorithm that employs the ACO meta-heuristics to learn cluster-based BMNs. Ant-Clust-B uses the cluster-then-learn two-step approach, where ACO is utilized in the clustering step, before a local Naïve-Bayes classifier is built on each data cluster in the next step.

The rest of the paper is structured as follows. The next section gives a brief overview on BN classifiers, followed by a background on ACO work in the related research areas in Section III. Section IV gives an overview of our new extensions and proposed approach. Section V describes the basic clustering algorithm used in CBBN. Our proposed extensions to the clustering technique for learning BMN classifiers are discussed in Section VI. The Ant-Clust-B algorithm is described in detail in Section VII. Experimental results on 18 benchmark UCI datasets, using different number of clusters, are shown in Section VIII, followed by the conclusions and future research directions in Section IX.

II. Bayesian Networks Overview

Bayesian networks (BN) is one of the most powerful types of method that model (in)dependence relationships between variables [19], representing those (in)dependencies in a graphical form. More precisely, a Directed Acyclic Graph (DAG) is used to represent the variables as nodes and statistical dependencies between the variables as edges between the nodes. In addition, a set of conditional probability tables (BN parameters), one for each variable, is obtained by computing the probability distribution of the variable given its parents. Inference is the main task of general-purpose BNs.

BN classifiers are a special kind of probabilistic networks built to answer queries about the probability of a specific node: the class attribute. Thus, the class node is treated as a special variable in the network; it is set as the parent of all other variables. The purpose is to compute the probability of each value c of the class variable C given $\mathbf{x}$ (an instance of the input attributes $\mathbf{X} = \{X_1, X_2, ..., X_n\}$), then label the instance with the class having the highest probability, as in the following formulas:

$$C(\mathbf{x}) = \arg\max_{\forall c \in C} P(C = c | \mathbf{x} = x_1, x_2, ..., x_n), \quad (1)$$

letting $\mathbf{Pa}(X_i)$ be the set of parent predictor variables of X_i in the network, according to the Bayes' Theorem:

$$\overbrace{P(c|x_1, x_2, ..., x_n)}^{\text{posterior probability}} \alpha \quad \overbrace{P(c)}^{\text{prior probability}} \quad \prod_{i=1}^{n} \overbrace{P(x_i|\mathbf{Pa}(X_i), c)}^{\text{likelihood}} \quad (2)$$

As mentioned earlier, Naïve-Bayes is the simplest type of BN classifier. Despite its very strong simplifying assumption – namely, that the attributes are independent from each other given the class label – Naïve-Bayes has outperformed several more sophisticated classification algorithms, especially in datasets where its independency assumption is true [15].

The literature describes several types of Bayesian network classifiers [16], [20], [17]: Tree Augmented Naïve-Bayes (TAN), Bayesian network Augmented Naïve-Bayes (BAN), and General Bayesian networks (GBN). TAN allows a node in a network to have one parent, in addition the class variable [17], which produces a BN with a tree-like structure. In a BAN, a node can have up-to k-parents in the network. GBN treats the class variable node as an ordinary node, which can have both parent and children nodes.

Unlike a BN classifier, where a single network structure is built to model the variable dependencies in the whole dataset, a Bayesian Multi-net (BMN) classifier consists of several local networks, one for each subset of the dataset, to model an asymmetric set of variable dependencies for each subset – to try to obtain a better dependency representation. This means that two variables X and Y might be directly dependent $(X \rightarrow Y)$ in one subset and independent in another subset, while $(X \leftarrow Y)$ might occur in a third one. Even if some relationships in different subsets are the same, the parameters that formalize the conditional dependencies might vary.

Typically, data subsets are obtained by partitioning the dataset according to the class values, where each Local BN corresponds to a value of class variable [20], [21]. In the class-based BMN, the dataset $\mathbf{D}$ is partitioned into $|C|$ subsets, where $|C|$ is the number of values in the domain of the class attribute, and each subset D_l would contain only the instances labeled by class value l. So, a *general* (not a classifier by itself) Bayesian network BN_l is built for each D_l subset.

Alternatively, arbitrary partitions can be discovered, where each partition holds more consistent variable dependencies given the data subset in the partition. Consequently, more effective local BN classifiers are built for each data subset. In which case, the dataset $\mathbf{D}$ is clustered into K data subsets, where K is an input value, and each cluster (subset) D_k may contain instances labeled by different class values. Hence, a local BN *classifier* BNC_k is built for each D_k with $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ and C variables.

