ABC-Miner: an Ant-based Bayesian Classification Algorithm

Khalid M. Salama and Alex A. Freitas

School of Computing, University of Kent,
Canterbury, CT2 7NF, UK
kms39@kent.ac.uk, A.A.Freitas@kent.ac.uk

Abstract. Bayesian networks (BNs) are powerful tools for knowledge representation and inference that encode (in)dependencies among random variables. A Bayesian network classifier is a special kind of these networks that aims to compute the posterior probability of each class given an instance of the attributes and predicts the class with the highest posterior probability. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-hard, heuristic search algorithms need to be applied effectively to build high-quality networks. In this paper, we propose a novel algorithm, called ABC-Miner, for learning the structure of BN classifiers using the Ant Colony Optimization (ACO) meta-heuristic. We describe all the elements necessary to tackle our learning problem using ACO, and experimentally compare the performance of our ant-based Bayesian classification algorithm with other algorithms for learning BN classifiers used in the literature.

Keywords: Ant Colony Optimization (ACO), Data Mining, Classification, Bayesian Network Classifiers.

1 Introduction

Classification is a data mining task where the goal is to build, from labeled cases, a model (classifier) that can be used to predict the class of unlabeled cases. Learning classifiers from datasets is a central problem in data mining and machine learning research fields. While different approaches for tackling this problem exist, such as decision trees, artificial neural networks and rule list [20], our focus in this paper is on the Bayesian approach for classification.

Naïve-Bayes is the first Bayesian classifier in the literature. Although it is a very simple kind of Bayesian networks that assumes the attributes are independent given the class label, Naïve-Bayes classifiers showed effective predictive performance under the aforementioned assumption [10]. However, since the independency assumption amongst the dataset attributes is not realistic, extended versions were developed to improve the performance of Naïve-Bayes, namely Tree Augmented Naïve-Bayes (TANs), Bayesian networks Augmented Naïve-Bayes (BANs) and General Bayesian Networks (GBNs) [10]. These algorithms consider dependencies between the attributes in the learning process to

build more complex and hopefully more accurate BN classifiers. Nonetheless, algorithms used in the literature for building such BNs utilize greedy and deterministic techniques. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-hard [5], several stochastic search algorithms can be effectively applied to build high-quality BN classifiers in an acceptable computational time.

Ant Colony Optimization (ACO) [9] is a meta-heuristic for solving combinatorial optimization problems, inspired by observations of the behavior of ant colonies in nature. ACO has been successful in solving several problems, including classification rule induction [13–15, 18] and general purpose BN construction [2, 8, 17, 21]. However, as far as we know, it has not been used for learning Bayesian network classifiers.

In this paper, we propose a novel ant-based Bayesian classification algorithm, called ABC-Miner, which learns the structure of a BAN with at most k-dependencies from a dataset using ACO technique for optimization. The rest of the paper is organized as follows. In Section 2 a brief overview on Bayesian networks' basic concepts is given as well as a discussion of various Bayesian network classifiers is shown. Section 3 exhibits the related work on the use of ACO algorithms for building BNs in the literature. In Section 4, we introduce our proposed ABC-Miner algorithm and describe each of the elements necessary to tackle our learning problem using the ACO meta-heuristics. After that, section 5 discusses our experimental methodology and results. Finally, we conclude with some general remarks and provide directions for future research in section 6.

2 Bayesian Networks Background

2.1 Overview on Bayesian Networks

Bayesian networks are knowledge representation tools that aim to model dependence and independence relationships amongst random variables [12]. In essence, BNs are used to describe the joint probability distribution of n random variables $\mathbf{X} = \{X_1, X_2, X_3, ..., X_n\}$. A directed acyclic graph (DAG) is used to represent the variables as nodes and statistical dependencies between the variables as edges between the nodes – child nodes (variables) depend on their parent ones. In addition, a set of conditional probability tables (CPTs), one for each variable, is obtained to represent the parameters Θ of the network. The graphical structure of the network along with its parameters specifies a joint probability distribution over the set of variables $\mathbf{X}$ that is formulated in the product form:

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i | \mathbf{Pa}(X_i), \Theta, G) \tag{1}$$

where $\mathbf{Pa}(X_i)$ are the parents of variable X_i in G (the DAG that represents the structure of the BN).

