Classification with Cluster-based Bayesian Multi-Nets using Ant Colony Optimization

Khalid M. Salama, Alex A. Freitas

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

Abstract

Bayesian Multi-net (BMN) classifiers consist of several local models, one for each data subset, to model asymmetric, more consistent dependency relationships among variables in each subset. This paper extends an earlier work of ours and proposes several contributions to the field of clustering-based BMN classifiers, using Ant Colony Optimization (ACO). First, we introduce a new medoid-based method for ACO-based clustering in the Ant-ClustB$_{MB}$ algorithm to learn BMNs. Both this algorithm and our previously introduced Ant-ClustB$_{IB}$ for instance-based clustering have their effectiveness empirically compared in the context of the "cluster-then-learn" approach, in which the ACO clustering step completes before learning the local BN classifiers. Second, we propose a novel "cluster-with-learn" approach, in which the ACO meta-heuristic performs the clustering and the BMN learning in a synergistic fashion. Third, we adopt the latter approach in two new ACO algorithms: ACO-ClustB$_{IB}$, using the instance-based method, and ACO-ClustB$_{MB}$, using the medoid-based method. Empirical results are obtained on 30 UCI datasets.

Keywords: Data mining, Ant Colony Optimization, Bayesian Network Classifiers, Cluster-based Bayesian Multi-nets

1. Introduction

Bayesian networks (BNs) excel in inference via modelling (in)dependency relationships between variables [1, 2]. BNs can also be used for the data mining classification problem [3, 4]. In this context, a BN classifier is a specific category of probabilistic networks that assigns, to a new data instance, the class which has the largest posterior probability, given the values of the predictor attributes of that instance. Several types of BN classifiers were introduced in the literature; Naïve-Bayes, Tree Augmented Naïve-Bayes (TANs), Bayesian networks Augmented Naïve-Bayes (BANs) and General Bayesian Networks (GBNs) [5, 6].

Unlike BN classifiers, where a single network (probably a complex structure with a large number of edges) is built to model the variable dependencies from 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 data subset. Typically, data subsets are obtained by partitioning the dataset based on the class values [6, 7]. Alternatively, data partitions can be automatically discovered by a learning algorithm, where each partition holds more consistent variable dependencies given the data subset in the partition. Consequently, more effective local BN classifiers, with less structural complexity, are built for each data subset.

1.1. Motivation

Ant Colony Optimization (ACO) [8] is a meta-heuristic inspired by the behaviour of natural ant colonies. ACO has been successfully employed in several machine learning research areas, namely classification [9, 10, 11, 12], clustering [13, 14, 15], and learning general-purpose BNs [16, 17, 18, 19]. Recently, the authors have introduced ABC-Miner [20, 21, 22], the first ACO-based algorithm to build BN classifiers, which has been shown to obtain better predictive performance, overall, when compared to some greedy BN learning algorithms. Hence, we carry on developing ACO-based BN classifiers.

The motivation behind this work is as follows. Most algorithms used in the literature based on BN approach for classification focus on building a single model (BN classifier) on the whole dataset. Only a few works have proposed to build several local Bayesian models via clustering the dataset, and in general those few works use greedy search methods. Hence, based on the BMN advantage of capturing asymmetric, more consistent variable dependency relationships from a well-partitioned dataset, it seems sensible to explore the use of a non-greedy, robust global search method to find a good partitoning of the data for the BMN approach. We pursue this direction by applying ACO, a meta-heuristic global search method which is less prone to get stuck in local optima than greedy search methods, to learn more effective (in terms of predictive accuracy) and less complex (in terms of the number of edges) BN classifiers. This work is a major extension of our recent work [23], which was the first work to utilize the ACO meta-heuristic for building clustering-based BMN classifiers.

1.2. A Brief Note on Our Earlier Work Being Extended in This Paper

The research described in this paper builds on our recent work on an ACO algorithm for clustering-based Bayesian Multi-Net learning [23]. That work introduced two basic ideas, namely:

- First, we modified the instance similarity and cluster-membership measures of the K-modes clustering algorithm, which was previously used to create data clusters before learning the BN classifiers in [24]. The modification consisted of proposing different instance similarity and cluster-membership measures that cope better with nominal attributes in the context of classification, which is the objective of the current work.

- Second, we introduced the Ant-ClustB$_{IB}$ algorithm that employs ACO to learn clustering-based BMNs, using an instance-based method for ACO

clustering solution representation. The Ant-ClustB$_{IB}$ employed the conventional‘ ‘clustering-then-learn” approach, where the data clusters are completely generated before starting the BMN learning phase. Given the good results obtained by Ant-ClustB$_{IB}$ we extend that algorithm in order to try to improve its performance, as discussed in the next subsection.

1.3. Contributions of This Work

This paper extends our aforementioned previous work, and gives the following contributions to the area of clustering-based BMN learning:

- First, we propose a new type of ACO clustering solution representation, the medoid-based representation, and utilize it in the new Ant-ClustB$_{MB}$ algorithm to learn BMNs. Note that both the instance-based Ant-ClustB$_{IB}$ and the medoid-based Ant-ClustB$_{MB}$ algorithms employ the sequential two-phase “cluster-then-learn” approach, in which the ACO meta-heuristics is used in a separate clustering phase before learning BMN classifiers.

- Second, we propose a novel “cluster-with-learn” approach, in which the ACO meta-heuristic performs both the clustering and the BMN learning in a synergistic fashion, in one integrated phase.

- Third, we incorporate the cluster-with-learn approach into two ACO-based clustering methods: the ACO-ClustB$_{IB}$and the ACO-ClustB$_{MB}$algorithms proposed in this paper.

- Fourth, we used two local BN classifiers: Naïve-Bayes and TAN, instead of just Naïve-Bayes as in [23]. We used these two algorithms since they tend to work well in small datasets, which is the case in our cluster-based approach after partitioning the whole (big) dataset into smaller data subsets. Besides, using more complex local BN learning algorithm would dramatically increase the computational time of the whole algorithm.

Moreover, in this work, we extend our experimental evaluations as follows. First, we run four sets of experiments, each used a different number of clusters (a parameter defining the number of local BN classifiers): 2 clusters in the first set, 4 clusters in the second set, 6 clusters in the third set, and 8 clusters in the fourht set. In our previous work [23], we only used three number of clusters: 2, 4, and 6. Second, we increased the number of the datasets from 18 to 30.

1.4. Paper Organization

The rest of the paper is structured as follows. The next section gives some background on various related topics: the different roles of classification and clustering in our work; an overview on BNs, BN classifiers and BMNs; and an overview on Ant Colony Optimization. We describe the clustering algorithm used for building BMNs in Section 3. Section 4 describes the ACO algorithm for clustering that uses the instance-based solution representation. Our proposed

extensions to the clustering technique for learning BMN classifiers are discussed in Section 5. We introduce the new medoid-based solution representation for the ACO-based clustering in Section 6. Section 7 describes the clustering then BMN learning approach, realized in the Ant-ClustB$_{IB}$ and the Ant-ClustB$_{MB}$ algorithms. Section 8 describes the clustering with BMN learning approach, realized in the ACO-ClustB$_{IB}$ and the ACO-ClustB$_{MB}$ algorithms. Experimental methodology and result analyses are discussed in Sections 9 and 10, respectively. We conclude with general remarks in Section 11.

2. Background

2.1. Classification vs. Clustering

Since this work involves both classification and clustering, it is worth emphasizing the different roles of these tasks in this research. In data mining [3, 25], classification is a supervised learning task that aims to build, from labelled instances, a model (classifier) used to predict the class of unlabelled instances. Clustering is an unsupervised learning task, where the target is to group a set of objects (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. Figure 1 shows a graphical illustration between the two tasks.

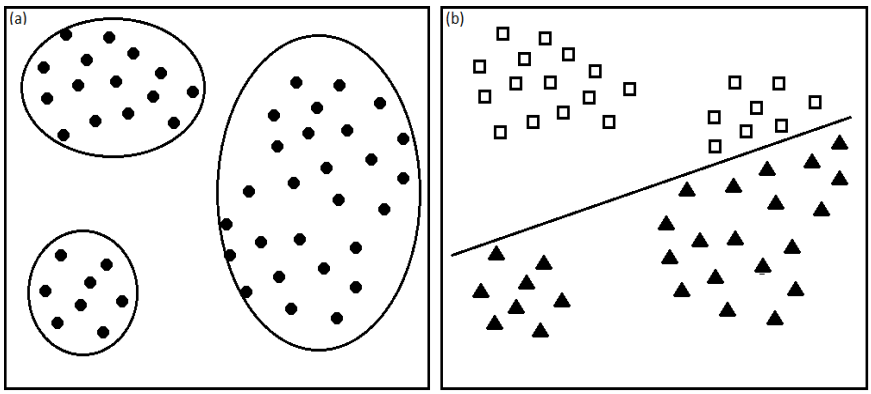

Figure 1: (a) Clustering: the data instances are grouped into three different clusters according to their location in the data space. (b) Classification: the labeled data instances are separated according to the classes they belong to. Note that instances belonging to the same cluster (because they are near to each other), can belong to different classes, as shown in the right-hand side of the figure.

This work is ultimately addressing a classification problem, the problem of learning BMN classifiers. As such, the performance of the proposed methods is evaluated mainly in terms of predictive accuracy, as usual. In order to solve that classification problem, however, we use clustering-based methods. That is, we use clustering methods as a means to learn more accurate BMN classifiers, rather than just generating well-separated goups of data instances. More precisely, we

use clustering methods to partition the data into a set of coherent clusters, where each cluster is supposed to contain relatively similar instances – at least the degree of similarity among instances within a cluster should be larger than the degree of similarity among instances in different clusters. Then a local BN classifier is learned from each cluster.

Note that, since in this work we use clustering to support the main objective of building more accurate classification models, the quality of a clustering solution is assessed by the predictive accuracy of the local BN classifiers constructed from the corresponding clusters. That is, we are not interested in evaluating the clustering results per se, we are only interested in the clustering results as a means to improve the classification results.

2.2. Bayesian Network Classifiers

A Bayesian Network (BN) is a type of probabilistic graphical model where nodes represent variables (features or attributes) and directed edges represent dependencies between variables. Each variable X_i is associated with a conditional probability table (CPT), which encodes the probability of each value of that variable given each combination of values of $\mathbf{Parents}(X_i)$ in the graph G [26, 2, 27].

A BN classifier is a specific type of probabilistic graphic model where the class variable is given a special treatement and the model is built with the specific purpose of representing dependencies that are relevant for predicting the class label of an instance, given the values of its predictor variables. That is, a BN classifier computes the probability of each value c of the class variable C given an instance $\mathbf{x}$ (with the attribute set $\mathbf{X}$), and then labels the instance with the class having the highest probability, as specified in the following formulas:

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

and according to the Bayes' Theorem and probability factorization:

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

where $C \in \mathbf{Parents}(X_i)\ \forall\ X_i \in \mathbf{X}$.

There are many different types of BN classifiers, varying mainly in the number and types of probability values (entries in the CPTs) that they need to compute [5, 7, 6, 28]. For the purposes of this paper, the main distinction that needs to be made is the difference between k-dependency BN classifiers and Bayesian Multi-Nets (the focus of this paper, discussed in the next Subsection).

Here we are using the term k-dependency BN classifier in a broad sense to refer to a type of BN classifier where the class node is the "root" of the network, there are edges pointing from the class node to each of the other variables in the network, and each predictor variable (attribute) has at most k parents in the network – not counting the class, which is the parent of all

nodes. k-dependency BN classifiers can be further categorized according to the value of the maximum number of parents for each variable (i.e. the value of k). The case where $k = 0$, where each predictor attribute has only the class node as its parent, is the well-known Naïve-Bayes classifier, which assumes that attributes are independent from each other given the class. The case where $k = 1$ is generally referred to as a Tree-Augmented Naïve-Bayes (TAN), since the dependencies among attributes are represented by a tree (each attribute has at most one parent attribute). More generally, when the value of k is 2 or more, the BN classifier is usually called a Bayesian network Augmented Naïve-Bayes (BAN).

Broadly speaking, the larger the value of k, the stronger the graphical model's ability in representing more dependencies among the variables, but the larger the number of parameters (entries in the CPTs) to be estimated – which leads to longer processing times and increased risk of overfitting. It should be noted that, in a k-dependency BN classifier, there is a single network structure modelling dependencies among variables in the entire dataset. This is not the case with Bayesian Multi-net classifiers, discussed next.

2.3. Bayesian Multi-nets

Unlike the aforementioned k-dependency BN classifiers, a Bayesian Multi-net (BMN) classifier consists of a set of local Bayesian networks (BNs), more precisely one local BN for each subset of instances in the dataset. The basic idea is that the actual dependencies among variables might be significantly different across different subsets of instances, and in this case it makes sense to build a different local BN, capturing a different set of dependencies among attributes, for each of those local subsets of instances. For instance, for a given pair of variables X_i and X_j, the actual dependency between them might be best represented as $(X_i \rightarrow X_j)$ in one instance subset and as $(X_j \rightarrow X_i)$ in another instance subset, and there might be no dependency between them in another yet instance subset. Even if the same kind of dependency (with the same direction) is represented in two different instance subsets, it is possible that the precise values of the CPTs associated with that dependency would be different in the two instance subsets. Hence, the idea of learning a variable-dependency BN (and associated CPTs) specific for each subset of instances, instead of a "one-size-fits-all" BN (like in k-dependency BN classifiers), is intuitively appealing, and could improve the predictive accuracy.