A clustering-based BMN classifies an instance $\mathbf{x}$ by assigning the instance $\mathbf{x}$ to its nearest data cluster, D_k (the cluster that the instance would belong to), and then uses the local BN classifier, BNC_k, that is built on D_k, to compute the class value $C(\mathbf{x})$ that maximizes the posterior probability, as shown in Equation 3. The clustering-based BMN approach is our focus in this work.

$$C(\mathbf{x}) = \arg\max_{\forall c \in C} P(c|\mathbf{x}, BNC_k), \mathbf{x} \in D_k \quad (3)$$

A common approach for learning a BN (classifier) from a dataset $\mathbf{D}$ is to use a scoring function that evaluates several candidate structures with respect to $\mathbf{D}$, searching for the best network G according to this function [20]. Then, parameter learning can be done by simply computing a conditional probability table (CPT) for each variable with respect to its parent variables. The CPT of variable X_i encodes the likelihood of each value of this variable given each combination of values of $\mathbf{Pa}(X_i)$ in the network structure G, and the likelihood of the dataset $\mathbf{D}$ given a network G is denoted by $P(\mathbf{D}|G)$, which the algorithm tries to maximize.

Most algorithms used in the literature that utilize the Bayesian approach for classification focus on building a single model (BN classifier) on the whole dataset, and use deterministic and greedy search strategies. However, based on the characteristics of the Bayesian multi-nets and the rationale behind modeling them, stochastic meta-heuristic global search methods such as ACO, which are less prone to get stuck in local optima than greedy search methods, can be applied to build more effective Bayesian classification models.

III. Ant Colony Optimization Background

Ant Colony Optimization (ACO) is a meta-heuristic global search method that is inspired by the collective behavior of natural ants and is designed to achieve a certain goal [1]. ACO is a swarm intelligence paradigm that simulates real ant colonies using artificial ants. Each artificial ant creates a candidate solution to the problem at hand, then deposits pheromone on the part of the search space that the ant visited

while constructing its solution. The amount of pheromone deposited is proportional to the quality of the constructed solution, so pheromone provides positive feedback to guide the search to be done by other ants. The iterative process of building candidate solutions, evaluating their quality and updating pheromone values allows an ACO algorithm to converge to near-optimal solutions.

ACO has been employed for learning *general*-purpose BNs in several works [9], [10], [11], [12]. In the area of Bayesian classification, the authors have recently introduced ABC-Miner [13], at present the only algorithm that uses ACO for learning a BN *classifier* in the structure of a BAN, rather than a Bayesian Multi-net. Besides, ACO has contributed effectively in tackling the classification problem. Ant-Miner was the first ACO algorithm for classification rule discovery. A recent survey on Ant-Miner and its related work is presented in [22]. AnTree-Miner [23] and *c*ACDT [24] are ACO algorithms for building decision trees.

Since we focus on using ACO for learning clustering-based BMN, we briefly review the use of ACO for clustering. Cluster analysis consists of grouping a set of objects (data instances) in such a way that objects in the same group (cluster) are more similar (in some sense) to each other than to those in other groups [25]. Many ACO algorithms for clustering are briefly reviewed in [6]. Amongst these various ACO algorithms for clustering, we extended the work in [7] for our clustering-based BMN classification algorithms.

The ACO algorithm for clustering [7] assigns N objects (data instances) to K clusters. The ACO construction graph, which represents the problem's search space, contains $N \times K$ decision components; each instance with each possible cluster assignment. Each ant in the colony starts with an empty solution of length N elements, then the ant selects a cluster assignment for each element from the construction graph. The selection is performed probabilistically, based on the pheromone amount associated with each instance-cluster decision component. When every instance has been assigned to a cluster, the ant has a complete candidate clustering solution. For example, a candidate clustering solution for a dataset with 10 instances and 4 clusters is represented in the following form:

3	1	3	4	1	2	1	3	4	2

This representation means that in this candidate clustering solution, for example, the first instance belongs to cluster number 3, along with third and the eighth instances.

After each ant constructs a candidate solution, the quality of the solution is evaluated. Then the best l ants perform pheromone update to influence the solution construction process of the ants in the following iterations. The algorithm stops after a certain number of iterations or when the colony converges on a clustering solution.

The algorithm's goal is to minimize the sum of squared Euclidean distances between each data instance and the center of the cluster to which the instance belongs [7]. Therefore, a candidate clustering solution S is evaluated accordingly:

$$Quality(S) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathbf{Euclidean}(\mathbf{x}_{ki}, cen_k), \qquad (4)$$

where N_k is the number of instances in the k-th cluster, $\mathbf{x}_{ki}$ is the i-th instance in the k-th cluster, and cen_k is the centroid of the k-th cluster.

For a more detailed discussion on the ACO algorithm for clustering, the reader is referred to [7].

IV. Proposed Contributions Overview

In the clustering-based BMN learning, the aim is to build several local BN classifiers for a given dataset, one for each data subset, where in each subset the variable dependences are more consistent. In this context, we use the cluster-then-learn two-step approach. The approach is to perform a data clustering process in a separate step to partition the dataset into several subsets. After the completion of the clustering step, the local BN classifiers (such as Naïve-Bayes, TAN, BAN, or GBN), one for each data cluster, are constructed in the following step.