Learning a Bayesian network from a dataset $\mathbf{D}$ with $\{d^1, d,^2, ..., d^m\}$ instances is decomposed into two phases; learning the network structure, and then learning the parameters of the network. As for parameter learning, it is

considered a straightforward process for any given BN structure with specified (in)dependencies between variables. Simply, a conditional probability table (CPT) is computed for each variable with respect to its parent variables. CPT of variable X_i encodes the likelihood of this variable given its parents $\mathbf{Pa}(X_i)$ in the network graph G, and the marginal likelihood of the dataset $\mathbf{D}$ given a structure G is denoted by $P(\mathbf{D}|G)$. The purpose is to find G that maximizes $P(\mathbf{D}|G)$ for a given $\mathbf{D}$, which is the role of BN structure learning phase. The common approach to this problem is to introduce a scoring function, f, that evaluates each G with respect to $\mathbf{D}$, searching for the best network structure according to f. Various scoring metrics are usable for this job [6, 12].

A well-known greedy approach for building BN structure is Algorithm B [1]. It starts with an empty DAG (edge-less structure) and at each step it adds the edge with the maximum increase in the scoring metric f, whilst avoiding the inclusion of directed cycles in the graph. The algorithm stops when adding any valid edge does not increase the value of the scoring metric. K2, a metric based on uniform prior scoring, is one of the most used scoring metrics for building and evaluating Bayesian networks [6].

For further information about Bayesian networks, the reader is referred to [11, 12], which provide a detailed discussion of the subject.

2.2 Bayesian Networks for Classification

Bayesian network classifiers are a special kind of BNs where the class attribute is treated as a unique variable in the network. The purpose is to compute the probability of each value of the class variable given an instance of the predictor attributes and assign this instance to the class that has the highest posterior probability value. The following are various types of BN classifiers studied in the literature.

- **Naïve-Bayes**: The classifier consists of a simple BN structure that has the class node as the only parent node of all other nodes. This structure assumes that all attributes are independent of each other given the class. In spite of its simplicity, Naïve-Bayes has surprisingly outperformed many sophisticated classifiers over a large number of datasets, especially where the attributes are not strongly correlated [10].
- **Tree Augmented Naïve-Bayes (TAN)**: As an extension to Naïve-Bayes, TAN allows a node in a BN to have more than one parent, besides the class variable. This produces a tree-like structure BN. A variation of the Chow-Liu algorithm [3] is the best known method for building TANs. First, it computes the conditional mutual information $I(X, Y|\mathbf{C})$ between each pair of variables X and Y given class variable $\mathbf{C}$. Then it builds a complete undirected graph connecting all the input variables to find the maximum weighted spanning tree from the graph, where the weight of edge $X \rightarrow Y$ is annotated with $I(X, Y|\mathbf{C})$. After that, it chooses a root variable and sets the direction of all edges to be outwards of it. Finally, it adds one edge from the class node to each of the other variables, building a TAN classifier.

- **BN Augmented Naïve-Bayes (BAN)**: It is an elaborated version of Naïve-Bayes, in which no restrictions (or at most k-dependencies) are enforced on the number of the parents that a node in the network can depend on. In other words, while each node in TAN can have only one parent besides the class node, and in Naïve-Bayes only the class node is allowed to be the parent, each node in BAN can have k of parents (dependencies) besides the class node. Another variation of the Chow-Liu algorithm that is used to build TANs, is utilized to BANs as well [4].
- **General Bayesian Network (GBN)**: Unlike the other BN classifier learners, the GBN treats the class variable node as an ordinary node. The idea is to build a general purpose Bayesian network, find the *Markov blanket* of the class node, delete all the other nodes outside it and use the resulting network as a Bayesian classifier. One *Markov blanket* of a node n is the union of the n's parents, n's children, and the parents of n's children.

Friedman et al. provided an excellent study of these algorithms in [10]. A comprehensive investigation and comparisons of these various Bayesian classifiers by Cheng and Greiner are found in [3, 4] .

3 ACO Related Work

Ant Colony Optimization has an effective contribution in tackling the classification problem. Ant-Miner [15] is the first ant-based classification algorithm. Several extensions on this algorithm have been introduced in the literature, such as AntMiner+ [13], cAnt-Miner [14], and multi-pheromone Ant-Miner [18]. However, the Ant-Miner algorithm as well as its various versions handles the classification problem by building a list of <**IF**_*Antecedent*_**THEN**_*Class*> classification rules. On the other hand, this paper proposes a new ant-based algorithm that handles classification problems, yet with a different approach; learning a Bayesian network to be used as classifier.

