Using Ant Colony Optimization to Build Cluster-Based Classification Systems

Khalid M. Salama[1] and Ashraf M. Abdelbar[2(✉)]

[1] School of Computing, University of Kent, Canterbury, UK
kms39@kent.ac.uk
[2] Department of Mathematics & Computer Science,
Brandon University, Manitoba, Canada
abdelbara@brandonu.ca

Abstract. Learning cluster-based classification systems is the process of partitioning a training set into data subsets (clusters), and then building a local classifier for each data cluster. The class of a new instance is predicted by first assigning the instance to its nearest cluster, and then using that cluster's local classification model to predict the instance's class. In this paper, we use the Ant Colony Optimization (ACO) meta-heuristic to optimize the data clusters based on a given classification algorithm in an integrated cluster-with-learn manner. The proposed ACO algorithms use two different clustering solution representation approaches: instance-based and medoid-based, where in the latter the number of clusters is optimized as part of the ACO algorithm's execution. In our experiments, we employ three widely-used classification algorithms, k-nearest neighbours, Ripper, and C4.5, and evaluate performance on 30 UCI benchmark datasets. We compare the ACO results to the traditional c-means clustering algorithm, where the data clusters are built prior to learning the local classifiers.

Keywords: Ant Colony Optimization (ACO) · Data mining · Classification · Clustering · Cluster-based classification system

1 Introduction

Classification is an important supervised learning task, concerned with predicting the class of a given instance based on its input attributes, using a well-constructed classification model [28]. The classification process consists of two stages. The training stage utilizes a *training set* of labelled instances, that is a set of instances along with their correct class labels, that should be sufficiently representative of the domain of interest. A classification algorithm uses the training set to construct an internal model of the relationships between the attributes of the input instances and their corresponding class labels. Then, during the subsequent operating stage, the classifier uses its internal model to predict the class of new unlabelled instances, which have not been presented to the algorithm during the training stage.

M. Dorigo et al. (Eds.): ANTS 2016, LNCS 9882, pp. 210–222, 2016.
DOI: 10.1007/978-3-319-44427-7_18

A cluster-based classification system consists of several local classifiers, one for each subset of the dataset, to model an asymmetric set of relationships between the predictor attributes and the predictable class for each data subset. Clustering techniques [3] are utilized to discover data partitions in which each cluster holds more consistent attributes-class relationships given the data subset in the partition. Consequently, more effective local classifiers are built for each data subset.

Ant Colony Optimization (ACO) [2] is a meta-heuristic for solving combinatorial optimization problems, inspired by observations of the behaviour of ant colonies in nature. While ACO has been used for data clustering in general [1,4,6,23], the focus of this paper is on using ACO to build classification systems, based on data partitions of the original training set. In other words, the performance of the proposed methods is evaluated mainly in terms of the predictive accuracy of the constructed classification system, rather than the cohesion/separation quality of the clusters *per se*.

In this paper we introduce AntClust-Miner, an algorithm which employs the ACO meta-heuristic to learn the data clusters and build the local classification models in a "cluster-with-learn" fashion. Such an integrated approach aims to find the best data partitions that improve the overall predictive accuracy of the cluster-based classification system. We use the AntClust-Miner algorithm with two ACO clustering methods: (1) instance-based, where a solution is a set of instance-class assignments, and (2) medoid-based, where a solution is a set of instances to be used as the mediods of clusters. In this paper, we extend the medoid-based ACO clustering method to automatically optimize the number of clusters as part of the ACO algorithm execution. For building the local classifiers, we use three widely used classification algorithms from three different classification families, namely, k-nearest neighbours from instance-based classification, Ripper from classification rule induction, and C4.5 from decision tree construction. We evaluate the performance of our proposed ACO algorithms on 30 UCI benchmark datasets, using 3, 5 and 8 clusters. We compare the ACO results to a cluster-based classification system based on the traditional c-means clustering algorithm, where the data clusters are built before learning the local classifiers.