The Case-Based Bayesian Network Classifier (CBBN) algorithm [18] utilizes such a two-step approach. In CBBN, K-modes – a variation of the conventional K-means clustering algorithm adapted to nominal attributes – is used in the clustering phase, and then a BN classifier is built on each of the produced clusters. As a first step in our work, we extend the clustering technique used in the CBBN algorithm to better cope with the nominal attributes considering the final classification objective of the algorithm (which was not the objective of CBBN in [18]). In this extension we use different proximity measures (specific to the classification task) to evaluate the similarity between two data instances. Moreover, we use a different cluster-belongingness notion, which measures the degree of belongingness of an instance to a cluster, to better handle nominal attributes. The details are shown in Section VI.

Second, we introduce a novel algorithm, Ant-Clust-B, which also utilizes the cluster-then-learn approach. However, Ant-Clust-B employs an extended ACO algorithm for clustering (discussed in Section III), instead of the K-modes algorithm, in the clustering phase. Then, the algorithm builds the BMN by learning a Naïve-Bayes classifier on each data subset (cluster) produced by the ACO clustering step. The ACO algorithm, described in Section VII, is extended via utilizing the new similarity measure and the cluster-belongingness notion proposed in Section VI.

V. K-means Clustering then BMN Learning

The key ideas of this approach are using the well-known K-means algorithm to cluster the dataset into subsets, and then learning a set of local BN classifiers, one for each subset. Algorithm 1 shows the outline of K-means.

The K-means clustering algorithm starts by randomly selecting K data instances from the dataset to be the centroids of the K clusters, where K is the number of clusters specified

Algorithm 1 Pseudocode of K-means.

Begin
$K \leftarrow$ input;
Select K data instances as initial centroids;
repeat
  Assign each data instance to its nearest centroid;
  Recompute the centroid of each cluster;
until Centroids do not change
End

by the user. Then, each instance in the dataset is assigned to its nearest centroid. Formally, the instance $\mathbf{x}$ is assigned to cluster j according to the following formula:

$$Cluster(\mathbf{x}) = \underset{\forall\ k \in \{1,..,K\}}{\arg\min}\ \mathbf{Distance}(\mathbf{x}, cen_k), \qquad (5)$$

where cen_k is the centroid of the k-th cluster. The distance between the centroid of a cluster and a data instance (or generally, the distance between two instances) is computed using Euclidean distance, as follows:

$$\mathbf{Euclidean}(\mathbf{x_1}, \mathbf{x_2}) = \sum_{v=1}^{n} \sqrt{(\mathbf{x_1}_v - \mathbf{x_2}_v)^2}, \qquad (6)$$

where n is the number of attributes in the instance $\mathbf{x}$. The centroid of each cluster is then updated based on the instances assigned to its cluster, i.e., by taking the mean value of each attribute across all the instances in the cluster. The algorithm repeats the instance assignment and the centroid update steps until the centroids and the clusters converge to fixed values. This kind of distance measure and centroid calculation, for assigning instances to clusters, is appropriate when the instances in the dataset contain continuous (numeric-valued) attributes. However, when the dataset contains nominal (categorical) attributes, a different behaviour is expected. And since most of the work in the field of BN learning, as well as the current one, focus on data with nominal attributes, some aspects of the basic K-means need to be modified to cope with datasets with nominal attributes.

The CBBN algorithm [18] used K-modes, a modified version of the K-means algorithm. K-modes has the same overall structure of K-means, but how the distance between two instances is computed, and how the centroids of the clusters are updated are different. In K-modes, the difference between two nominal values of the same attribute in two data instances is either 1, if the two values are the same, or 0, if the two values are the different. The Euclidean distance (Equation 6) is computed between the two instances according to their attribute value differences. In addition, the centroid of a cluster is given by the mode value (the value that has the maximum number of occurrences) of each attribute, taken across all the instances in the cluster. After the K-modes algorithm finishes partitioning the dataset into several clusters, a BMN is constructed by learning a local BN classifier for each data cluster.

This clustering approach has a number of drawbacks, especially when the ultimate data mining task being solved is classification (like learning a BMN classifier), rather than clustering. These drawbacks have led us to extend the algorithm to better cope with nominal attributes in the context of clustering-for-classification. The next section discusses the drawbacks as well as our proposed extensions.

VI. Extensions to the Clustering Technique for Classification

The K-modes algorithm, used by CBBN, has two drawbacks in tackling nominal attributes. First, it considers that the distance between two values in the domain of an attribute is equals to 0 if these two values are different, and the same 0 value is assigned to any pair of different values. However, in some applications a pair of attribute values can be clearly considered closer to each other than other pairs of values, even if the individual values are all different. This case can be found in ordinal nominal attributes, where the attribute values have a sematic ordering, like {“high”,“medium”,“low”}. In this example, the distance between “high” and “medium” should be less than the distance between “high” and “low”.