As for the use of ACO for building Bayesian networks, to date, there has been only a few research utilizing such a heuristic in learning BN structure, namely: ACO-B [2], MMACO [16, 17], ACO-E [7, 8] and CHAINACO - K2ACO [21]. Moreover, none of them has been used for building BN classifiers. As far as we know, our proposed ABC-Miner is the first algorithm to use ACO, or any evolutionary algorithm, in the task of learning Bayesian networks specific for the classification problem.

Campos et al. introduced the first ant-based algorithm for learning Bayesian networks, ACO-B [2], where each ant iteratively constructs a complete Bayesian network from scratch by selecting edges to be added to the network and updates the pheromone on the construction graph according to the quality of the constructed BN. Edge selection is carried out in stochastic fashion, according to the pheromone and the heuristic values associated with the edge. The heuristic function used is the same function used for evaluating the quality of the BN, which is the K2 scoring metric [6].

Pinto et al. used a different local discovery approach for learning BNs in [16, 17]. This is hybrid approach, MMACO, based on the local discovery algorithm Max-Min Parents and Children (MMPC) and ant colony optimization (ACO). MMPC is used to construct the skeleton of the Bayesian network and then ACO is used to orientate its edges, thus returning the final structure. Here all the ants are involved in building a single solution by testing several possible edge additions and orientation at the same iteration. BDEu [12] is the function used by MMACO to calculate the heuristics and evaluates the BN quality.

Daly et al. studied learning the structure of a Bayesian network by performing a search through the space of its equivalence classes via extending traditional ACO-based algorithm, ACO-E [7, 8]. An equivalence class includes all network structures where changing the orientation (dependency relationship) of one or more edges in a BN obtains the same quality according to a given scoring metric. In which case, not all the edges in an equivalence class of a BN are oriented, since the direction of the dependencies of some edge does not change the quality of the network.

Yanghui et al. proposed two novel ACO approaches for Bayesian network structure learning, CHAINACO and K2ACO [21]. The former is based on a GA algorithm. It consists of two phases; constructing chains (the order of nodes according to dependencies) using ACO instead of GA, then applies K2 to the best ordering found and returns the best structure. K2ACO is also based on another algorithm, K2GA, which only consists of a single phase. The quality of each node ordering chosen by an ant is evaluated by running the K2 search algorithm to construct a BN calculating the score of the network structure found. The best structure returned is that generated by K2 from the best ordering evaluated in this fashion.

Note that the goal of the aforementioned algorithms is to build general purposes BNs. In other words, the selection of the heuristics, quality evaluation metric and other elements of the algorithm are suitable for this aim, but not for building BN classifiers. Hence, in spite of having some similarities, essential aspects of our algorithm are different due to the diversion in the target; our algorithm is only focused on learning BN classifiers. Next we will explore these aspects as we describe our novel Ant-based Bayesian Classifier.

4 A Novel ACO Algorithm for Learning BN Classifiers

4.1 ABC-Miner Algorithm

The overall process of ABC-Miner is illustrated in Algorithm 1. The core element of any ACO-based algorithm is the construction graph that contains the decision components in the search space, with which an ant constructs a candidate solution. As for the problem at hands, the decision components are all the edges $X \rightarrow Y$ where $X \neq Y$ and X, Y belongs to the input attributes of a given training set. These edges represent the variable dependencies in the resulting Bayesian network classifier.

At the beginning of the algorithm, the pheromone amount is initialized for each decision component with the same value. The initial amount of pheromone on each edge is $1/|TotalEdges|$. In addition, the heuristic value for each edge $X \rightarrow Y$ is set using the conditional mutual information, which is computed as follows:

$$I(X, Y|\mathbf{C}) = \sum_{c \in C} p(c) \sum_{x \in X} \sum_{y \in Y} p(x, y|c) \log \frac{p(x, y|c)}{p(x|c)p(y|c)} \tag{2}$$

where $\mathbf{C}$ is the class variable. $p(x, y|c)$ is the conditional probability of value $x \in X$ and $y \in Y$ given class value c, $p(x|c)$ is the conditional probability of x given c, $p(y|c)$ is the conditional probability of y given c and $p(c)$ is the prior probability of value c in the class variable. Conditional mutual information is a measure of correlation between two random variables given a third one. In our case, we want to lead the ant during the search process to the edges between correlated variables given the class variable, and so we use such a function as heuristic information associated with the selectable edges. Note that the procedure of heuristic calculation is called only once at the beginning and its calculations used throughout the algorithm.