Typically, instance subsets (hereafter referred to as data subsets) are trivially obtained by partitioning the dataset according to the class values. In this case, each local BN is built from the instances belonging to a distinct value of the class variable [7, 29]. In more precise notation, in a class-based BMN the dataset $\mathbf{D}$ is partitioned into $|C|$ data subsets, where $|C|$ is the cardinality of the domain of the class variable (i.e. the number of class values), and each data subset D_c contains only the instances labeled by the class value c. Note that the class attribute is not represented anymore in each data subset, and therefore a *general* Bayesian network BN_l (which is not a classifier by itself) is built for each data subset D_c. Thus, a Bayesian multi-net classifier is a set of

local BNs $\{BN_1, BN_2, ..., BN_{|C|}\}$ that, together with the prior probability of C, classifies an instance x by choosing the class $C(x)$ that maximizes the posterior probability, as shown in the following equation:

$$C(x) = \underset{\forall\ c \in C}{\arg\max}\, P(\mathbf{x} = x_1, x_2, ..., x_n | BN_c) \times P(C = c), \tag{3}$$

It is important to note, though, that partitioning the data according to the values of the class variable is not necessarily the best approach to maximize the predictive accuracy of a BMN classifier. One can use a type of machine learning algorithm to learn the best partition (the one maximizing the predictive accuracy of the BMN classifier). The basic idea is to search for partitions where the variable dependency relationships are more consistent within each data subset, so that a more effective local BN is built for each data subset. For example, if we have a world-wide demographic dataset, we might not find a "statistically" strong relationship between two variables like [Gender] and [Education]. However, if the dataset is partitioned, for instance, by geographic regions, we can find a strong dependency relationship between [Education] and [Gender] in one partition, and a different strong dependency relationship between [Education] and [Income Level] in another partition that does not include the previous relationship. Therefore, local models can better represent the dependency relationships in data sub-domains, rather than having one model to represent the whole domain of the dataset.

As a natural approach to achieve that goal, the dataset $\mathbf{D}$ can be clustered into K data subsets by a clustering algorithm, where K is an input value, and each cluster (subset) D_k may contain instances labeled by different class values. Hence, unlike the class-based BMN approach, in the clustering-based BMN approach 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 $\mathbf{x}$ to its nearest cluster D_k (the cluster which the instance belongs to), and then uses the local BN classifier BNC_k, which is built on D_k, to compute the class value $C(\mathbf{x})$ that maximizes the posterior probability, as shown in Equation 4. As mentioned earlier, the clustering-based BMN approach is our focus in this work.

$$C(\mathbf{x}) = \underset{\forall c \in C}{\arg\max}\, P(c | \mathbf{x}, BNC_k), \mathbf{x} \in D_k \tag{4}$$

The graphical difference between class-based Bayesian multi-nets and clustering-based Bayesian multi-nets is illustrated in Figure 2.

2.4. Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta-heuristic search and optimization method that is inspired by the "intelligent" behaviour of natural ant colonies when they are foraging for food, and it has been widely used to solve (mainly combinatorial) optimization problems [8, 30]. There are also many works applying ACO to data mining, both in clustering and in classification – some of

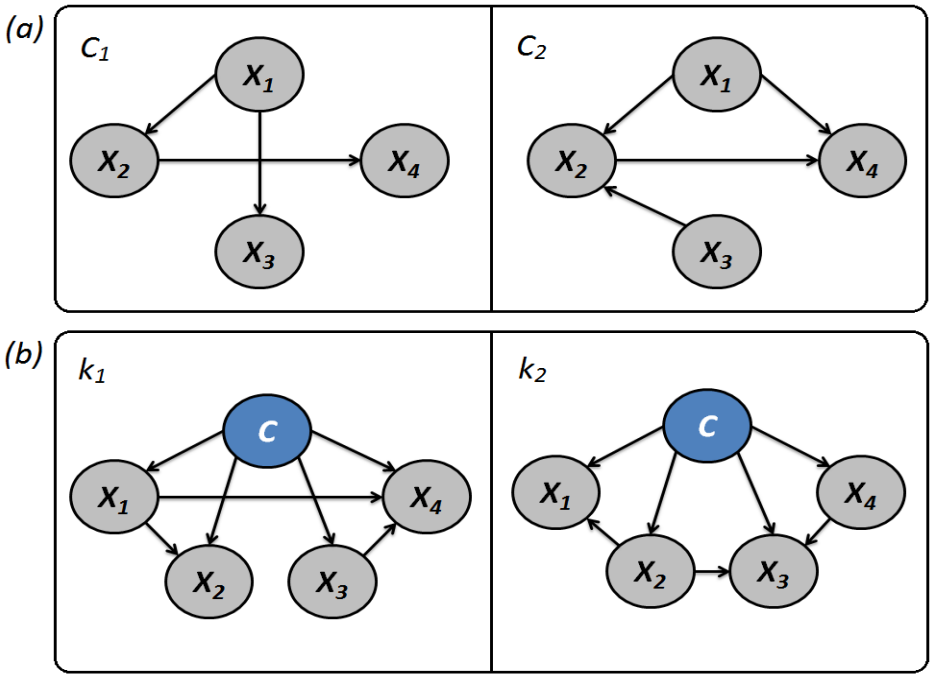

Figure 2: Given a dataset with four input attributes and a binary class variable: (a) represents a class-based BMN with two local BNs, one for each class value, (b) represents a clustering-based BMN with two arbitrary data clusters. Each local BN (classifier) has a different network structure that asserts the variable dependencies in its corresponding data subset. Note that the class variable is present in the local models of (b), unlike (a).

these works are briefly mentioned below. The basic principle of ACO is that a population of artificial ants cooperate with each other to find the best path in a graph, representing a candidate solution to the target problem. The way the artificial ants cooperate with each other is inspired by the way that natural ants cooperate to find the shortest path between two points in a given terrain, like their nest and a food source. The basic pseudo-code of an ACO algorithm, at a high level of abstraction, is presented in Algorithm 1.

Unlike conventional, greedy local search methods, in ACO the ants perform a global search for new solutions in the search space, influenced by the amounts of pheromone representing the qualities of solutions found so far. In nature, pheromone is a chemical substance which the ants are attracted to. In ACO, pheromone is a type of information with the analogous meaning of attracting artificial ants: the larger the amount of pheromone in a given region of the search space, the higher the probability that the ants will visit that region during the solution construction process.

When an ant constructs a complete candidate solution, it deposits an amount of pheromone proportional to the solution's quality in the region of the search space where that solution is located. Hence, with time the ants tend to con-

Algorithm 1 Pseudo-code of basic ACO algortihm

```
Begin ACO
ConstructionGraph ← Problem_definition;
Initialize();
best ← φ; /* best solution found so far */
repeat
    current ← ant.ConstructSolution()
    ApplyLocalSearch(current)
    if Quality(current) > Quality(best) then
        best ← current;
    end if ant.UpdatePheromone(current);

until termination_condition
return best;
End
```

verge to paths representing near-optimal solutions in the search space. The global search aspect of ACO stems from the fact that a population of ants explores many different regions of the search space in parallel, and they make probabilistic decisions about which region of the search space to move next. The global search behaviour of ACO makes it less likely to get trapped into local optima, compared to conventional (greedy and deterministic) local search methods. Note that, in one commonly used variation of the ACO algorithm, multiple ants construct solutions independently in one iteration before the iteration-best ant updates the pheromone to guide the search in the subsequent iterations. The existence of multiple ants is not mentioned in the high-level pseudocode of Algorithm 1 in order to keep it simple.

ACO has been employed for learning *general*-purpose BNs in several works [16, 17, 18, 19]. In the area of Bayesian classification, the authors have recently introduced ABC-Miner [20, 21] and its extended version, ABC-Miner+ [22], at present the only algorithms that use ACO for learning a BN *classifier* in the structure of a BAN and a Markov blanket respectively; 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 [10]. A recent and comprehensive survey on Ant-Miner and several of its variants or extended versions, as well as on several other types of ACO algorithms for data mining, is presented in [31]. Ant-Tree-Miner [32] and *c*ACDT [33] are recently-developed ACO algorithms for building decision trees.

In the field of clustering, ACO has introduced several algorithms, briefly reviewed in [13]. Amongst them, we extended the work in [14] (described in Section 4) for our clustering-based BMN classification algorithms. The choice of this particular ACO technique for clustering is based on its fitness in terms of problem representation and procedure adaptability to our learning task.

3. Data Clustering for Learning BMNs

3.1. Literature Review on BMN Learning

In class-based BMNs, building the local models for each data subset can be performed as a general BN learning process. A well-known greedy algorithm for learning a BN structure is Algorithm-B, proposed in [34], which searches for the network structure that optimizes the value of a scoring function, such as K2, cross-entropy or the Kullback distance [6, 35]. Chow-Liu tree Multi-net [6, 7, 35] is a remarkable algorithm for learning local BNs due to its simplicity and effectiveness. The algorithm builds a tree-like network structure, based on the mutual information between the variables.

An extension to the Chow-Liu tree Multi-net was proposed in [35], which involves maximizing the cross-class divergence. The Bayesian class-matched multi-net algorithm [36] is another extension that uses a scoring function based on detection-rejection behavior. The recursive Bayesian Classifier induction [37] can build multiple local BNs for the same class by further dividing its data subset recursively.

As for clustering-based BMN learning, it seems the "clustering-based" term has not been introduced in the BMN learning literature before this work. However, a distinct research direction in the field of learning BMNs has been exploiting the idea of building several BN classifiers based on arbitrary partitions of the data, using different techniques [38, 39, 40, 24, 41].

Broadly speaking, the idea of partitioning the dataset into arbitrary groups of instances to build local BN classifiers can be achieved in two ways. The first is recursive tree-like partitioning, in which the system chooses one attribute at a time to expand the partitioning tree with a set of branches – one branch for each nominal value or each interval of numerical values of the chosen attribute. The BN classifiers are built on the leaf nodes, where each leaf node represents a specific data subset characterized by the attribute values occurring in the path from the root node to the leaf node in question. The second is the clustering approach, in which a conventional clustering algorithm is utilized to partition the data into data subsets. The BN classifiers are built on the output clustering solution, and each cluster is characterized by the attribute values of the centroid (pivot instance) of the cluster.

Naïve-Bayes Tree (NBTree), introduced in [38], is the first algorithm that can be classified under the category tree-like partitioning. The algorithm follows the same procedure as the well-known decision tree C4.5 algorithm. That is, it recursively selects the attribute with the highest utility to expand the tree. If the utility of this attribute is significantly better than the utility of the current node, the attribute is used to further partition the current (sub) dataset; otherwise the current node is considered a leaf node, and a Naïve-Bayes classifier is built from the data subset in that leaf.

Another recent work in this category is learning Recursive Bayesian Multi-nets (RBMN) [39], which utilizes a greedy, heuristic decision tree learning approach with local Enhanced Naïve-Bayes (ENB) classifiers in the leaves. The

RBMN algorithm also tackles the problem of learning BN models from incomplete data via constructive induction with Bound and Collapse (BC) and Expectation Maximization (EM) methods [39]. Wrapper and filter methods to determine the branching attribute in BMN classifiers were introduced in [40].

As for the second category, where a direct partitioning of the dataset is applied using a clustering algorithm, Case-Based Bayesian Network Classifiers (CBBN) [24, 41] is the only approach – as far as we know - introduced in the area of clustering-based BMN learning. Since this is the most related approach to our work, it is described in the next subsection.

3.2. Clustering in the CBBN Algorithm

Santos et. al introduced the Case-Based Bayesian Network Classifier (CBBN) algorithm in [24], which utilizes what we call the "cluster-then-learn" two-phase approach. The key idea of this approach is using a clustering algorithm to partition the dataset into subsets. When the clustering phase is completely finished, a set of local BN classifiers, one for each subset, is built. In [24], the well-known K-means clustering algorithm was used. This algorithm iteratively selects K data instances from the dataset to be the centroids of the K clusters (where K is the user-specified number of clusters), assigns each instance to its nearest centroid, and recalculates each centroid by taking the mean value of each attribute across the instances in that centroid's cluster. These steps are repeated until the centroids converge to fixed values.

In K-means, an instance $\mathbf{x}$ is assigned to cluster j according to this formula:

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

where cen_k is the centroid of the k-th cluster. The distance between two instances is computed using the Euclidean distance, as follows:

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

where n is the number of attributes in the instance $\mathbf{x}$.

This kind of distance measure and centroid calculation, for assigning instances to clusters, is appropriate for datasets containing only continuous attributes. However, since most of the work in the field of BN classifier learning (including this work) focus on data with nominal attributes, some aspects of the basic K-means need to be modified to cope with such datasets.