Moreover, a similar but more subtle issue can be found even in some applications involving unordered nominal attributes, in the context of the classification task. For example, consider a dataset about customers who bought a specific product, where the aim is to build a classifier that can predict whether a new customer would be interested in buying this product or not. An attribute that may describe a customer (data instance) is [Profession], which may contain values like {“teacher”,“doctor”,“professor”,“engineer”}. In this case the values are unordered, and at first glance there is no principled way of saying that a pair of values is closer to each other than another pair. However, in the context of the classification task (rather than the clustering task), it is possible to assign different degrees of similarity (or distance) to two different pairs of such nominal values. The trick is to measure similarity not between (the names of) the attribute values themselves, but rather between the attribute values’ associations with the classes to be predicted.

In the previous example, it is possible that, say, professors and teachers have a higher degree of similarity than teachers and engineers, because professors and teachers could be associated mainly with the class “buy = yes”, whilst engineers could be associated mainly with the opposite class, “buy = no”. In a different dataset, however, teachers and engineers may be more similar than professors and teachers. Hence, the degree of similarity between a pair of attribute values should be computed dynamically by taking into account the class distribution associated with each of the attribute values being compared. Hence, the distance measure used by K-modes (which returns either 1 or 0) neglects a lot of information in the data, especially when the aim of the clustering phase is to produce data subsets that are more consistent for the classification task.

The second drawback of the K-modes algorithm is related to how an instance is assigned to a cluster. In K-modes, the distance between the instance (to be assigned to cluster) and the mode of each cluster is calculated, and the instance is assigned to the cluster of the nearest mode. However, the mode omits a lot of information about the attribute values in the cluster. For example, suppose we have a dataset with two attributes, and each can take two values $A_1 = \{v_{11}, v_{12}\}$ and $A_2 = \{v_{21}, v_{22}\}$. Suppose also that the dataset is clustered into two subsets, where each has the following attribute value distributions:

$Cluster_1$ [v_{11}=70%, v_{12}=30% — v_{21}=80% , v_{22}=20%]
$Cluster_2$ [v_{11}=90%, v_{12}=10% — v_{21}=60% , v_{22}=40%]

Note that the mode of both clusters is the same: $mode = (v_{11}, v_{21})$. In this case, an instance $\mathbf{x} = (v_{11}, v_{22})$ could be considered to belong equally to both clusters, since it has the same level of similarity to each of the clusters' modes. The distance between $\mathbf{x}$ and each cluster's mode is 1, according to Equation 6.

Nevertheless, and according to the attribute value distributions in the clusters, the instance $\mathbf{x}$ seems to belong to $Cluster_2$ more than $Cluster_1$, because the values of the instance $\mathbf{x}$ have a higher occurrence in $Cluster_2$ than $Cluster_1$. The K-modes algorithm neglects such facts, and would assign the instance to any of the two clusters randomly. Hence, we need a belongingness measure that would consider such value distributions to assign $\mathbf{x}$ to $Cluster_2$.

Due to the aforementioned drawbacks of K-modes in handling nominal attributes, in order to better achieve the classification objective of the clustering-then-BMN learning approach, we propose two extensions to the clustering technique. First, we use a class-based similarity function. This function calculates the similarity between two instances according to their attribute values with respect to the classes; two different values of the same attribute are considered to be similar if they share a similar relationship to predicting the classes. The similarity function is shown in the following formula [26]:

$$\mathbf{Similarity}(\mathbf{x_1}, \mathbf{x_2}) = -\sum_{v=1}^{n}\sum_{l=1}^{|C|} |P(\mathbf{x_1}_v|C_l) - P(\mathbf{x_2}_v|C_l)|, \quad (7)$$

where n is the number of attributes and $P(\mathbf{x}_v|C_l)$ is conditional probability of the attribute value $\mathbf{x}_v$ given the class value C_l. This conditional probability is the ratio of the number of instances with attribute value $\mathbf{x}_v$ and the class value C_l over the number of instances with class value C_l in the dataset.

According to the previous formula, if two attribute values in two different instances have similar values of that conditional probability with respect to a class value, the difference term in the equation will be small; and if a small difference is observed in general across all (or the vast majority of) attributes and class values, the two instances are considered similar to each other. And vice versa: if two attribute values in two different instances have very different values of that conditional probability with respect to a class value, the difference term in the equation will be large; and if a large difference is observed in general across all (or the vast majority of) attributes and class values, the two instances are considered dissimilar to each other. Hence, the class-based similarity function calculates the similarity between two instances based on how similar they are in predicting the various class values, which should lead to a better clustering-for-classification process than the conventional Euclidean distance used by K-modes (which is purely for clustering, ignoring the classification goal).