Algorithm 1 Pseudo-code of ABC-Miner.

```
Begin ABC-Miner
BNC_gbest = φ; Q_gbest = 0
InitializePheromoneAmounts();
InitializeHeuristicValues();
t = 0;
repeat
    BNC_tbest = φ; Q_tbest = 0;
    for i = 0 → colony_size do
        BNC_i = CreateSolution(ant_i);
        Q_i = ComputeQuality(BNC_i);
        if Q_i > Q_tbest then
            BNC_tbest = BNC_i;
            Q_i = Q_tbest;
        end if
    end for
    PerformLocalSearch(BNC_tbest);
    UpdatePheromone(BNC_tbest);
    if Q_tbest > Q_gbest then
        BNC_gbest = BNC_tbest;
    end if
    t = t + 1;
until t = max_itrations or Convergence()
return BNC_gbest;
End
```

The outline of the algorithm is as follows. In essence, each ant_i in the colony creates a candidate solution BNC_i, i. e. a Bayesian network classifier. Then the quality of the constructed solution is evaluated. The best solution BNC_{tbest} produced in the colony is selected to undergo local search before the ant updates the pheromone trail according to the quality of its solution Q_{tbest}. After that, we compare the iteration best solution BNC_{tbest} with the global best solution BNC_{gbest} to keep track of the best solution found so far. This set of steps is considered an iteration of the $repeat - until$ loop and is repeated until the same solution is generated for a number of consecutive trials specified by the `conv_iterations` parameter (indicating convergence) or until `max_iterations` is reached. The values of `conv_iterations`, `max_iterations` and `colony_size` are user-specified thresholds. In our experiments (see section 5), we used 10, 500 and 5 for each of these parameters respectively.

4.2 Solution Creation

Instead of having the user selecting the optimum maximum number of dependencies that a variable in the BN can have (at most k parents for each node), this selection is carried out by the ants in ABC-Miner. Prior to solution creation, the ant selects the maximum number of dependencies (k) as a criterion for the currently constructed BN classifier. This selection of k value is done probabilistically from a list of available numbers. The user only specifies `max_parents` parameter (that we set to 3 in our experiments), and all the integer values from 1 to this parameter are available for the ant to use in the BN classifier construction. The various values of the k are treated as decision components as well. More precisely, the ant updates the pheromone on the value k of the maximum number of parents after solution creation according to the quality of this solution, which used value k as a criterion in the BN classifier construction. This pheromone amount represents the selection probability of this value by subsequent ants, leading to convergence on an optimal value of k dependencies. Algorithm 2 shows the outline of the solution creation procedure.

Algorithm 2 Pseudo-code of Solution Creation Procedure.

```
Begin CreateSolution()
BNC_i ← {Naïve-Bayes structure};
k = ant_i.SelectMaxParents();
while GetValidEdges() <> φ do
      {i → j} = ant_i.SelectEdgeProbablistically();
      BNC_i = BNC_i ∪ {i → j};
      RemoveInvalidEdges(BNC_i, k);
end while
BNC_i.LearnParameters();
return BNC_i;
End
```

Each ant starts with the network structure of the Naïve-Bayes classifier, i. e. a BN in which all the variables have only the class variable as a parent. From that point, it starts to expand this Naïve-Bayes network into a Bayesian Augmented Naïve-Bayes (BAN) by adding edges to the network. The selection of the edges is performed according to the following probabilistic state transition formula:

$$P_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{a}^{I} \sum_{b}^{J} [\tau_{ab}(t)]^{\alpha} \cdot [\eta_{ab}]^{\beta}} \tag{3}$$

In this equation, P_{ij} is the probability of selecting the edge $i \rightarrow j$, $\tau_{ij}(t)$ is the amount of pheromone associated with edge $i \rightarrow j$ at iteration t and η_{ij} is the heuristic information for edge $i \rightarrow j$ computed using conditional mutual information (equation 2). The edge $a \rightarrow b$ represents a valid selection in the available edges. The exponents α and β are used to adjust the relative emphases of the pheromone (τ) and heuristic information (η), respectively. Note that edges available for selection are directed, i. e. $i \rightarrow j \neq j \rightarrow i$.