The CBBN algorithm [24] used K-modes, a variation of the K-means algorithm adapted to nominal attributes. K-modes has the same overall structure of K-means. Nevertheless, how the distance between two instances is computed, and how the centroids of the clusters are updated are different in the two algorithms. In K-modes, the difference between two nominal values of the same attribute in two data instances is either 0, if the two values are the same, or 1, 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 of each attribute –i.e., the value with the highest number of occurrences across all 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.

4. An Overview of the ACO Algorithm for Clustering

The ACO algorithm for clustering [14] assigns each of the N instances in the dataset to one out of K clusters. The construction graph contains $N \times K$ decision components; each instance with each possible cluster assignment. Each ant in the colony starts with a list of N elements with unassigned clusters (an empty solution). Then the ant selects a cluster assignment from the construction graph for each element. 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. The procedure of the clustering solution construction is shown in Algorithm 2.

Algorithm 2 Pseudo-code of IB Clustering Solution Construction

```
Begin
ClustSolution = φ;
for i = 1 → N do
    k_i = ant.SelectClusterAssignment(i);
    ClustSolution[i] = k_i;
end for
return ClustSolution;
End
```

This method of clustering solution representation is referred to as "instance-based" (IB), where a solution consists of N elements, the index of an element represents the instance, and the element represents the cluster assignment of this instance. Figure 3 shows a graphical example of such a method.

After each ant constructs a candidate solution, the quality of the solution is evaluated. Then the best l ants (a user-specified parameter) perform pheromone update. 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 Euclidean distances between each data instance and the center of the cluster to which the instance belongs [14]. Hence, a candidate clustering solution S is evaluated accordingly:

$$Quality(S) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathbf{Euclidean}(\mathbf{x}_{ki}, cen_k), \tag{7}$$

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.

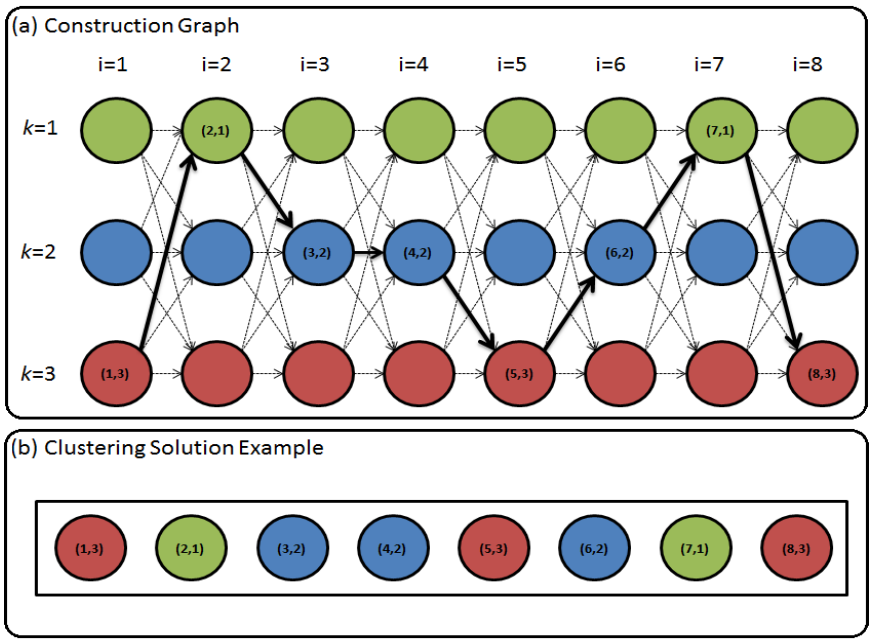

Figure 3: (a) represents the construction graph of a IB clustering representation for an 8-instance dataset and 3 clusters. Each node in the graph represents an (instance, cluster) assignment decision component. The nodes for instance i are connected to the nodes for instance i+1. The bold-face edges represent the ant's path to construct the example clustering solution in (b). Instances with same-colored nodes represents instances in the same cluster. According to this representation, the number of candidate clustering solutions is K^N.

For a more detailed discussion on the ACO algorithm for clustering, the reader is referred to [14].

5. A Brief Review of Extensions to the Clustering Technique for Classification

Proposed by the authors in [23], these extensions were motivated to overcome two drawbacks of the K-modes algorithm used by CBBN [24] for data clustering. More precisely, the K-modes clustering algorithm has two shortcomings in tackling nominal attributes. We proposed two extensions, one for each shortcoming in the clustering technique, in order to better achieve the classification objective of the learning approach. These extensions are briefly reviewed here, to make this paper more self-contained.

First, when measuring the distance between two nominal attribute values, K-modes returns a very coarse-grained value, either 0 (if the values are the same) or 1 (if they are different). Our first extension to K-modes avoids this problem, by measuring the similarity between instances with nominal attributes with the following value difference metric [42]:

$$\textbf{Similarity}(\mathbf{x_1}, \mathbf{x_2}) = 1 - \sum_{v=1}^{n} \sum_{l=1}^{|C|} |P(C_l|\mathbf{x_1}_v) - P(C_l|\mathbf{x_2}_v)|, \tag{8}$$

where n is the number of attributes and $P(C_l|\mathbf{x}_v)$ is conditional probability of the class value C_l given the class value $\mathbf{x}_v$. 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 attribute value $\mathbf{x}_v$ in the dataset. Accordingly, if a class value occurs frequently with two attribute values, then these two values are considered similar, same if the class value does not occur frequently with both of the attribute values. However, if a class value occurs frequently with an attribute value, but does not occur frequently with the other attribute values, then these two attribute values are considered dissimilar.

This formula has the advantage of returning a fine-grained value (a real-valued number in [0..1]) for the distance between two (unordered) nominal attribute values. It is also very suitable for our ultimate classification goal, since according to this formula two nominal attribute values (in different instances) are considered similar to the extent that they predict the same class. E.g., the values "single" and "divorced" of the attribute "marital status" might be more similar to each other (tending to predict the same class) than to the value "married", depending on the class attribute being predicted.

The second drawback of the K-modes algorithm is that, when computing the distance between an instance and the mode vector of the cluster (formed by the mode – most frequent value – of each nominal attribute), the mode vector is again representing very coarse-grained information. E.g., consider two binary attributes, one of them with relative frequencies of 60% and 40% for its "yes" and "no" values; and the other with relative frequencies of 90% and 10% for its "yes" and "no" values, respectively. The mode of both attributes is the value "yes", but the mode of the second attribute is much more representative of the value of the attribute in the training set. Unfortunately, however, when measuring the degree of membership of an instance to a cluster, K-modes measures only the distance between an instance and a cluster's mode vector using that coarse-grained notion of mode.

To avoid this problem, our second extension to K-modes consisted of measuring the degree of membership of an instance $\mathbf{x}$ to a cluster as the average similarity between that instance and all instances $\mathbf{x_i}$ in that cluster, and then assigning $\mathbf{x}$ to the cluster to which it has the highest degree of membership, as specified in the following equation:

$$Cluster(\mathbf{x}) = \underset{\forall\ k \in \{1,..,K\}}{\arg\max} \frac{\sum_{i=1}^{N_k} \textbf{Similarity}(\mathbf{x}, \mathbf{x}_i)}{N_k}, \tag{9}$$

where K is the number of clusters and N_k is the number of the instance in k-th cluster. For more details about these two extensions to the K-modes algorithm and their motivation, the reader is referred to [23].

6. Introducing the Medoid-based ACO Clustering Method

The medoid-based solution representation for ACO clustering is the first contribution of this work. The motivation behind introducing such a representation is to mitigate a problem in the previously described instance-based clustering solution representation, namely the fact that, according to the solution encoding scheme used by the IB method (i.e. a string of instance-cluster assignments for all the instances as shown in Figure 3), the mapping between a solution and its encoding is one-to-many. That is, the same clustering solution can be represented by several instance-based encodings, as illustrated in Figure 4. The redundancy of this kind of encoding enlarges the size of the search space, which affects the search's efficiency, and may have a noticeable impact on the effectiveness of the ACO algorithm in terms of the quality of the solution found.

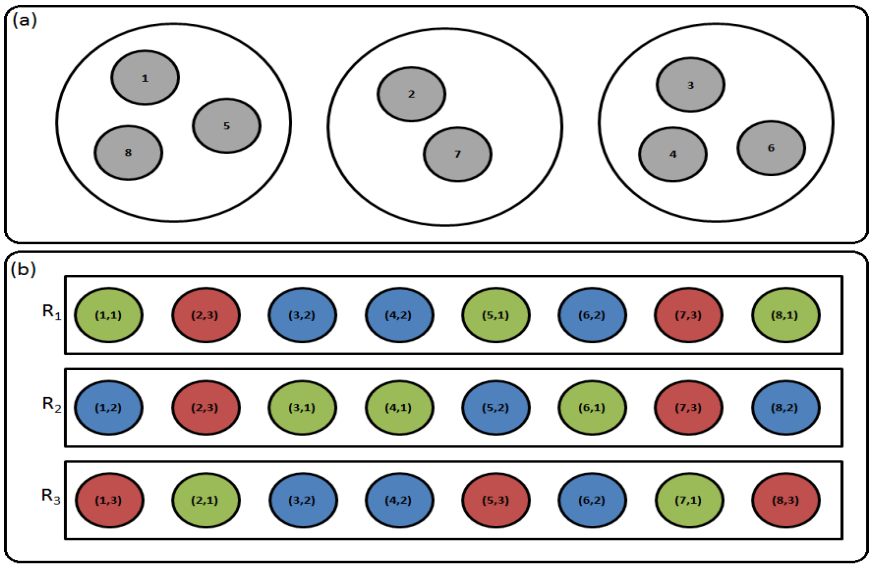

Figure 4: The clustering of the 8 instances shown in (a) (e.g instance 1, 5 and 8 are in the same cluster) can be represented by 3 different solution encodings (R_1, R_2, and R_3) in (b), using the instance-based method.

To mitigate such a problem, we propose the new medoid-based (MB) method for clustering solution representation. In the MB representation, the clustering solution is not represented by a full instance-cluster assignment for all the instances (whose construction graph would contain N times K nodes). Rather, a clustering solution is represented by the choice of K instances acting as the cluster medoids, where K is the number of clusters. More precisely, in the MB representation the construction graph is a complete (fully connected) graph containing only N decision components, each representing an instance that is a candidate cluster medoid.

The ant uses the random-proportional rule, as in Ant System [8] (except that our ACO does not use a heuristic function), to choose K instances based on the pheromone information associated with each decision component. The chosen instances represent an MB clustering solution, where the k-th element (instance) in the MB solution represents the medoid of k-th cluster. The medoid selection is performed under the constraint that a node can be visited at most once by an ant during a single solution construction. Figure 5 shows the construction graph and a sample solution in the context of the MB representation.

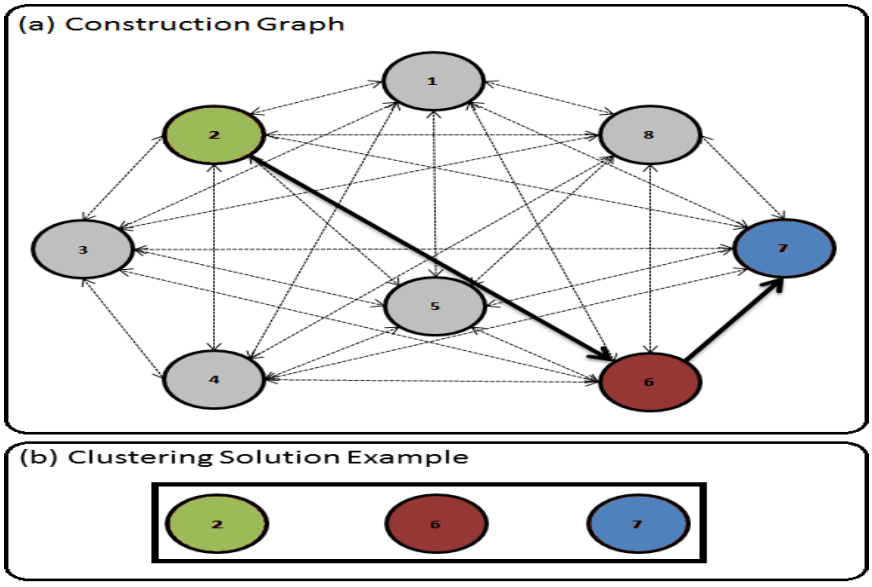

Figure 5: (a) represents the construction graph of a MB clustering problem for an 8-instance dataset and 3 clusters. Each node in the graph represents a candidate medoid for a cluster. Each node is connected to all the other nodes. The bold-face edges represent the ant's path to construct the example clustering solution in (b). The dark-colored nodes represent the instances selected to be the medoids of the clustering solution.

Once a MB candidate solution has been constructed by an ant, each instance i in the dataset is assigned to the k-th cluster where the similarity between instance i and the k-th medoid is the highest (comparing to the other medoids). The instance-medoid similarity is calculated according to the similarity measure in Equation 9, and this is one of the spots where the extended ACO algorithm proposed in this work utilizes a component of the ACO algorithm described in Section 5. Further details are discussed in Sections 7 and 8. Algorithm 3 shows the procedure of constructing a clustering solution using the MB representation.