The second extension to the clustering technique is related to how an instance is assigned to a cluster in the testing phase, or how it changes its cluster-belongingness during the clustering phase. The idea is that, instead of using the $mode$ of a cluster to measure how near/far an instance is from the cluster according to the distance/similarity between the $mode$ and the instance, the degree of belongingness of an instance to a cluster is the average similarity between that instance and all other instances in the cluster, and then the instance is assigned to its most similar cluster, as formalized in the following equation:

$$Cluster(\mathbf{x}) = \underset{\forall\ k\in\{1,..,K\}}{\arg\max} \frac{\sum_{i=1}^{N_k} \mathbf{Similarity}(\mathbf{x}, \mathbf{x}_i)}{N_k}, \quad (8)$$

where K is the number of clusters and N_k is the number of the instance in k-th cluster. The rationale behind this technique is to avoid depending on the mode of the cluster, which (as discussed earlier) omits a lot of information about the cluster's data distribution, and to consider every instance in the cluster when calculating the degree of belongingness of the instance to the cluster. Hence, in each iteration of the clustering algorithm, the degree of belongingness is calculated for each instance in the dataset and each cluster, and then the instance is assigned to the cluster to which it belongs the most. The algorithm stops when no instance changes its cluster.

Note that the CBBN algorithm performs a post-processing step after clustering, which consists of finding a vector of attribute values for each cluster, called index, which discriminates one cluster from others by a unique value assignments to its most relevant and descriptive attributes [18]. This is to improve the instance-cluster assignment process in the classification phase. However, our proposed cluster-belongingness notion should not need this post-processing step, since it already directly improves the instance-cluster assignment process in the clustering phase, to create better data clusters, as well as in the classification phase, especially when the class-based similarity function is used.

VII. ACO Clustering then BMN Learning

We propose Ant-Clust-B, the ant-based clustering algorithm for learning BMN classifiers. The algorithm applies the clustering-then-learning approach, in which ACO is utilized in the clustering step to produce data subsets. Then a set of local BN classifiers, one for each data subset, is constructed. The reason for applying the ACO meta-heuristic, instead of the K-means algorithm, in the clustering step, is as follows. The K-means algorithm is very sensitive to the values of the initial

centroids (or modes in the case of K-modes) randomly chosen to start the algorithm. And since K-means can be viewed as a greedy search technique for an optimization problem, which is to minimize the sum of the Euclidean distances between the instances and the centroids of the clusters (Equation 4), a bad initialization of the clusters' centroids may lead the algorithm to get trapped into a bad local optima, which in turn affects the predictive quality of the BN classifiers built on the resultant data clusters.

On the other hand, ACO is a stochastic meta-heuristic global search method. ACO's global search is due to the use of a population of artificial ants that cooperatively search for the best solution in parallel, exploring different regions of the search space at each iteration of the algorithm, and the use of pheromone to influence the probability of the ants in the following iterations to visit the "good" regions of the search space when constructing their solution. As a result of this global search, ACO is less likely to get trapped into local optima in the search space, which improves the chances of finding better clustering solutions. Algorithm 2 shows the overall process of Ant-Clust-B.

Algorithm 2 Pseudo-code of Ant-Clust-B.

Begin
$K = input$;
$BMN = \phi$;
$ClustSolution_{gbest} = \phi$;
$Q_{gbest} = 0$;
$InitializePheromoneAmounts()$;
$t = 1$;
repeat
  $ClustSolution_{tbest} = \phi$;
  $Q_{tbest} = 0$;
  for $i = 1 \rightarrow$ `colony_size` **do**
    $ClustSolution_i = CreateSolution(ant_i)$;
    $Q_i = ComputeQuality(ClustSolution_i)$;
    if $Q_i > Q_{tbest}$ **then**
      $ClustSolution_{tbest} = ClustSolution_i$;
      $Q_{tbest} = Q_i$;
    end if
  end for
  $PerformLocalSearch(ClustSolution_{tbest})$;
  $UpdatePheromone()$;
  if $Q_{tbest} > Q_{gbest}$ **then**
    $ClustSolution_{gbest} = ClustSolution_{tbest}$;
    $Q_{gbest} = Q_{tbest}$;
  end if
  $t = t + 1$;
until $t =$ `max_iterations` **or** $Convergence()$;
for $k = 1 \rightarrow K$ **do**
  $BNC_k = LearnBNClassifier(ClustSolution(k))$;
  append BNC_k **to** BMN;
end for
return BMN;
End