ABC-Miner adapts the "ants with personality" approach, proposed by the author in [18]. Each ant_i is allowed to have its own *personality* by allowing it to have its own values of the α_i and β_i parameters. In other words, some ants will give more importance to pheromone amount, while others will give more importance to heuristic information. The α_i and β_i parameters are each independently drawn from a Gaussian distribution centered at 2 with a standard deviation of 1. This approach aims to advance exploration and improve search diversity in the colony.

An edge $i \rightarrow j$ is valid to be added in the BN classifier being constructed if the following two criteria are justified: 1) its inclusion does not create a directed cycle, 2) the limit of k parents (chosen by the current ant) for the child variable j is not violated by the inclusion of the edge. After the ant adds a valid edge to the BN classifier, all the invalid edges are eliminated from the construction graph. The ant keeps adding edges to the current solution until no valid edges are available. When the structure of BNC_i is finished, the parameters Θ are learnt by calculating the CPT for each variable, according to the network structure, producing a complete solution. Afterward, the quality of the BN classifier is evaluated, and all the edges become available again for the next ant to construct another candidate solution.

4.3 Quality Evaluation and Pheromone Update

Unlike the traditional Bayesian networks, the target of our algorithm is to build an effective BN in terms of predictive power with respect to a specific class attribute. In other words, BN learning algorithms aim to maximize a scoring function that seeks a structure that best represents the dependencies between all the attributes of a given dataset. This structure should fit the knowledge representation and inference purposes of a BN, which treats all the variables in the same way, without distinguishing between the predictor and the class attributes. On the other hand, the purpose of learning a BN classifier is to build

a structure that can calculate the probability of a class value given an instance of the input predictor variables, and predict the class value with the highest probability to label the instance.

Therefore, using traditional scoring functions to evaluate the quality of a BN classifier should not fit the purpose of building a classifier [10]. According to this reasoning, we evaluate the quality of the constructed network directly as a classifier, where the predictive efficiency is the main concern. We use the *accuracy*, a conventional measure of predictive performance, to evaluate the constructed BN model, computed as follows:

$$Accuracy = \frac{|Correctly\_Classified\_Cases|}{|Validation\_Set|} \tag{4}$$

The best BN classifier BNC_{tbest} constructed amongst the ants in the colony undergoes local search, which aims to improve the predictive accuracy of the classifier. The local search operates as follows. It temporarily removes one edge at a time in a reverse order (removing last the edge that was added to the network first). If the quality of the BN classifier improves, this edge is removed permanently from the network, otherwise it is added once again. Then we proceed to the next edge. This procedure continues until all the edges are tested to be removed from the BN classifier and the BN classifier with the highest quality – with respect to classification accuracy – is obtained.

After BNC_{tbest} is optimized via local search, pheromone levels are increased on decision components (edges) in the construction graph included in the structure of the constructed BN classifier, using the following formula:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t).Q_{tbest}(t) \tag{5}$$

To simulate pheromone evaporation, normalization is then applied as in [15]; each τ_{ij} is divided over the total pheromone amounts in the construction graph. Note that pheromone update is carried out for the decision components representing the number of dependencies used for building the BN classifier structure as well.

5 Experimental Methodology and Results

The performance of ABC-Miner was evaluated using 15 public-domain datasets from the UCI (University of California at Irvine) dataset repository [19]. Datasets containing continuous attributes were discretized in a pre-possessing step, using the C4.5-Disc [20] algorithm. The main characteristics of the datasets are shown in Table 1. We compare the predictive accuracy of our proposed ant-based algorithm with three other widely used algorithms for learning Bayesian classifiers. In our experiment, we used Weka [20] implementations for these algorithms. Table 2 presents the main characteristics of the used algorithms.