According to the characteristics of the MB clustering solution representation, we have a smaller search space (N^K compared to K^N in the case of the IB representation, where $K << N$), and an efficient one-to-one solution representation encoding that should have a positive effect on performance.

Algorithm 3 Pseudo-code of MB Clustering Solution Construction

Begin
$ClustSolution = \phi$;
$MedoidList = \phi$;
for $k = 1 \ \rightarrow$ `K` **do**
  $M_k = ant.SelectClusterMedoid()$;
  Add M_k **to** $MedoidList$;
end for
for $i = 1 \ \rightarrow$ `N` **do**
  $k_i = GetNearestMedoid(MedoidList, i)$;
  $ClustSolution[i] = k_i$;
end for
return $ClustSolution$;
End

7. ACO Clustering Then BMN Learning

The first approach we utilize for the clustering-based BMN learning is the two-phase "cluster-then-learn" approach. This is a sequential two-phase approach, where in the first phase an ACO algorithm divides the dataset into clusters, and in the second phase a set of local BN classifiers, one for each cluster, is constructed. The motivations for this approach were mentioned in Section 1.1. In particular, it uses ACO as a global search method that is less likely to get trapped into local optima in the search space [8, 30] than greedy search methods. Intuitively, that global search should lead to more robust results by comparison with the K-means' search, which has the well-known drawback of being very sensitive to the values of the initial centroids (or the mode vectors, in the case of K-modes). This ACO-based "cluster-then-learn" approach has recently been proposed in [23], and a brief review of it is presented here, to make this paper more self-contained. The outline of the ACO "cluster-then-learn" approach is shown in Algorithm 4.

Each ant creates a complete candidate clustering solution, specifying the cluster to which each instance is assigned. At each iteration, the system evaluates the quality of each ant's clustering solution, according to a cluster-quality measure that uses the extensions discussed in Section 5, as specified in the following formula:

$$Q(ClustSolution) = \sum_{k=1}^{K} \frac{\sum_{i=1}^{N_k} \sum_{j=i}^{N_k} \mathbf{Similarity}((\mathbf{x}_{ki}, \mathbf{x}_{kj}))}{N_k}, \qquad (10)$$

where K is the number of the clusters, and N_k is the number of the instances in the k-th cluster.

Next, the algorithm selects the best clustering solution at the current iteration to undergo local search, which was designed to improve that solution's quality using a relatively fast search procedure. More precisely, the local search

Algorithm 4 Pseudo-code of ACO Clustering then BMN Learning

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**
  $BNC_k = LearnBNClassifier(ClustSolution_{gbest}(k))$;
  append BNC_k **to** BMN;
end for
return BMN;
End

consists of running one iteration of the K-Clust algorithm on the best clustering solution of the current iteration. The K-Clust algorithm essentially performs a local search that assigns each instance to its nearest cluster based on the previously discussed class-based similarity measure and procedure for computing the degree of membership of an instance to a cluster.

Next, the current iteration's best clustering solution (which has just been improved by the local search procedure) is used to update the pheromone trail according to that solution's quality. The algorithm also keeps track of the best clustering solution found by the search so far (up to the current iteration).

The aforementioned steps constitute one iteration of the ACO algorithm, and that process is iteratively repetead until a stopping criterion is satisfied – more precisely, until a maximum number of iterations is performed (specified by the user-defined parameter `max_iterations`) or until the algorithm converges in the sense that the best clustering solution found so far remains the same for a number of consecutive trials (specified by the user-defined `conv_iterations` parameter). At the end of the execution of the ACO clustering algorithm, a local BN classifier is built from each cluster in the best clustering solution output by the ACO algorithm.

The above ACO-based "cluster-then-learn" approach has been used to implement the Ant-ClustB$_{IB}$ algorithm proposed in [23]. That ACO algorithm uses an instance-based clustering solution representation, as discussed in Section 4. In this work we propose a new ACO algorithm for clustering, which uses the medoid-based clustering solution representation method, introduced in Section 6. The only difference between the two algorithms is that, in order to create a candidate clustering solution, Ant-ClustB$_{IB}$ carries out the procedure described in Algorithm 2; while Ant-ClustB$_{MB}$ carries out the procedure described in Algorithm 3.

8. ACO Clustering with BMN Learning

The novel ACO Clustering with BMN Learning approach is the second contribution of this work. We propose a clustering-based BMN learning approach that carries out the data clustering process as well as the local BN classifier construction process in a synergistic fashion via the ACO meta-heuristics. In other words, instead of having the clustering phase completely finished before the BMN learning phase starts, the "cluster-with-learn" approach integrates the clustering phase and the BMN learning phase in a single phase. In this approach, each ant in the colony creates a complete cluster-based BMN classifier, rather than just a clustering solution.

Figure 5 shows a conceptual comparison between the "cluster-then-learn" and "cluster-with-learn" approaches using ACO. It is worthy mentioning that no conventional clustering algorithm can fit such an approach, since the clustering process does not merely groups the similar instances together. Rather, the clustering process is merged with the classification learning process, so that the ACO meta-heuristic's search for good clusters is guided by the predictive

accuracy of candidate clustering solutions, as discussed in the next subsections.

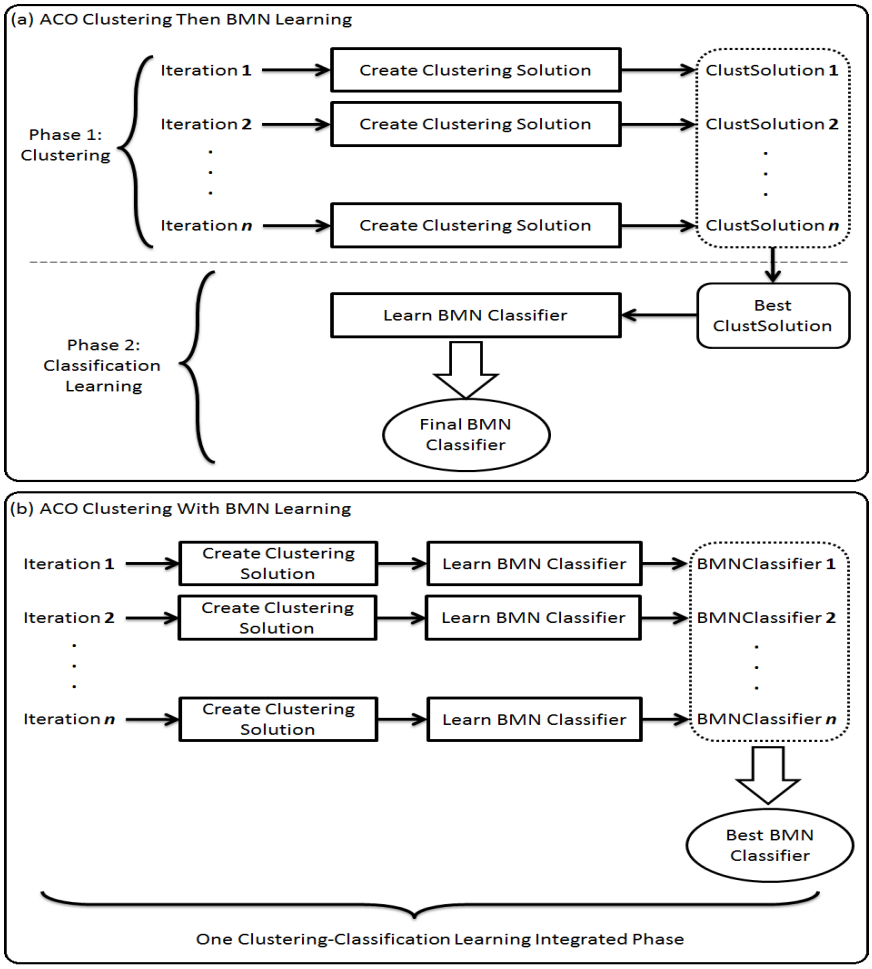

Figure 6: (a) represents the two-phase "cluster-then-learn" approach for learning clustering-based BMN classifiers. (b) represents the one-phase "cluster-with-learn" approach, in which the clustering and the classification learning process are integrated in a single phase.

8.1. Approach Outline

Algorithm 5 shows the outline of our proposed ACO-based "cluster-with-learn" approach. In fact, each ant_i constructs a candidate BMN_i classifier in two united steps. It constructs a clustering solution $ClustSolution_i$ and then learns a BMN_i based on the produced data clusters. It is important to emphasize that the candidate solution that each ant constructs in the "cluster-with-learn" approach is a complete cluster-based BMN classifier, rather than

a clustering solution as in the "cluster-then-learn" approach. This distinction is crucial in terms of solution quality evaluation and pheromone update, as discussed in the next subsection.

Algorithm 5 Pseudo-code of ACO Clustering with BMN Learning

```
Begin
K = input;
BMN_gbest = φ;
Q_gbest = 0;
InitializePheromoneAmounts();
t = 1;
repeat
    BMN_tbest = φ;
    Q_tbest = 0;
    for i = 1 → colony_size do
        ClustSolution_i = CreateClusteringSolution(ant_i);
        for k = 1 → K do
            BNC_k = LearnBNClassifier(ClustSolution_i(k));
            append BNC_k to BMN_i;
        end for
        Q_i = ComputeQuality(BMN_i);
        if Q_i > Q_tbest then
            BMN_tbest = BMN_i;
            Q_tbest = Q_i;
        end if
    end for
    PerformLocalSearch(BMN_tbest);
    UpdatePheromone();
    if Q_tbest > Q_gbest then
        BMN_gbest = BMN_tbest;
        Q_gbest = Q_tbest;
    end if
    t = t + 1;
until t = max_iterations or Convergence();
return BMN_gbest;
End
```

As the third contribution of this paper, we use this novel cluster-with-learn approach in two different new algorithms, namely ACO-ClustB$_{IB}$ and ACO-ClustB$_{MB}$. The former employs the instance-based clustering solution representation, discussed in Section 4, while the latter employs the medoid-based clustering solution representation, introduced in Section 6. The only difference between the two algorithms is the implementation of the $CreateClusteringSolution(ant)$ procedure in Algorithm 5.

8.2. Solution Quality Evaluation

The advantage of the integrated "cluster-with-learn" approach over the two-phase "cluster-then-learn" approach is that the product of each ant's trail is a complete candidate solution (a BMN classifier), not just a clustering solution. This has the benefit that the quality of a candidate solution can be directly evaluated according to its predictive accuracy, whose maximization is the ultimate objective of our work. Let us elaborate on this point.

On one hand, a solution created by an ant in the "cluster-then-learn" is a clustering solution, rather than a classifier. So, the solution can only be evaluated according to clustering-quality measures, concerning instance similarity and cluster cohesiveness (see Sections 5). However, the target of our work is to build a high-quality clustering-based BMN in terms of its classification accuracy. The gap between how an ACO candidate solution is evaluated (cluster cohesiveness) and the target objective (classification accuracy) is a drawback of the "cluster-then-learn" approach, since the clustering solution with the best cohesiveness does not necessarily lead to build the cluster-based BMN with the best classification accuracy.

On the other hand, in the "cluster-with-learn" approach, the candidate solution constructed by each ant is by itself a classifier (a cluster-based BMN classifier), which can be evaluated using a classification measure. When ants update the pheromone according to the classification accuracy of the constructed solution, the subsequent ants will follow this pheromone to construct the clustering solution that will lead to build accurate BMN classifiers (in the same ant trail). Therefore, the gap between the clustering solution quality and the BMN quality is eliminated, since the clustering solution is not evaluated until the BMN classifier is built upon it, and the whole product is evaluated as a cluster-based BMN classifier.

Accordingly, ACO clustering with BMN learning evaluates the quality of the constructed BMN classification solution using *accuracy*, a popular measure of predictive performance, to evaluate the constructed model, computed as follows:

$$Accuracy = \frac{|Correctly\_Classified\_Cases|}{|Validation\_Set|} \tag{11}$$

The cluster-with-learn approach uses the same pheromone update strategy, discussed in Section 7, as the cluster-then-learng approach; and it invokes the same K-Clust local search technique, discussed in Section 7. However, concerning the local search, after the instance-cluster membership changes, a new BMN classifier is learnt upon the new clusters, and the new solution (optimized by the local search) is accepted only if it improves the classification quality of the solution according to Equation 11.

9. Experimental Methodology

9.1. Comparative Evaluations

We compare our new ACO algorithms with three types of conventional algorithms widely used for learning BN classifiers: Naïve-Bayes, TAN, and GBN, where each algorithm builds a single classifier on the whole training set. In addition, we include in our experiments the well-known Chow-Liu tree Multi-net algorithm [7], which builds class-based BMN classifiers. This set of the four aforementioned algorithms act as a baseline for our comparative evaluation.

As for the clustering-based BMN learning, we compare our proposed ACO algorithms with the use of K-modes (described in Section 3.2) to cluster the data for building the local BN classifiers, denoted as K-ModesB. We also report the performance of K-ClustB, which uses our previously introduced extended clustering algorithm (described in Section 5) in the clustering phase before the classification phase. Table 1 presents the main properties of the used algorithms. In that table, "class-BMN" means a local BN is learned for each value of the class attribute, and "cluster-BMN" means a local BN classifier is learned for each cluster.