The outline of the algorithm is as follows. In essence, each ant_i in the colony creates a candidate clustering solution $ClustSolution_i$, i. e. a full instance-cluster assignment, with K clusters. Then the quality of the constructed solution is evaluated. The best solution $ClustSolution_{tbest}$ produced in the colony at the current iteration t is selected to undergo local search before the ant updates the pheromone trail according to the quality of its solution Q_{tbest}. Next, the algorithm compares the current iteration's best solution $ClustSolution_{tbest}$ with the global best solution $ClustSolution_{gbest}$ to keep track of the best solution found along the entire search so far. This set of steps is considered an iteration of the $repeat - until$ loop and is repeated until the same solution is generated for a number of consecutive trials, specified by the `conv_iterations` parameter (indicating convergence) or until `max_iterations` is reached. The values of `max_iterations`, `conv_iterations` and `colony_size` are user-specified parameters.

An ant constructs a clustering solution in the same manner as in [7], as discussed in Section III. It starts with an empty solution of length N elements, where each element represents an instance in the dataset. Then, the ant selects a cluster assignment for each element in the clustering solution. The selection is performed probabilistically according to the pheromone amount associated with the decision components in the construction graph, where each decision component represents a possible instance-cluster assignment.

The quality of a candidate clustering solution is evaluated according to a cohesiveness measure that uses the extensions introduced in Section VI, as shown in the following formula:

$$Q(ClustSolution) = \sum_{k=1}^{K} \frac{\sum_{i=1}^{N_k} \sum_{j=i}^{N_k} \mathbf{Similarity}((\mathbf{x}_{ki}, \mathbf{x}_{kj}))}{N_k}, \quad (9)$$

where K is the number of the clusters, and N_k is the number of the instances in the k-th cluster. The proposed clustering quality evaluation function measures the degree of cohesiveness in each cluster by taking the average of the class-based similarity (Equation 7) between each instance and all the other instances in its cluster, and sums up the averages over all the clusters to get the quality of the constructed candidate clustering solution.

At each iteration, the best clustering solution constructed amongst the ants in the colony undergoes local search. We propose running one iteration of the K-means algorithm on the $ClustSolution_{tbest}$, that is, assigning each instance to its nearest cluster, with respect to the class-based similarity measure and the cluster-belongingness notion.

We use a new strategy for pheromone update in Ant-Clust-B. The pheromone amount is deposited according to the quality of two constructed solutions: the iteration best Q_{tbest} and the global best Q_{gbest}, in a weighting strategy, as follows:

$$\tau_{ik}(t+1) = \tau_{ik}(t) + \phi_1.Q_{tbest}(t) + \phi_2.Q_{gbest}(t) \quad (10)$$

where τ_{ik} is the amount of pheromone associated with the instance-cluster $(i - k)$ assignment decision compontnet. ϕ_1 and ϕ_2 represents the *intensity* of the pheromone to be deposited in iteration t according to the quality of the iteration best and global best solutions respectively, as:

$$\phi_1 = \frac{max\_iterations - t}{max\_iterations}, \quad \phi_2 = \frac{t}{max\_iterations} \quad (11)$$

Hence, in the early iterations, more weight is given to the local best rather than the global best (as $max\_iterations - t$ is greater than t). This is applied in order to introduce search diversity. However, as the iterations go on, the quality of the global best increases, which gains more weight (as t increases and $max\_iterations - t$ decreases) in directing the search, leading to convergence. Note that $\phi_1 + \phi_2$ always equals 1 at any given iteration t.

Pheromone normalization is then applied to all of the decision components to simulate evaporation as in [13].

VIII. Experiments and Results

The performance of our extended clustering for classification algorithm, denoted as K-Clusts-B, as well as the Ant-Clust-B algorithm were evaluated using 18 public-domain datasets from the University of California at Irvine (UCI) dataset repository [27]. Datasets containing continuous attributes were discretized in a pre-possessing step, applying the C4.5-Disc [14] algorithm to the training-set folds. The main characteristics of the datasets are found using the URL in [27]. We used Naïve-Bayes to build the local BN classifiers for each data subset produced in the clustering step.

We compare the predictive accuracy of the extended algorithm and the proposed ant-based algorithm with conventional Naïve-Bayes (built on the whole dataset without clustering) and with conventional K-modes (described in Section VI to cluster the data for building the local Naïve-Bayes, denoted as K-Modes-B).

We performed 3 experiments for each data set with 3 different numbers of clusters: 2, 4, and 6. The experiments were carried out using the *stratified* 10-fold cross validation procedure. In essence, a dataset is divided into 10 mutually exclusive partitions (folds), with approximately the same number of instances in each partition. Then each classification algorithm is run 10 times, where each time a different partition is used as the test set and the other 9 partitions are used as the training set. The results (accuracy rate on the test set) are averaged and reported as the accuracy rate of the classifier. We run each algorithm (Ant-Clust-B, K-Modes-B, and K-Clust-B) 10 times – using a different random seed each time (to initialize the search for the ant-based algorithm, and to select the initial centroids for the K-means-based algorithms) – for each of the 10 iterations of the cross-validation procedure (i.e. 100 runs in total, for each dataset). The parameter configuration used in our experiments is shown in Table 1.