The rest of the paper is structured as follows. We begin in Sect. 2 with a brief overview on ACO related work in the field of classification and clustering data mining problems. We then present our ACO approach for learning cluster-based classification systems in Sect. 3. Experimental methodology and results are presented in Sect. 4, and final remarks are offered in Sect. 5.

2 ACO Related Work

Inspired by the "intelligent" behaviour of natural ant colonies foraging to find the shortest path between a food source and the nest, Dorigo et al. have introduced Ant Colony Optimization (ACO) [2] as a population-based, global search meta-heuristic to solve a wide range of (mainly combinatorial) optimization problems.

A number of ACO-based classification algorithms have been introduced in the literature with different classification learning approaches [8]. Ant-Miner [12] is the first ant-based classification algorithm, which discovers a list of classification rules. The algorithm has been followed by several extensions in [7,9,10,12,15]. Two different ACO-based algorithms were proposed in [11,22] for inducing decision trees. ACO was also employed by Salama and Freitas in [18,19,21] to learn various types of Bayesian network classifiers. Blum and Socha applied an ACO variation for continuous optimization [5,25] to train feed-forward neural networks [24]. ANN-Miner [13] was introduced by Salama and Abdelbar as an ACO-based algorithm for learning neural network topologies. ACO has recently been utilized in data reduction for classification [14]. Furthermore, ACO has recently been applied to feature weighting in the context of instance-based learning [16].

The authors have introduced the idea of using ACO in building cluster-based classification systems in [17,20], but only using Bayesian network classifiers. In other words, in [17,20], ACO was employed to produce cluster-based Bayesian multi-nets models, which consist of only Naïve-Bayes or Tree-Augmented Naïve-Bayes local classifiers. However, in the present work we explore using ACO for building cluster-based classification models by employing three different algorithms to build the local classifiers: the k-nearest neighbour algorithm, the Ripper rule induction algorithm, and the C4.5 decision tree construction algorithm. Also, while the Medoid-based method for clustering solution representation was introduced in previous work [17,20], it had the limitation of needing the number of clusters to be supplied by the user. In this work, we extend the Medoid-based ACO method for building cluster-based classification systems by automatically optimizing the number of clusters to be used in the system.

A useful general review of (unsupervised) clustering is [29]. ACO algorithms for clustering are reviewed in [4].

3 Ant Colony for Building Cluster-Based Classifiers

3.1 The AntClust-Miner Overall Algorithm

The AntClust-Miner algorithm for building cluster-based classification systems carries out the data clustering process as well as the local classifiers construction process in a synergistic fashion via the ACO meta-heuristic. In such a "cluster-with-learn" approach, each ant in the colony creates a complete cluster-based classification system, rather than just a clustering solution. This is in contrast to having a—conventional or evolutionary-based—clustering algorithm complete the creation and optimization of the data clusters *before* the local models are constructed. The advantage of the integrated "cluster-with-learn" approach is that the clusters are optimized to maximize the predictive accuracy of the classification system, rather than just optimizing the cohesion/separation of the clusters. The overall procedure of AntClust-Miner is shown in Algorithm 1.

Before the ACO procedure begins, the *trainingSet* data is split into a *learningSet* and a *validationSet* (lines 6–7). The *learningSet* is 75 % of the

Algorithm 1. Pseudo-code of AntClust-Miner.

```
1: begin
2: K ⇐ input; /* number of clusters */
3: algorithm ⇐ input; /* classification algorithm */
4: trainingSet ⇐ input;
5: system_gbest = φ; /* final cluster-based classification system */
6: datasets = Split(training_set);
7: learningSet = datasets[0]; validationSet = datasets[1];
8: InitializeConstructionGraph(learningSet, K);
9: quality_gbest = 0; t = 1;
10: repeat
11:     system_tbest = φ;
12:     quality_tbest = 0;
13:     for i = 1 → colonySize do
14:         clustSolution_i = ant_i.CreateClusteringSolution();
15:         for k = 1 → K do
16:             dataSubset = clustSolution_i.GetDataCluster(k)
17:             classifier_k = LearnClassifier(dataSubset, algorithm);
18:             append classifier_k to system_i;
19:         quality_i = ComputeQuality(system_i, validationSet);
20:         if quality_i > quality_tbest then
21:             system_tbest = system_i;
22:             quality_tbest = quality_i;
23:     PerformLocalSearch(system_tbest);
24:     UpdatePheromone();
25:     if quality_tbest > quality_gbest then
26:         system_gbest = system_tbest;
27:         quality_gbest = quality_tbest;
28:     t = t + 1;
29: until t = maxIterations or Convergence();
30: return system_gbest;
31: end
```