Table 1: Summary of the BN Classifier Learning Algorithms Used in the Experiments

Algorithm	Search Strategy	Output	Optimization
Naïve-Bayes	-	Naïve-Bayes	-
CL-Tree	Finding Max. Spanning Tree	TAN	Cond. Mutual Info.
Algorithm-B	Greedy Hill Climbing	GBN	K2 Scoring Function
CL-Tree MN	Finding Max. Spanning Tree	Class-BMN	Mutual Information
K-ModesB	Greedy Hill Climbing	Cluster-BMN	Sum Squared Error
K-ClustB	Greedy Hill Climbing	Cluster-BMN	Cohesiveness
Ant-ClustB$_{IB}$	ACO "cluster-then-learn" IB	Cluster-BMN	Cohesiveness
Ant-ClustB$_{MB}$	ACO "cluster-then-learn" MB	Cluster-BMN	Cohesiveness
ACO-ClustB$_{IB}$	ACO "cluster-with-learn" IB	Cluster-BMN	Predictive Accuracy
ACO-ClustB$_{MB}$	ACO "cluster-with-learn" MB	Cluster-BMN	Predictive Accuracy

Predictive accuracy was measured as follows. For each class, we first compute a conventional confusion matrix [3], where that class is the positive class and the other classes are grouped together to form the negative class. Next, we compute a conventional measure of accuracy for that class, i.e. the number of correctly classified (true positive or true negative) instances in the test set divided by the total number of instances in the test set. Then we report, as the predictive accuracy, the arithmetic average of those accuracies per class [43]. This procedure was used because it is more robust against the class imbalance problem (where some classes are much more frequent than others [25]), by comparison with the simpler procedure of computing a single confusion matrix and corresponding accuracy.

Note that, in the latter, if the vast majority of the instances (for example, 90%) belong to one class, one could obtain a very high value of accuracy by always predicting that majority class, trivially achieving a very high accuracy (90% in the above example), which would be misleading – since the other classes would never be predicted. The average accuracy per class avoids the above problem, since it heavily penalizes classifiers predicting a single class and rewards classifiers that predict well all the classes.

9.2. Experiment Setup

For the clustering-based learning algorithms, we performed 4 experiments for each dataset, with 4 different numbers of clusters: 2, 4, 6 and 8. For each number of clusters, we used two different BN classifier learning algorithms to build the local models, namely Naïve-Bayes and TAN. Hence, the number of results reported for each clustering-based learning algorithm on a given dataset is 8 (4 numbers of clusters $\times$ 2 local BN learning algorithms). As for the 4 baseline algorithms, each has one result for each dataset. The experiments were carried out using the well-known *stratified* 10-fold cross-validation procedure [3]. Since the ACO algorithms are stochastic, we ran each 10 times – using a different random seed each time – for each of the 10 iterations of the cross-validation procedure. Moreover, we did the same for the K-ModesB and K-ClustB algorithms to start with different initial centroids. The average of the 10 runs is used as the result of one iteration of the procedure.

The parameter configuration used in our experiments is shown in Table 2. The initial pheromone value of each decision component in the construction graph is set to 1 in all the ACO algorithms used in the experiments. In addition, only the quality of iteration-best solution is used for pheromone reinforcement. In order to achieve the effect of pheromone evaporation, we use the approach of normalizing all pheromone values after pheromone update, as proposed in [8]. More precisely, after increasing the pheromone values of the decision components chosen by the iteration-best ant, the pheromone value of each decision component (regardless of whether or not it was chosen by the iteration-best ant) is normalized by dividing its value by the total amount of pheromone values for all decision components in the construction graph. Since the decision components not chosen by the iteration-best ant have not been previously increased, their normalized value will be effectively reduced. Note that this approach avoids the need to specify a pheromone evaporation rate parameter.

Note that the same computational budget, represented by the number of ants per iteration (colony size) and the maximum number of iterations, has been assigned for each of the four ACO algorithms, for the sake of fair comparison. However, the maximum number of iterations may not utilized by if the algorithm converged earlier. Convergence occurs if the iteration-best ants produced the same solution quality for 10 consecutive iterations.

Predictive accuracy evaluation was performed using 30 public-domain datasets from the well-known UCI (University of California at Irvine) dataset repository. The main characteristics of the datasets can be found in [44]. Datasets having continuous attributes were discretized in a pre-processing step, using the

Table 2: Parameter settings used in experiments

Parameter	Value
`max_iterations`	1000
`colony_size`	10
`conv_ iterations`	10

C4.5-Disc algorithm [3], applied to the training set in each iteration of the cross-validation procedure.

10. Results and Analysis

10.1. Experimental Results

First, we report the results of the 4 baseline algorithms (which do not use any clustering algorithm) in a separate table. Then, for the clustering-based BMN learning algorithm, we report the results in 8 different tables, one for each experimental setup – i.e., each combination of a certain number of clusters and one of the two types of local BN classifier learning algorithms (NB or TAN) applied after the clustering (cluster-then-learn approach) or with the clustering (cluster-with-learn approach). We also included the results of Naïve-Bayes as a baseline (without clustering) in the result tables of the clustering-based experiments where Naïve-Bayes is used as a local BN classifier, to facilitate the comparisons. Likewise, we included the results of TAN as a baseline in the clustering-based tables where TAN is the local BN classifier. In each result table, the entry with the best result for each dataset is shown in boldface.

The last row of each result table shows the average rank of each algorithm in the experimental setup associated with that table. The average rank for a given algorithm g is obtained by first computing the rank of g on each dataset individually. The individual ranks are then averaged across all datasets to obtain the overall average rank. Note that the lower the value of the rank, the better the algorithm.

■ **Baseline Algorithms** – Table 3 shows the predictive accuracy results of the four baseline algorithms. As shown, the CL-Tree MN algorithm (TAN-MN), which uses the class-based partitioning approach for building local TAN classifiers, achieved the best overall ranking of 1.3, and obtained the best results in 25 (out of 30) datasets. The conventional TAN came in the second place with overall ranking of 2.2 and achieved the best results in 10 datasets. Algorithm-B (GBN) and Naïve-Bayes came in the third and the fourth places, obtaining overall rankings of 2.2 and 3.7 respectively. Naïve-Bayes achieved the best results in only 1 dataset.

■ **2-Cluster BMN Learning with Naïve-Bayes** – Table 4 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 2

clusters and Naïve-Bayes as local BN classifier, along with the results of the baseline Naïve-Bayes. As shown, ACO-ClustB$_{MB}$ obtained the best overall ranking of 2.0, achieving the best results in 20 datasets; followed by ACO-ClustB$_{IB}$ which obtained an overall ranking of 2.2 and achieved the best results in 17 datasets. TAN-MN, the best performing baseline, came in the third place with overall ranking of 2.7.

■ **2-Cluster BMN Learning with TAN** – Table 5 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 2 clusters and TAN as local BN classifier, along with the results of the baseline TAN. As shown, ACO-ClustB$_{MB}$ obtained the best ranking of 2.0, achieving the best results in 18 datasets; followed by ACO-ClustB$_{IB}$ which obtained an overall ranking of 2.1 and achieved the best results in 16 datasets. Ant-ClustB$_{IB}$ and Ant-ClustB$_{MB}$ obtained 3.7 and 4.1 rankings and came in the third and the fourth places respectively. TAN-BMN obtained 5.2 overall ranking.

■ **4-Cluster BMN Learning with Naïve-Bayes** – Table 6 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 4 clusters and Naïve-Bayes as local BN classifier, along with the results of the baseline Naïve-Bayes. As shown, ACO-ClustB$_{MB}$ obtained the best overall ranking of 1.5, achieving the best results in 24 datasets; followed by ACO-ClustB$_{IB}$ which obtained 1.8 as an overall ranking and achieved the best results in 13 datasets. Ant-ClustB$_{MB}$ and Ant-ClustB$_{IB}$ obtained 4.0 and 4.8 overall ranking and came in the third and the fourth places respectively. TAN-MN came in the fifth place after Ant-ClustB$_{MB}$ with overall average ranking of 4.1.

■ **4-Cluster BMN Learning with TAN** – Table 7 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 4 clusters and TAN as local BN classifier, along with the results of the baseline TAN. As shown, ACO-ClustB$_{MB}$ obtained the best ranking of 1.6, achieving the best results in 26 datasets; followed by ACO-ClustB$_{IB}$ which obtained 2.2 as an overall ranking and achieved the best results in 13 datasets. Ant-ClustB$_{MB}$ and Ant-ClustB$_{IB}$ obtained 3.6 and 4.4 rankings and came in the third and the fourth places respectively. TAN-BMN obtained 5.6 overall ranking.

■ **6-Cluster BMN Learning with Naïve-Bayes** – Table 8 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 6 clusters and Naïve-Bayes as local BN classifier, along with the results of the baseline Naïve-Bayes. As shown, ACO-ClustB$_{MB}$ obtained the best overall ranking of 1.6, achieving the best results in 25 datasets; followed by ACO-ClustB$_{IB}$ which obtained 2.2 as an overall ranking and achieved the best results in 11 datasets. Ant-ClustB$_{IB}$ and Ant-ClustB$_{MB}$ obtained 4.1 and 4.2 overall rankings and came in the third and the fourth places respectively, while TAN-MN came in the fifth place with 4.5 overall ranking.

■ **6-Cluster BMN Learning with TAN** – Table 9 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 6 clusters and TAN as local BN classifier, along with the results of the baseline TAN. As shown, ACO-ClustB$_{MB}$ obtained the best ranking of 1.7, achieving the best results in 24 datasets; followed by ACO-ClustB$_{IB}$ which obtained 2.1 as an overall ranking and achieved the best results in 18 datasets. Ant-ClustB$_{MB}$ and Ant-ClustB$_{IB}$ obtained 3.5 and 4.6 rankings and came in the third and the fourth places respectively. TAN-BMN obtained 5.9 overall ranking.

■ **8-Cluster BMN Learning with Naïve-Bayes** – Table 10 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 8 clusters and Naïve-Bayes as local BN classifier, along with the results of the baseline Naïve-Bayes. As shown, ACO-ClustB$_{MB}$ obtained the best overall ranking of 1.6, achieving the best results in 23 datasets; followed by ACO-ClustB$_{IB}$ which obtained 2.0 as an overall ranking and achieved the best results in 11 datasets. Ant-ClustB$_{MB}$ and Ant-ClustB$_{IB}$ obtained 4.0 and 4.3 overall rankings and came in the third and the fourth places respectively, while TAN-MN came in the fifth place with 4.7 overall ranking.

■ **8-Cluster BMN Learning with TAN** – Table 11 shows the predictive accuracy results for the clustering-based BMN learning algorithms with 8 clusters and TAN as local BN classifier, along with the results of the baseline TAN. As shown, ACO-ClustB$_{MB}$ obtained the best ranking of 1.5, achieving the best results in 28 datasets; followed by ACO-ClustB$_{IB}$ which obtained 2.2 as an overall ranking and achieved the best results in 10 datasets. Ant-ClustB$_{MB}$ and Ant-ClustB$_{IB}$ obtained 4.0 and 4.5 rankings and came in the third and the fourth places respectively. TAN-BMN obtained 6.0 overall ranking.