TABLE I
Parameter Settings Used in the Experiments

Parameter	Value
`max_iterations`	1000
`colony_size`	10
`conv_ iterations`	10
`K` (number of clusters)	2, 4, 6

Table 2 reports the predictive accuracy (%) results for the 3 used clustering-for-classification algorithm with 3 different numbers of clusters. The results for Naïve-Bayes are also reported as a baseline. A value in bold face is the highest accuracy value for the corresponding dataset among all accuracy values obtained using the same number of clusters in different algorithms. An underlined value is the best accuracy value for the dataset.

As shown in Table 2, using 2 clusters, the extended K-Clust-B algorithm outperforms the K-Modes-B algorithm in 15 out of 18 datasets, while Ant-Clust-B obtains the best results across all the algorithms in 16 datasets. Using 2 clusters obtains the best results compared to other numbers of clusters only in one dataset.

With 4 clusters, the K-Clust-B algorithm outperforms the conventional K-Modes-B algorithm in 16 datasets, while the ant-based algorithm for learning clustering-based BMN classifiers obtains the best results across all algorithms in 16 out of 18 datasets. Using 4 clusters obtains the best results compared to other numbers of clusters in 4 datasets.

With 6 clusters, K-Cluster-B outperforms K-Modes-B in building BMN classifiers in 15 datasets, while our proposed Ant-Clust-B algorithm obtains the best results amongst all algorithms in 13 out of 18 datasets. Using 6 clusters obtains the best results compared to other numbers of clusters in 13 datasets.

According to the results, our proposed extensions regarding the class-based similarity measure and the cluster-belongingness notion, used in both K-Clust-B and Ant-Clust-B, have improved the performance of the algorithms in clustering the dataset for building effective BMN classifiers. On the other hand, the use of the ACO meta-heuristic for finding data clusters has improved the performance of the algorithm and produced the majority of the best results.

IX. Concluding Remarks

The paper has addressed the clustering approach for building a set of BN classifiers, one for each data cluster. We proposed two extensions to the K-modes clustering algorithm to better cope with nominal attributes in the context of clustering for classification: the use of a class-based similarity measure and a cluster-belongingness notion for instance-cluster assignment. Moreover, we proposed an ACO-based algorithm for learning clustering-based BMN classifiers, where ACO is used in the clustering phase. Empirical results showed that the proposed extensions as well as the use of the ACO-meta heuristic have produced better predictive accuracy results.

In the future, we would like to extend the ACO algorithm by performing the clustering phase and the BMN-construction phase in a synergistic way, where an ant produces a clustering solution and builds a BMN in a single iteration before the solution is evaluated for pheromone update. In addition, we would like to use a different representation for the clustering solution constructed by an ant, in which a solution might contain only the modes of the clusters.

TABLE II
PREDICTIVE ACCURACY (%) RESULTS

Dataset	Naïve-B	K-Modes-B			K-Clust-B			Ant-Clust-B		
		k=2	k=4	k=6	k=2	k=4	k=6	k=2	k=4	k=6
balance scale	76.1	76.5	76.8	76.9	76.5	76.9	**77.2**	**76.8**	**77.4**	**77.2**
car evaluation	86.7	86.9	88.7	92.7	87.3	92.7	92.9	**88.8**	**92.9**	**93.3**
chess (rook vs. pawn)	86.5	87.6	89.4	91.2	**88.6**	90.1	92.4	86.8	**91.9**	**93.2**
contraceptive method choice	50.8	51.7	53.6	53.8	53.4	55.7	**57.8**	**53.9**	**56.1**	57.6
statlog credit (australian)	79.3	80.8	81.7	82.8	81.5	**82.7**	83.2	**81.7**	**82.7**	**83.8**
dermatology	96.1	97.2	97.6	96.5	97.6	98.0	**97.9**	**97.8**	**98.4**	97.6
glass	61.6	64.2	64.8	65.2	**65.7**	66.1	66.8	65.5	**66.3**	**67.8**
hayes-roth	84.0	84.1	85.0	84.7	**84.7**	84.1	**85.0**	**84.7**	**85.6**	84.8
heart disease (cleveland)	54.6	61.5	66.4	69.8	64.1	69.8	71.9	**65.9**	**71.0**	**73.6**
ionosphere	91.1	92.8	91.8	90.8	**93.8**	**92.6**	90.4	**93.9**	91.8	**92.9**
lung cancer	84.6	**91.6**	92.5	94.6	**91.6**	**95.0**	95.7	**91.6**	94.8	**95.8**
monks	60.5	58.6	60.6	61.7	58.6	60.9	61.7	**59.5**	**61.7**	**63.2**
nursey	90.1	91.3	92.4	93.0	92.1	93.0	93.2	**92.6**	**94.1**	**94.4**
parkinsons	93.9	94.2	95.1	95.4	95.0	96.1	96.7	**95.1**	**96.5**	**97.0**
post-operative patient	69.5	68.0	72.1	73.1	68.2	72.1	70.8	**71.9**	**75.2**	**79.9**
segmentation	93.7	94.2	94.3	94.6	94.5	95.6	**95.8**	**94.6**	**95.9**	95.7
soybean	47.6	42.7	50.6	52.8	43.6	52.5	**61.1**	**43.8**	**52.6**	59.6
tic-tac-to	68.7	71.3	74.2	79.8	75.2	79.1	87.88	**76.3**	**79.9**	**88.9**