The experiments were carried out using 10-fold cross validation procedure. In essence, a dataset is divided into 10 mutually exclusive partitions, were each time a different partition is used as the test set and the other 9 partitions are used

Table 1. Description of Datasets Used in Experimental Results

Dataset	Size	Attributes	Classes
balance scale	625	4	3
breast cancer (wisconsin)	286	9	2
car evaluation	1,728	6	4
contraceptive method choice	1,473	9	3
statlog credit (australian)	690	14	2
statlog credit (german)	1,000	20	2
dermatology	366	33	6
hayes-roth	160	4	3
heart (cleveland)	303	12	3
iris	150	4	3
monks	432	6	2
nursey	12,960	8	5
soybean	307	35	19
tic-tac-to	958	9	2
voting records	435	16	2

Table 2. Summary of the BN Classifier Learning Algorithms Used in the Experiments

Algorithm	Type	Search Strategy	Optimization
Naïve-Bayes	Deterministic	-	-
TAN	Deterministic	Finding Max. Spanning Tree	Cond. Mutual Info.
GBN	Deterministic	Greedy Hill Climbing	K2 Function
ABC-Miner	Stochastic	Ant Colony Optimization	Predictive Accuracy

as the training set. The results (accuracy rate on the test set) are then averaged and reported in Table 3 as the accuracy rate of the classifier. Since ABC-Miner is a stochastic algorithm, we run it 10 times – using a different random seed to initialize the search each time – for each cross-validation fold. In the case of the deterministic algorithms, each is run just once for each fold.

Table 3 reports the mean and the standard error of predictive accuracy values obtained by 10-fold cross validation for the 15 datasets, where the highest accuracy for each dataset is shown in bold face. As shown, ABC-Miner has achieved the highest predictive accuracy amongst all algorithms in 12 datasets (with 2 ties), while Naïve-Bayes achieved the highest accuracy in 3 datasets (with 2 ties), TAN in 2 datasets (both are ties) and finally GBN in 4 datasets (with 3 ties).

Ranking the algorithms in descending order of accuracy for each dataset and taking the average ranking for each algorithm across all 15 datasets, ABC-Miner obtained a value of 1.6, which is the best predictive accuracy average rank

Table 3. Predictive Accuracy % (*mean* ± *standard error*) Results.

Dataset	Naïve-Bayes	TAN	GBN	ABC-Miner
bcw	92.1 ± 0.9	**95.4 ± 0.9**	93.8 ± 0.9	**95.4 ± 0.6**
car	85.3 ± 0.9	93.6 ± 0.6	86.2 ± 0.9	**97.2 ± 0.3**
cmc	52.2 ± 1.2	49.8 ± 1.2	49.8 ± 1.2	**67.3 ± 0.6**
crd-a	77.5 ± 1.2	85.1 ± 0.9	85.7 ± 0.9	**87.3 ± 0.6**
crd-g	**75.6 ± 0.9**	73.7 ± 1.2	**75.6 ± 1.2**	69.5 ± 0.9
drm	96.2 ± 0.6	97.8 ± 0.9	97.2 ± 0.6	**99.1 ± 0.3**
hay	80.0 ± 2.8	67.9 ± 3.1	**83.1 ± 2.5**	80.0 ± 3.1
hrt-c	56.7 ± 2.2	58.8 ± 2.5	56.7 ± 2.2	**73.3 ± 0.9**
iris	**96.2 ± 1.5**	94.2 ± 1.8	92.9 ± 1.8	**96.2 ± 0.9**
monk	**61.6 ± 0.6**	58.8 ± 0.6	**61.6 ± 0.9**	51.9 ± 0.9
nurs	90.1 ± 0.9	94.3 ± 0.9	90.1 ± 0.9	**97.0 ± 0.9**
park	84.5 ± 2.5	91.7 ± 2.2	84.5 ± 2.5	**94.2 ± 2.8**
pima	75.4 ± 1.2	**77.8 ± 1.5**	**77.8 ± 1.5**	**77.8 ± 1.5**
ttt	70.3 ± 0.3	76.6 ± 0.6	70.3 ± 0.3	**86.4 ± 0.6**
vot	90.3 ± 0.6	92.1 ± 0.4	90.3 ± 0.6	**94.6 ± 0.9**

amongst all algorithms. On the other hand Naïve-Bayes, TAN and GBN have obtained 3.1, 2.5, 2.8 in predictive accuracy average rank respectively. Note that the lower the average rank, the better the performance of the algorithm.

Statistical test according to the non-parametric Friedman test with the Holm's post-hoc test was performed on the average rankings. Comparing to Naïve-Bayes and GBN, ABC-Miner is statistically better with a significance level of 5% as the tests obtained p - values of 0.0018 and 0.013 respectively. Comparing to TAN, ABC-Miner is statistically better with a significance level of 10% as the tests obtained p -value of 0.077.