10.2. Statistical Significance Results

We used the non-parametric Friedman test with the Holm's post-hoc test [45, 46], with respect to the average rank. The statistical test is carried out on each of the 8 experimental setups, each including the results of 8 algorithms: the results of the 6 clustering-based BMN algorithms, the results of the corresponding baseline algorithm without clustering (i.e. Naïve-Bayes or TAN, depending on each experimental setup), and the results of TAN-MN as best performing algorithm in the baseline set. We performed the Friedman test using the freely available Java program suggested by Garcia et al. in [46], which applies the tests with two different statistical significance levels: 5% and 10%. Table 12 shows the significance level at which the difference of predictive accuracy between two algorithms was considered statistically significant, according to the results of the Friedman and Holm's tests, when comparing each of the four proposed ACO algorithms and the extended K-clust algorithm (on columns) against the other algorithms (on the rows), for each experimental setup - i.e., each pair of a type of local BN classifier (Naïve-Bayes or TAN) and a given number of clusters K (2, 4, 6 or 8). In that table, an entry with the symbol "-" denotes that the

Table 3: Predictive Accuracy (%) Results for the Baseline Algorithms

Dataset	Naïve-Bayes	TAN	GBN	TAN-MN
abalone	56.1	67.4	68.3	**71.2**
balance	76.1	76.4	75.5	**77.8**
breast-w	92.5	**95.7**	93.8	**95.7**
car	85.3	**93.6**	87.1	92.8
chess	86.5	**91.5**	88.8	**91.5**
contraceptive	50.8	**58.2**	54.6	**58.2**
credit-a	79.3	81.1	81.7	**82.8**
credit-g	75.4	82.3	77.1	**85.2**
dermatology	96.2	**97.8**	97.2	97.2
ecoli	59.1	66.2	55.6	**66.4**
glass	61.6	71.1	68.4	**73.3**
hayes-roth	84.0	84.0	85.1	**87.6**
heart-c	54.6	63.3	66.1	**68.8**
heart-s	81.8	80.3	82.5	**85.4**
ionosphere	88.6	90.7	90.7	**92.1**
iris	92.2	90.2	92.9	**94.2**
lung-c	**82.5**	68.4	73.2	76.6
monk	60.5	**66.1**	61.6	62.7
mushrooms	95.8	**98.8**	96.1	**98.8**
nursery	90.1	**94.2**	92.7	**94.2**
parkinsons	84.5	**91.7**	84.5	88.4
page block	87.2	92.3	91.6	**92.9**
post-operative	69.5	71.1	76.6	**79.8**
segmentation	93.8	93.4	93.1	**94.5**
soybean	47.6	53.5	53.2	**55.2**
SPECT	72.3	76.2	75.0	**77.5**
tic-tac-to	68.7	76.6	74.3	**78.4**
vote	87.5	92.1	90.3	**94.1**
wine	95.6	**97.3**	97.0	**97.3**
yeast	59.7	61.2	60.2	**62.1**
Avg. Rank	3.7	2.2	2.8	1.3

Table 4: Predictive Accuracy (%) Results for 2-Cluster BMN Learning with Naïve-Bayes

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	Naïve	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	56.1	61.3	64.3	63.8	64.1	65.8	**66.8**
balancee	76.1	76.5	76.5	76.8	76.9	**77.5**	**77.5**
breast-w	92.5	91.4	92.8	93.1	93.4	94.8	**95.7**
car	85.3	86.9	87.3	88.8	88.8	92.3	**92.5**
chess	86.5	87.6	88.6	86.8	82.4	91.5	**92.1**
contraceptive	50.8	51.7	53.4	53.9	53.4	**55.8**	54.3
credit-a	79.3	80.8	81.5	81.7	81.8	**83.1**	**83.1**
credit-g	75.4	76.8	77.0	77.0	77.6	**78.7**	78.2
dermatology	96.2	97.2	97.6	97.8	98.1	98.1	**99.0**
ecoli	59.1	61.4	62.6	63.9	62.6	**64.1**	**64.1**
glass	61.6	64.2	65.7	65.7	65.8	**67.9**	66.5
hayes-roth	84.0	84.1	84.7	84.7	84.7	85.0	**85.2**
heart-c	54.6	61.5	64.1	65.9	65.9	**68.4**	68.1
heart-s	81.8	82.7	82.7	83.8	84.4	86.1	**86.9**
ionosphere	88.6	92.8	93.8	93.8	**94.5**	94.1	94.1
iris	92.2	92.2	92.4	92.4	92.4	**93.5**	**93.5**
lung-c	82.5	91.6	91.6	91.6	93.7	**95.7**	**95.7**
monk	60.5	58.6	58.6	59.5	59.8	62.4	**62.8**
mushrooms	95.8	96.7	97.2	97.2	97.6	98.0	**98.2**
nursery	90.1	91.3	92.1	92.6	94.6	**95.8**	95.0
parkinsons	84.5	94.2	95.0	95.0	97.2	**97.8**	**97.8**
page block	87.2	88.5	88.5	88.7	87.8	89.3	**89.4**
post-operative	69.5	68.0	68.2	71.9	72.1	73.0	**74.7**
segmentation	93.8	94.2	94.5	94.6	93.6	**95.1**	94.7
soybean	47.6	42.7	43.6	43.8	44.7	53.3	**56.7**
SPECT	72.3	73.9	75.1	76.5	76.9	**78.7**	78.1
tic-tac-to	68.7	71.3	75.2	76.3	75.3	**78.7**	77.5
vote	87.5	91.6	94.3	94.3	94.9	**96.2**	**96.2**
wine	95.6	94.0	95.6	95.8	97.3	**97.9**	**97.9**
yeast	59.7	60.1	60.5	60.1	61.8	**64.7**	62.6
Avg. Rank	7.5	6.8	5.6	4.9	4.3	2.2	2.0

Table 5: Predictive Accuracy (%) Results for 2-Cluster BMN Learning with TAN

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	TAN	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	67.4	65.8	67.9	67.9	67.4	**70.3**	70.0
balancee	76.4	77.4	77.4	77.8	**78.9**	**78.9**	77.8
breast-w	95.7	95.9	96.7	97.5	96.5	**97.8**	**97.8**
car	93.6	94.2	95.1	96.6	96.6	97.4	**97.8**
chess	91.5	94.0	94.7	95.3	95.7	**96.7**	96.4
contraceptive	58.2	60.0	60.2	61.9	61.5	**64.7**	**64.7**
credit-a	81.1	83.6	84.4	84.5	84.7	**86.6**	86.4
credit-g	82.3	84.2	85.1	85.9	85.9	87.6	**88.1**
dermatology	97.8	98.0	**98.9**	**98.9**	**98.9**	**98.9**	**98.9**
ecoli	66.2	**66.7**	**66.7**	64.9	65.5	**66.7**	**66.7**
glass	71.1	71.6	71.6	72.4	72.4	73.6	**73.9**
hayes-roth	84.0	84.5	85.2	85.4	83.6	85.2	**85.6**
heart-c	63.3	65.9	68.9	69.8	69.8	72.2	**74.3**
heart-s	80.3	82.5	84.7	86.8	86.8	88.7	**89.1**
ionosphere	90.7	94.2	95.7	**97.4**	95.8	96.6	96.6
iris	90.2	90.2	90.6	90.6	90.8	**92.4**	90.6
lung-c	68.4	72.8	75.7	74.2	77.1	**82.6**	80.7
monk	66.1	68.1	69.8	71.9	70.5	72.0	**72.9**
mushrooms	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**
nursery	94.2	94.1	94.2	94.2	94.2	95.1	**96.0**
parkinsons	91.7	92.3	94.1	96.0	96.2	**98.1**	98.0
page block	92.3	93.2	93.4	93.6	93.2	**94.7**	94.1
post-operative	71.1	75.4	75.4	82.3	82.3	82.0	**85.2**
segmentation	93.4	93.9	93.9	94.5	94.7	**95.2**	94.7
soybean	53.5	55.9	56.3	61.6	61.6	66.2	**66.6**
SPECT	76.2	82.6	82.8	79.1	73.9	85.5	**85.6**
tic-tac-to	76.6	82.2	82.2	**87.6**	81.8	**87.6**	**87.6**
vote	92.1	93.2	93.5	94.9	95.8	96.1	**96.5**
wine	97.3	98.9	98.9	**98.9**	98.9	98.9	98.9
yeast	61.2	62.4	62.4	64.4	**67.2**	**67.2**	67.0
Avg. Rank	7.3	6	4.9	3.7	4.1	2.1	2

Table 6: Predictive Accuracy (%) Results for 4-Cluster BMN Learning with Naïve-Bayes

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	Naïve	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	56.1	62.7	65.9	66.2	67.7	68.2	**74.7**
balancee	76.1	76.8	76.9	77.4	77.8	**77.9**	77.8
breast-w	92.5	93.1	98.1	97.7	98.3	**98.6**	98.5
car	85.3	88.7	92.7	92.9	92.9	96.2	**96.9**
chess	86.5	89.4	90.1	91.9	90.5	94.3	**94.5**
contraceptive	50.8	53.6	55.7	56.1	56.8	59.0	**60.0**
credit-a	79.3	81.7	82.7	82.7	82.3	84.1	**85.8**
credit-g	75.4	78.1	79.1	74.6	79.5	82.6	**83.5**
dermatology	96.2	97.6	98.0	98.4	98.4	**99.1**	98.9
ecoli	59.1	62.6	65.2	60.8	61.5	**66.7**	**66.7**
glass	61.6	64.8	66.1	66.3	66.5	71.3	**71.5**
hayes-roth	84.0	85.0	84.1	84.4	85.6	87.5	**87.8**
heart-c	54.6	66.4	69.8	71.0	72.8	74.1	**74.7**
heart-s	81.8	83.2	84.6	85.8	85.9	**98.8**	90.6
ionosphere	88.6	91.8	93.8	91.8	93.8	94.8	**95.4**
iris	92.2	92.2	92.2	92.2	92.2	**94.5**	**94.5**
lung-c	82.5	92.5	95.0	94.8	94.8	**95.7**	**95.7**
monk	60.5	60.6	60.9	61.7	62.9	66.2	**71.2**
mushrooms	95.8	96.9	97.2	97.8	98.0	**98.8**	98.0
nursery	90.1	92.0	92.6	92.8	94.0	**96.4**	**96.4**
parkinsons	84.5	95.1	96.1	96.5	97.2	**98.0**	**98.0**
page block	87.2	88.5	89.8	89.6	89.8	**92.9**	92.5
post-operative	69.5	72.1	72.1	75.2	75.8	80.7	**81.5**
segmentation	93.8	94.3	95.6	95.9	95.9	96.9	**97.2**
soybean	47.6	50.6	52.5	52.6	52.8	63.6	**69.6**
SPECT	72.3	80.8	82.1	83.6	84.1	87.5	**87.6**
tic-tac-to	68.7	74.2	79.1	79.9	79.9	82.1	**86.4**
vote	87.5	91.9	94.9	95.7	95.7	96.8	**96.9**
wine	95.6	96.2	96.2	95.8	96.2	**98.9**	**98.9**
yeast	59.7	60.5	62.6	62.9	61.1	**67.9**	**67.9**
Avg. Rank	7.9	6.6	5.2	4.8	4.0	1.8	1.5

Table 7: Predictive Accuracy (%) Results for 4-Cluster BMN Learning with TAN

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	TAN	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	67.4	67.4	67.5	67.6	67.9	72.2	**76.7**
balancee	76.4	76.6	77.4	77.4	73.7	**77.8**	**77.8**
breast-w	95.7	96.5	**98.9**	98.5	98.6	98.7	**98.9**
car	93.6	95.2	96.5	96.8	96.8	97.2	**97.4**
chess	91.5	94.0	94.8	94.8	94.2	97.7	**97.8**
contraceptive	58.2	60.5	62.2	63.9	64.0	**64.9**	**64.9**
credit-a	81.1	82.8	84.4	85.3	85.0	**88.1**	86.4
credit-g	82.3	84.2	85.9	87.0	87.3	89.9	**90.2**
dermatology	97.8	98.8	98.8	98.8	**98.9**	**98.9**	**98.9**
ecoli	66.2	66.2	66.7	65.4	**66.8**	66.7	66.7
glass	71.1	71.0	71.9	68.6	71.8	71.8	**73.3**
hayes-roth	84.0	84.3	85.0	84.4	85.6	87.5	**87.8**
heart-c	63.3	73.8	73.8	74.2	74.0	**78.6**	76.3
heart-s	80.3	82.9	84.1	90.8	93.7	94.2	**95.1**
ionosphere	90.7	92.1	92.9	93.4	94.5	**95.5**	**95.5**
iris	90.2	**94.2**	**94.2**	**94.2**	**94.2**	**94.2**	**94.2**
lung-c	68.4	92.5	95.0	94.8	94.8	**95.7**	**95.7**
monk	66.1	65.7	66.8	72.1	73.7	**78.8**	**78.8**
mushrooms	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**
nursery	94.2	96.0	96.0	96.3	96.4	96.6	**96.8**
parkinsons	91.7	95.7	96.1	96.5	97.6	98.1	**98.4**
page block	92.3	93.6	94.1	94.8	94.8	**95.6**	94.8
post-operative	71.1	75.4	77.4	79.3	79.8	82.4	**83.6**
segmentation	93.4	94.3	95.6	95.9	95.9	96.9	**97.2**
soybean	53.5	60.1	63.8	65.4	66.7	67.9	**69.9**
SPECT	76.2	85.2	88.9	82.0	87.1	**90.5**	**90.5**
tic-tac-to	76.6	90.4	92.6	92.7	94.2	97.2	**97.6**
vote	92.1	95.8	95.8	95.9	95.9	98.1	**98.5**
wine	97.3	96.2	95.8	96.2	**98.9**	**98.9**	**98.9**
yeast	61.2	64.0	64.4	65.5	64.5	68.1	**68.4**
Avg. Rank	7.4	6.1	4.7	4.4	3.6	2.2	1.6

Table 8: Predictive Accuracy (%) Results for 6-Cluster BMN Learning with Naïve-Bayes