REFERENCES

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

[2] 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, pp. 651–665, 2007.

[3] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 321–332, 2002.

[4] K. M. Salama, A. Abdelbar, and A. A. Freitas, "Multiple Pheromone Types and Other Extensions to the Ant-Miner Classification Rule Discovery Algorithm," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 149–182, 2011.

[5] K. M. Salama, A. M. Abdelbar, F. E. Otero, and A. A. Freitas, "Utilizing Multiple Pheromones in an Ant-based Algorithm for Continuous-Attribute Classification Rule Discovery." *Applied Soft Computing*, vol. 13, no. 1, 2012.

[6] M. Jafar and R.Sivakumar, "Ant-based Clustering Algorithms: A Brief Survey," *International Journal of Computer Theory and Engineering*, vol. 2, pp. 787–796, 2010.

[7] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An ant colony approach for clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187–195, 2004.

[8] X. yong Liu and H. Fu, "An Effective Clustering Algorithm With Ant Colony," *Journal of Computers*, vol. 5, pp. 598–605, 2010.

[9] Y. Wu, J. McCall, and D. Corne, "Two novel Ant Colony Optimization approaches for Bayesian network structure learning," *International Conference on Evolutionary Computation (CEC)*, pp. 1–7, 2010.

[10] L. M. de Campos, J. M. Fernandez-Luna, J. A. Gamez, and J. M. Puerta, "Ant colony optimization for learning Bayesian networks," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 291–311, 2002.

[11] R. Daly and Q. Shen, "Learning Bayesian Network Equivalence Classes with Ant Colony Optimization," *Artificial Intelligence Research*, vol. 35, pp. 391–447, 2009.

[12] P. C. Pinto, A. Nägele, M. Dejori, T. A. Runkler, and Ao, "Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 767–779, 2009.

[13] K. M. Salama and A. A. Freitas, "ABC-Miner: an Ant-based Bayesian Classification Algorithm," *International Conference on Swarm Intelligence (ANTS)*, pp. 2677–2694, 2012.

[14] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2010.

[15] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, 2nd ed. John Wiley & Sons Inc, 1995.

[16] J. Cheng and R. Greiner, "Comparing bayesian network classifiers," *15th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 101–108, 1999.

[17] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, "Bayesian Network Classifiers," *Machine Learning*, pp. 131–163, 1997.

[18] E. S. Jr. and A. Hussein, "Case-Based Bayesian Network Classifiers," *17th International FLAIRS Conference, AAAI*, vol. 5, pp. 598–605, 2004.

[19] R. Daly, Q. Shen, and S. Aitken, "Learning bayesian networks: Approaches and issues," *Knowledge Engineering Reviews*, vol. 26, no. 2, pp. 99–157, 2011.

[20] J. Cheng and R. Greiner, "Learning bayesian belief network classifiers: Algorithms and system," *14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pp. 141–151, 2001.

[21] D. Geiger and D. Heckerman, "Knowledge representation and inference in similarity networks and Bayesian multinets," *Artificial Intelligence*, vol. 82, no. 1-2, pp. 45–74, 1996.

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

[23] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Inducing Decision Trees with an Ant Colony Optimization Algorithm," *Applied Soft Computing*, vol. 12, no. 11, pp. 3615–3626, 2012.

[24] U. Boryczka and J. Kozak, "An Adaptive Discretization in the ACDT Algorithm for Continuous Attributes," in *3rd International Conference on Computational Collective Intelligence: Technologies and Applications (ICCCI'11)*. Springer-Verlag, 2011, pp. 475–484.

[25] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2000.

[26] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, pp. 1213–1228, 1986.

[27] UCI Repository of Machine Learning Databases. Retrieved Oct 2011 from, URL:www.ics.uci.edu/ mlearn/MLRepository.html.