6 Concluding Remarks

In this paper, we introduced a novel ant-based algorithm for learning Bayesian network classifiers. Empirical results showed that our proposed ABC-Miner significantly out performs the well-known Naïve-Bayes, TAN, and GBN algorithms in term predictive accuracy. Moreover, the automatic selection of the maximum number of k-parents value makes ABC-Miner more adaptive and autonomous than conventional algorithms for learning BN classifiers. As a future work, we would like to explore the effect of using different scoring functions for computing the heuristic value used by ABC-Miner, as well as other scoring functions to evaluate the quality of a constructed BN classifier. Another direction is to explore different methods of choosing the value of k parents for building a network structure for the Bayesian classifier.

References

1. Buntine, W. : Theory refinement on Bayesian networks. *17th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann*, pp. 52–60 (1991).
2. De Campos, L.M., Gmez, J.A., Puerta, J.M. : Learning Bayesian network by ant colony optimisation. *Mathware and Soft Computing*, pp. 251–268 (2002).
3. Cheng, J. and Greiner, R. : Comparing Bayesian network classifiers. *15th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 101–108 (1999).
4. Cheng, J. and Greiner, R. : Learning Bayesian Belief Network Classifiers: Algorithms and System. *14th Biennial Conference: Advances in Artificial Intelligence*, pp. 141–151 (2001).
5. Chickering, D., Geiger, M., Heckerman, D. : Learning Bayesian networks is NP-complete. *Advanced Technologies Division, Microsoft Corporation, Redmond, WA, Technical Report*, (1994).
6. Cooper, G.F. and Herskovits, E. : A Bayesian method for the induction of probabilistic networks from data. *Machine Learning Journal*, pp. 309–348 (1992).
7. Daly, R., Shen, Q., Aitken, S. : Using ant colony optimization in learning Bayesian network equivalence classes. *Proceedings of UKCI*, pp. 111–118 (2006).
8. Daly, R. and Shen, Q. : Learning Bayesian network equivalence classes with ant colony optimization. *Journal of Artificial Intelligence Research*, pp. 391–447 (2009).
9. Dorigo, M. and Stützle, T. : Ant Colony Optimization. *MIT Press*, (2004).
10. Friedman, N., Geiger, D., Goldszmidt, M. : Bayesian Network Classifiers. *Machine Learning Journal*, pp. 131–161 (1997).
11. Friedman, N. and Goldszmidt, M. : Learning Bayesian networks with local structure. *Learning in Graphical Models, Norwell, MA: Kluwer*, pp. 421–460 (1998).
12. Heckerman, D., Geiger, D., Chickering, D.M. : Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning Journal*, pp. 197–244 (1995).
13. Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B. : Classification with ant colony optimization. *IEEE TEC*, pp. 651–665 (2007).
14. Otero, F., Freitas, A., Johnson, C.G. : *c*Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. *Ant Colony Optimization and Swarm Intelligence*, pp. 48–59 (2008).
15. Parpinelli, R.S., Lopes, H.S., Freitas, A. : Data mining with an ant colony optimization algorithm. *IEEE TEC*, pp. 321–332 (2002).
16. Pinto, P.C., Ngele, A., Dejori, M., Runkler, T.A., Costa, J.M. : Learning of Bayesian networks by a local discovery ant colony algorithm. *IEEE World Congress on Computational Intelligence*, pp. 2741–2748 (2008).
17. Pinto, P.C., Ngele, A., Dejori, M., Runkler, T.A., Costa, J.M. : Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning. *IEEE TEC*, pp. 767–779 (2009).
18. Salama, K.M., Abdelbar, A.M., Freitas A.A. : Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm. *Swarm Intelligence Journal*, pp. 149–182 (2011).
19. UCI Repository of Machine Learning Databases. Retrieved Oct 2011 from, URL:http://archive.ics.uci.edu/ml/index.html
20. Witten, H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. *second edition, Morgan Kauffman*, (2005).
21. Yanghui, Wu., McCall, J., Corne, D. : Two novel Ant Colony Optimization approaches for Bayesian network structure learning. *IEEE World Congress on Evolutionary Computation*, pp. 1–7 (2010).