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	Naïve	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abl	56.1	63.8	66.1	66.9	65.3	69.6	**74.8**
bal	76.1	76.9	77.2	77.2	75.4	**77.4**	**77.4**
bcw	92.5	95.3	98.4	98.6	98.4	**98.7**	**98.7**
car	85.3	92.7	92.9	93.3	94.5	**96.7**	96.5
chess	86.5	91.2	92.4	93.2	93.8	94.2	**96.2**
cmc	50.8	53.8	57.8	57.6	57.9	61.4	**61.7**
crd-a	79.3	82.8	83.2	83.8	84.0	**86.5**	86.3
crd-g	75.4	80.0	80.4	81.5	81.5	**86.1**	85.8
drm	96.2	96.5	97.9	97.9	97.3	**98.1**	**98.1**
ecoli	59.1	64.2	66.1	**66.3**	66.1	66.1	66.1
glass	61.6	65.2	66.8	67.8	68.2	69.9	**71.3**
hay	84.0	84.7	**85.0**	84.8	84.8	**85.0**	**85.0**
hrt-c	54.6	69.8	71.9	73.6	73.8	74.9	**75.4**
hrt-s	81.8	85.2	86.1	87.2	87.5	91.2	**92.4**
iono	88.6	90.8	90.4	92.9	93.0	93.0	**95.7**
iris	92.2	92.2	92.2	92.2	92.2	94.8	**95.4**
lng-c	82.5	90.6	90.7	91.8	91.5	91.8	**92.7**
monk	60.5	61.7	61.7	63.2	64.7	67.6	**69.1**
mush	95.8	96.1	96.1	96.2	98.0	98.0	**98.8**
nurs	90.1	93.0	93.0	94.1	94.2	95.2	**96.6**
park	84.5	95.4	96.7	97.0	**97.2**	**97.2**	**97.2**
pbd	87.2	88.0	89.0	89.2	89.5	92.6	**92.9**
pop	69.5	73.1	70.8	79.9	81.1	82.3	**82.5**
seg	93.8	94.6	95.8	95.9	94.4	**97.2**	**97.2**
soy	47.6	52.8	61.1	59.6	61.9	**70.7**	68.4
SPECT	72.3	82.7	86.9	86.9	86.9	**90.5**	**90.5**
ttt	68.7	79.8	87.8	88.9	87.8	92.7	**94.1**
vot	87.5	92.8	95.8	96.5	91.1	97.5	**97.8**
wine	95.6	96.2	96.2	**97.3**	97.0	97.0	**97.3**
yst	59.7	42.6	64.1	64.9	66.9	68.1	**68.4**
Avg. Rank	7.8	6.6	5.2	4.1	4.2	2.2	1.6

Table 9: Predictive Accuracy (%) Results for 6-Cluster BMN Learning with TAN

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	TAN	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abl	67.4	67.5	67.8	67.4	67.9	72.8	**75.9**
bal	76.4	77.2	77.4	77.4	76.5	**77.8**	**77.8**
bcw	95.7	98.4	**98.7**	98.5	**98.7**	**98.7**	**98.7**
car	93.6	94.5	96.3	97.6	96.8	**98.3**	98.0
chess	91.5	93.2	93.2	94.1	94.6	96.2	**97.0**
cmc	58.2	62.5	64.7	63.4	64.1	**68.4**	**68.4**
crd-a	81.1	84.4	87.5	86.8	87.5	**90.1**	88.8
crd-g	82.3	91.8	93.6	92.7	93.6	**95.5**	94.2
drm	97.8	96.5	97.2	97.2	97.9	**98.1**	**98.1**
ecoli	66.2	64.2	66.1	**66.3**	66.1	66.1	66.1
glass	71.1	69.8	71.2	69.8	71.2	**71.3**	**71.3**
hay	84.0	84.0	84.2	84.5	85.0	87.6	**87.8**
hrt-c	63.3	77.6	80.1	80.2	81.8	**86.5**	86.2
hrt-s	80.3	91.9	94.2	94.2	95.9	**97.2**	**97.2**
iono	90.7	92.1	92.1	92.7	94.3	94.3	**96.6**
iris	90.2	94.2	95.4	96.4	96.4	95.2	**96.8**
lng-c	68.4	91.0	91.0	91.7	91.9	**95.6**	**95.6**
monk	66.1	72.8	74.1	75.4	76.7	**79.6**	**79.6**
mush	95.8	96.1	96.1	96.2	98.0	98.0	**98.8**
nurs	94.2	95.0	95.6	96.8	96.8	97.2	**97.4**
park	91.7	94.3	95.7	96.2	96.3	**98.0**	**98.0**
pbd	92.3	93.9	94.4	94.2	94.4	**95.6**	94.8
pop	71.1	80.1	82.5	83.5	84.8	86.3	**86.9**
seg	93.4	96.0	96.1	95.7	96.4	**97.9**	**97.9**
soy	53.5	62.5	63.6	64.6	64.9	**74.7**	**74.7**
SPECT	76.2	88.2	**90.5**	89.9	89.5	**90.5**	**90.5**
ttt	76.6	90.4	92.8	95.9	95.9	96.2	**96.6**
vot	92.1	95.8	95.8	96.5	97.9	98.1	**98.5**
wine	97.3	94.2	95.8	94.2	**97.8**	97.5	**97.8**
yst	61.2	64.9	66.0	66.0	67.3	**68.9**	**68.9**
Avg. Rank	7.3	6.3	4.8	4.6	3.5	2.1	1.7

Table 10: Predictive Accuracy (%) Results for 8-Cluster BMN Learning with Naïve-Bayes

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	Naïve	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	56.1	66.2	66.8	66.8	66.8	68.8	**72.9**
balancee	76.1	76.6	77.2	**77.8**	**77.8**	**77.8**	**77.8**
breast-w	92.5	96.8	98.0	98.2	98.0	98.5	**98.7**
car	85.3	90.5	93.2	94.1	94.1	95.7	**96.4**
chess	86.5	90.1	91.4	93.3	93.6	95.4	**96.3**
contraceptive	50.8	56.5	57.7	57.7	57.9	59.9	**60.7**
credit-a	79.3	84.1	84.1	85.2	84.1	**86.1**	85.3
credit-g	75.4	79.8	81.5	81.6	81.5	**86.5**	85.9
dermatology	96.2	98.0	98.0	96.5	96.7	98.7	**98.8**
ecoli	59.1	62.8	65.5	65.5	66.0	**66.1**	**66.1**
glass	61.6	67.2	68.8	69.9	68.8	71.3	**71.6**
hayes-roth	84.0	84.0	85.0	85.6	85.6	**85.8**	85.6
heart-c	54.6	70.2	72.4	72.6	73.3	75.6	**77.2**
heart-s	81.8	86.2	86.2	88.4	88.7	90.7	**91.2**
ionosphere	88.6	91.8	93.8	91.8	93.8	94.8	**95.4**
iris	92.2	92.2	92.2	92.2	92.2	**94.5**	**94.5**
lung-c	82.5	90.0	90.2	92.8	92.2	94.0	**94.5**
monk	60.5	61.0	62.8	63.9	63.9	66.8	**68.0**
mushrooms	95.8	96.9	97.2	97.8	98.0	**98.8**	98.0
nursery	90.1	92.0	92.6	92.8	94.2	**96.0**	**96.0**
parkinsons	84.5	92.8	94.2	94.5	95.9	96.0	**96.8**
page block	87.2	90.2	90.2	90.2	92.9	**93.3**	92.9
post-operative	69.5	70.6	74.7	72.1	77.3	79.5	**79.8**
segmentation	93.8	96.2	96.4	96.4	96.6	96.6	**97.1**
soybean	47.6	55.4	56.1	62.6	60.6	69.1	**69.6**
SPECT	72.3	84.7	84.7	88.8	88.8	89.5	**89.7**
tic-tac-to	68.7	82.5	82.9	90.4	89.4	92.2	**92.8**
vote	87.5	95.8	95.7	95.8	95.8	**96.9**	96.5
wine	95.6	96.2	96.2	**97.3**	97.0	97.0	**97.3**
yeast	59.7	65.1	65.1	64.7	63.7	**68.7**	68.4
Avg. Rank	7.8	6.2	5.2	4.3	4	2	1.6

Table 11: Predictive Accuracy (%) Results for 8-Cluster BMN Learning with TAN

				Cluster-Then-Learn		Cluster-With-Learn	
Dataset	TAN	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
abalone	67.4	67.4	67.5	67.5	67.6	71.4	**73.6**
balancee	76.4	76.6	77.4	77.4	73.7	**77.8**	**77.8**
breast-w	95.7	96.5	97.2	98.0	98.6	98.0	**98.8**
car	93.6	96.2	96.0	96.8	96.8	97.2	**97.4**
chess	91.5	94.1	95.2	96.2	96.2	97.1	**97.8**
contraceptive	58.2	64.6	64.6	66.2	68.5	**70.2**	**70.2**
credit-a	81.1	84.5	85.8	87.3	85.8	88.1	**88.3**
credit-g	82.3	85.1	88.7	92.1	90.9	**93.5**	93.1
dermatology	97.8	97.4	97.4	98.0	97.4	98.0	**98.2**
ecoli	66.2	66.2	66.7	65.4	**66.8**	66.7	66.7
glass	71.1	72.0	72.0	72.8	70.5	73.1	**73.8**
hayes-roth	84.0	84.3	85.2	84.0	85.5	87.6	**87.8**
heart-c	63.3	73.4	76.9	75.8	78.3	82.1	**82.5**
heart-s	80.3	85.6	85.6	90.4	91.9	96.2	**97.5**
ionosphere	90.7	93.6	95.1	94.2	95.5	**97.0**	**97.0**
iris	90.2	94.2	**96.4**	**96.4**	**96.4**	**96.4**	**96.4**
lung-c	68.4	93.2	94.4	94.2	94.2	95.0	**95.6**
monk	66.1	69.3	71.9	73.9	75.7	80.8	**81.2**
mushrooms	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**	**98.8**
nursery	94.2	95.1	96.0	95.6	96.0	97.2	**97.5**
parkinsons	91.7	92.9	94.6	94.8	94.8	96.0	**96.2**
page block	92.3	94.5	94.5	94.5	92.9	**95.8**	**95.8**
post-operative	71.1	77.4	79.2	77.9	78.7	84.7	**85.2**
segmentation	93.4	96.4	96.1	95.9	96.4	96.9	**97.2**
soybean	53.5	64.6	67.8	68.6	68.6	70.8	**72.1**
SPECT	76.2	88.2	**90.5**	89.9	89.5	**90.5**	**90.5**
tic-tac-to	76.6	94.5	94.5	94.8	94.5	**97.5**	**97.5**
vote	92.1	94.7	95.2	95.2	94.3	98.1	**98.5**
wine	97.3	94.2	95.8	94.2	**97.8**	97.5	**97.8**
yeast	61.2	67.4	69.1	67.1	**69.9**	**69.9**	**69.9**
Avg. Rank	7.3	5.9	4.6	4.5	4	2.1	1.5

difference of predictive accuracy between the two algorithms being compared is not statistically significant at the 10% level.

10.3. Summary of Results

We summarize the average ranking results for the clustering-based BMN learning algorithm across the 8 experimental setups in Table 13. The last row of the table is the average of average rankings obtained for each algorithm across the 8 experimental setups – i.e. considering all combinations of 4 different numbers of clusters and 2 different local BN classifiers (Naïve-Bayes and TAN).

According to this summary, as well as the previous result tables, we can conclude the following:

- **K-Clust vs. K-Modes** – The extended K-Clust algorithm outperforms the conventional K-Modes clustering algorithm in finding good clusters for building BMN classifiers. This is shown in the average rankings obtained by K-ClustB (with overall average value of 5) compared to K-ModesB (with overall average value of 6.3).
- **ACO vs. Conventional Algorithms** – The 4 ACO meta-heuristic algorithms used for building clustering-based BMN classifiers outperform the 2 conventional clustering algorithms. This is shown in the average rankings obtained by the ACO-based algorithm compared to both K-ModesB and K-ClustB.
- **Cluster-then-learn vs. Cluster-with-learn** – The integrated approach of ACO clustering with BMN learning outperforms the sequential approach of ACO clustering then BMN learning. This is observed with both clustering solution representations, since the cluster-with-learn ACO-ClustB$_{IB}$ (with overall average ranking 2.1) outperforms the cluster-then-learn Ant-ClustB$_{IB}$ (with overall average ranking 4.4); and ACO-ClustB$_{MB}$ (with overall average ranking 1.7) outperforms Ant-ClustB$_{MB}$ (with overall average ranking 4).
- **Medoid-based vs. Instance-based Solution Representations** – The ACO medoid-based (MB) clustering solution representation outperforms the instance-based (IB) representation, which can be observed in both ACO learning approaches. Ant-ClustB$_{MB}$ (with overall average ranking of 4.0) outperforms Ant-ClustB$_{IB}$ (with overall average ranking of 4.4), and ACO-ClustB$_{MB}$ (with overall average ranking of 1.7) outperforms ACO-ClustB$_{IB}$ (with overall average ranking of 2.1).

As for the execution time, in general, the Medoid-based (MB) method in the ACO clustering solution representation takes longer execution time than the Instance-based (IB) IB method, since the latter converges faster than the former and does not utilize the maximum number of iterations. Besides, there is no obvious difference in the execution time between ACO "cluster-then-lear" and the "cluster-with-learn" ACO approaches.

Table 12: Results (Statistical Significance levels) of the non-parametric Friedman test

				Cluster-Then-Learn		Cluster-With-Learn	
Local BN	K	Algorithm	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
Naïve	2	Naïve	0.05	0.05	0.05	0.05	0.05
		K-Modes	0.05	0.05	0.05	0.05	0.05
		K-Clust	–	–	0.05	0.05	0.05
		TAN-MN	–		–	–	–
		Ant_{IB}	–	–	–	0.1	0.1
		Ant_{MB}	–	–	–	–	–
Naïve	4	Naïve	0.05	0.05	0.05	0.05	0.05
		K-Modes	–	0.05	0.05	0.05	0.05
		K-Clust	–	0.1	0.05	0.05	0.05
		TAN-MN	–	–	0.1	0.1	0.1
		Ant_{IB}	–		–	0.1	0.1
		Ant_{MB}	–	–	–	0.1	0.1
Naïve	6	Naïve	0.05	0.05	0.05	0.05	0.05
		K-Modes	–	0.05	0.05	0.05	0.05
		K-Clust	–	0.05	–	0.05	0.05
		TAN-MN	–	0.1	–	0.05	0.05
		Ant_{IB}	–	–	–	0.5	0.05
		Ant_{MB}	–	–	–	0.1	0.05
Naïve	8	Naïve	0.05	0.05	0.05	0.05	0.05
		K-Modes	0.1	0.05	0.05	0.05	0.05
		K-Clust	–	0.1	0.1	0.1	0.05
		TAN-MN	–	–	0.1	0.05	0.05
		Ant_{IB}	–	–	–	0.05	0.05
		Ant_{MB}	–	–	–	0.05	0.05
TAN	2	TAN	0.05	0.05	0.05	0.05	0.05
		K-Modes	–	0.05	0.05	0.05	0.05
		K-Clust	–	0.05	0.1	0.05	0.05
		TAN-MN	–	–	–	0.05	0.05
		Ant_{IB}	–	–	–	0.1	0.05
		Ant_{MB}	–	–	–	–	0.1
TAN	4	TAN	0.05	0.05	0.05	0.05	0.05
		K-Modes	–	0.05	0.05	0.05	0.05
		K-Clust	–	0.05	0.05	0.05	0.05
		TAN-MN	–	0.1	0.1	0.05	0.05
		Ant_{IB}	–	–	–	0.1	0.1
		Ant_{MB}	–	–	–	–	–
TAN	6	TAN	0.05	0.05	0.05	0.05	0.05
		K-Modes	0.1	0.1	0.1	0.5	0.05
		K-Clust	–	0.1	–	0.1	0.05
		TAN-MN	–	0.1	0.1	0.05	0.05
		Ant_{IB}	–	–	–	0.1	0.05
		Ant_{MB}	–	–	–	0.05	0.05
TAN	8	TAN	0.05	0.05	0.05	0.05	0.05
		K-Modes	–	0.1	0.1	0.05	0.05
		K-Clust	–	–	0.1	0.1	0.05
		TAN-MN	–	0.1	0.1	0.1	0.1
		Ant_{IB}	–	–	0.1	0.05	0.05
		Ant_{MB}	–	–	–	0.05	0.05

Table 13: Average Rankings Summary for the Clustering-based BMN Learning Algorithms

				Cluster-Then-Learn		Cluster-With-Learn	
Local BN	K	K-Modes	K-Clust	Ant_{IB}	Ant_{MB}	ACO_{IB}	ACO_{MB}
Naïve	2	6.8	5.6	4.9	4.3	2.2	2.0
	4	6.6	5.2	4.8	4.0	1.8	1.5
	6	6.6	5.2	4.1	4.2	2.2	1.6
	8	6.2	5.2	4.3	4.0	2.0	1.6
TAN	2	6.0	4.9	3.7	4.1	2.1	2.0
	4	6.1	4.7	4.4	3.6	2.2	1.6
	6	6.3	4.8	4.6	3.5	2.1	1.7
	8	5.9	4.6	4.5	4.0	2.1	1.5
Average Rank		6.3	5	4.4	4	2.1	1.7

11. Concluding Remarks

In this paper, we have proposed several contributions to the area of clustering-based BMN learning, which can be summarized as follows. We introduced a new medoid-based clustering solution representation for ACO clustering in the Ant-ClustB$_{MB}$ algorithm, in order to learn more effective clusters to be used to build BMNs. We proposed a novel "cluster-with-learn" approach, in which the ACO meta-heuristic performs the clustering and the BMN learning in a synergistic fashion, and produced two new ACO algorithms based on that approach: ACO-ClustB$_{IB}$, using the instance-based representation, and ACO-ClustB$_{MB}$, using the medoid-based representation.

Empirical results were obtained from experiments on 30 UCI datasets, using four different numbers of clusters and two types of local BN classifiers. The experiments' results showed that: 1) our extensions enhance the performance of the clustering algorithm in the sense of producing clusters that lead to BMN classifiers with higher predictive accuracies; 2) the new integrated ACO "cluster-with-learn" approach outperforms the sequential ACO "cluster-then-learn" approach; and 3) the proposed medoid-based representation is more effective than the instance-based representation as a clustering solution representation for ACO algorithms.

As a future work, we would like to try utilizing different types of optimization meta-heuristics, such as genetic algorithms (GAs) and paritcle swarm optimization (PSO), for learning the cluster-based BMN classifiers, and compare them to the use of ACO in our algorithms. Besides, it will be interesting to calculate the time complexity of the introduced algorithms. In addition, we would like to try some techniques to remove outliers from the clusters, as in the CBBN algorithm [24], and build a generic model besides the local models that would host all the outliers.

References

[1] G. F. Cooper, E. Herskovits, A Bayesian Method for the Induction of Probabilistic Networks from Data, Machine Learning 9 (4) (1992) 309–347.

[2] D. Heckerman, D. Geiger, D. M. Chickering, Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, Machine Learning 20 (3) (1995) 197–243.

[3] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann, San Francisco, CA, USA, 2010.

[4] J. Han, M. Kamber, Data Mining: Concepts and Techniques, 2nd Edition, Morgan Kaufmann, San Francisco, CA, USA, 2000.

[5] J. Cheng, R. Greiner, Comparing Bayesian Network Classifiers, in: 15th Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, USA, 1999, pp. 101–108.

[6] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, P. Smyth, Bayesian Network Classifiers, Machine Learning 29 (1997) 131–163.

[7] J. Cheng, R. Greiner, Learning Bayesian Belief Network Classifiers: Algorithms and System, in: 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, Springer, London, UK, 2001, pp. 141–151.

[8] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.

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

[10] R. S. Parpinelli, H. S. Lopes, A. A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 321–332.

[11] K. M. Salama, A. M. Abdelbar, A. A. Freitas, Multiple Pheromone Types and Other Extensions to the Ant-Miner Classification Rule Discovery Algorithm., Swarm Intelligence 5 (3-4) (2011) 149–182.

[12] K. M. Salama, A. M. Abdelbar, Exploring Different Rule Quality Evaluation Functions in ACO-based Classification Algorithms, in: IEEE Swarm Intelligence Symposium, IEEE Press, Piscataway, NJ, USA, 2011, pp. 1–8.

[13] M. Jafar, R.Sivakumar, Ant-based Clustering Algorithms: A Brief Survey, International Journal of Computer Theory and Engineering 2 (2010) 787–796.

[14] P. S. Shelokar, V. K. Jayaraman, B. D. Kulkarni, An ant colony approach for clustering, Analytica Chimica Acta 509 (2) (2004) 187–195.

[15] X. yong Liu, H. Fu, An Effective Clustering Algorithm With Ant Colony, Journal of Computers 5 (2010) 598–605.

[16] Y. Wu, J. McCall, D. Corne, Two Novel Ant Colony Optimization Approaches for Bayesian Network Structure Learning, in: IEEE Congress on Evolutionary Computation (CEC), IEEE Press, New York, NY, USA, 2010, pp. 1–7.

[17] L. M. de Campos, J. M. Fernandez-Luna, J. A. Gamez, J. M. Puerta, Ant Colony Optimization for Learning Bayesian Networks, International Journal of Approximate Reasoning 31 (3) (2002) 291–311.

[18] R. Daly, Q. Shen, Learning Bayesian Network Equivalence Classes with Ant Colony Optimization, Journal of Artificial Intelligence Research (JAIR) 35 (2009) 391–447.

[19] P. C. Pinto, A. Nägele, M. Dejori, T. A. Runkler, Ao, Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning, IEEE Transactions on Evolutionary Computation 13 (4) (2009) 767–779.

[20] K. M. Salama, A. A. Freitas, ABC-Miner: an Ant-based Bayesian Classification Algorithm, in: 8th International Conference on Swarm Intelligence (ANTS'12), LNCS 7461, Springer, Berlin, 2012, pp. 13–24.

[21] K. M. Salama, A. A. Freitas, Learning Bayesian Network Classifiers Using Ant Colony Optimization, Swarm Intelligence 7 (2-3) (2013) 229–254.

[22] K. M. Salama, A. A. Freitas, Extending the ABC-Miner Bayesian Classification Algorithm, in: 6th International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO'13), Vol. 512 of Studies in Computational Intelligence, Springer, Berlin, 2013, pp. 1–12.

[23] K. M. Salama, A. A. Freitas, Clustering-based Bayesian Multi-net Classifier Construction with Ant Colony Optimization, in: IEEE Congress on Evolutionary Computation (IEEE CEC) (2013), IEEE Press, New York, NY, USA, 2013, pp. 3079–3086.

[24] E. S. Jr., A. Hussein, Case-Based Bayesian Network Classifiers, in: 17th International FLAIRS Conference, AAAI, Vol. 5, AAAI Press, Stanford, USA, 2004, pp. 598–605.

[25] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, 2nd Edition, Addison Wesley, 2005.

[26] R. Daly, Q. Shen, S. Aitken, Review: Learning Bayesian Networks: Approaches and Issues, Knowledge Engineering Reviews 26 (2) (2011) 99–157.

[27] D. Heckerman, Studies in Computational Intelligence: Innovations in Bayesian Networks., Vol. 156, Springer, Berlin, 2008, Ch. 3: A Tutorial on Learning with Bayesian Networks., pp. 33–82.

[28] L. Jiang, D. Wang, Z. Cai, X. Yan, Survey of Improving Naive Bayes for Classification, in: 3rd International Conference on Advanced Data Mining and Applications (ADMA'07), no. 4632 in LNCS, Springer, Berlin, 2007, pp. 134–145.

[29] D. Geiger, D. Heckerman, Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets, Artificial Intelligence 82 (2) (1996) 45–74.

[30] M. Dorigo, T. Stützle, The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Vol. 57, Springer, New York, NY, USA, 2003.

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

[32] F. Otero, A. Freitas, C. Johnson, Inducing Decision Trees with an Ant Colony Optimization Algorithm, Applied Soft Computing 12 (11) (2012) 3615–3626.

[33] U. Boryczka, J. Kozak, Ant Colony Decision Trees, in: 4th International Conference on Computational Collective Intelligence: Technologies and Applications (ICCCI'11), Springer, Berlin, 2010, pp. 4373–382.

[34] W. Buntine, Theory Refinement on Bayesian Networks, in: 17th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, USA, 1991, pp. 52–60.

[35] K. Huang, I. King, M. Lyu, Discriminative Training of Bayesian Chow-Liu Multinet Classifiers, in: International Joint Conference on Networks, Vol. 1, IEEE Press, New York, NY, USA, 2003, pp. 484–488.

[36] Y. Gurwicz, B. Lerner, Bayesian Class-Matched Multinet Classifier, in: International Conference on Structural, Syntactic, and Statistical Pattern Recognition (IAPR'6), Springer, Berlin, 2006, pp. 145–153.

[37] P. Langley, Induction of Recursive Bayesian Classifiers, in: European Conference on Machine Learning (ECML), Springer, Berlin, 1993, pp. 153–164.

[38] R. Kohavi, Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid, in: 2nd International Conference on Knwoledge Discovery and Data Mining, AAAI Press, Stanford, USA, 1996, pp. 202–207.

[39] J. M. Peña, J. A. Lozano, P. Larrañaga, Learning recursive bayesian multinets for data clustering by means of constructive induction, Machine Learning 47 (1) (2002) 63–89.

[40] A. Cano, J. G. Castellano, A. R. Masegosa, S. Moral, Methods to determine the branching attribute in bayesian multinets classifiers, in: 8th European conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer-Verlag, 2005, pp. 932–943.

[41] E. Santos, A. Hussein, Comparing case-based bayesian network and recursive bayesian multi-net classifiers, in: International Conference on Artificial Intelligence (ICAI), 2004, pp. 627–633.

[42] C. Stanfill, D. Waltz, Toward memory-based reasoning, Communications of the ACM 29 (1986) 1213–1228.

[43] K. M. Salama, A. A. Freitas, Investigating the Impact of Various Classification Quality Measures in the Predictive Accuracy of ABC-Miner, in: IEEE Congress on Evolutionary Computation (IEEE CEC) (2013), IEEE Press, New York, NY, USA, 2013, pp. 2677–2694.

[44] A. Asuncion, D. Newman, UCI Machine Learning Repository. URL:http://www.ics.uci.edu/ mlearn/MLRepository.html.

[45] J. Demsar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 1 (7) (2006) 1–30.

[46] S. Garca, F. Herrera, An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons, Journal of Machine Learning Research 9 (2008) 2321–2328.