trainingSet and is used to construct the data clusters and build the local classifiers, while the *validationSet*, consisting of the remaining 25 % of the training set, is used for evaluating the constructed classification system. Note that the class distribution in *learningSet* and *validationSet* is kept roughly the same as in *trainingSet*. The construction graph (discussed in the following subsections) is initialized in line 8 prior to the commencement of the ACO procedure.

In essence, the AntClust-Miner algorithm works as follows. In the inner **for**-loop (lines 13 to 22), each ant_i in the colony constructs a complete cluster-based classification $system_i$. A candidate $system_i$ is produced by first constructing a clustering solution $clustSolution_i$ in line 14 (this is based on the ACO clustering method used, which will be discussed in the following two subsections), then for each data cluster k, a local $classifier_k$ is created using the input `algorithm`, and appended to the classification $system_i$ (lines 15 to 18). Then, the $quality_i$ of the candidate $system_i$ produced by ant_i is evaluated using the *validationSet*

in line 19 (this will be discussed in Subsect. 3.4). The iteration-best $system_{tbest}$ produced in the colony at iteration t is maintained (lines 20 to 22).

At the end of iteration t, a local search procedure is performed on the iteration-best cluster-based classification $system_{tbest}$ (line 23). Basically, the clustering solution of $system_{tbest}$ undergoes c-means clustering, initialized by the centroids of the clustering solution. Then the pheromone amounts are updated (line 24) and the global-best cluster-based classification $system_{gbest}$ produced throughout the execution of the ACO algorithm is maintained (lines 25 to 27). This concludes one iteration of the outer **repeat**-loop, which continues until either the `maxIteration` is reached, or the ACO algorithm converges on a solution. The ACO algorithm converges when it produces the same solution in `convIteration` consecutive iterations. In our experiments, we set `colonySize` to 10, `maxIteration` to 10000, and `convIteration` to 10. Finally, $system_{gbest}$ is returned as the output of the algorithm.

3.2 Instance-Based ACO Clustering Method

As discussed in [17,20], the instance-based method for ACO clustering assigns each of the N instances in the *learningSet* to one out of K clusters. The construction graph contains $N \times K$ solution 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.

A candidate clustering solution in the instance-based method consists of N elements, the index of an element represents the instance, and the element represents the cluster assignment of this instance. For example, a candidate clustering solution for a dataset with 10 instances and 4 clusters is represented in the following form:

3	1	3	4	1	2	1	3	4	2

This representation means that in this candidate clustering solution, for example, the first instance belongs to cluster number 3, along with the third and the eighth instances. However, such a clustering solution representation method has two drawbacks. First, the mapping between a clustering solution and this representation is one-to-many [20]. That is, the same clustering solution can be obtained by several instance-based representations. This is an example of the classical *competing conventions* problem studied for example by Whitley et al. [27]. 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. Second, the number of the clusters has to be supplied to the algorithm, rather than being optimized within the algorithm.

3.3 Medoid-Based ACO Clustering Extended Method

The medoid-based ACO clustering method avoids the competing conventions problem of the instance-based method. In the medoid-based method, a clustering solution is represented by the choice of V instances to act as the cluster medoids, where V is the number of clusters. More precisely, the construction graph is a complete (fully connected) graph containing only N solution components (recall that N denotes the number of instances), each representing an instance that is a candidate cluster medoid.

When constructing a solution, an ant chooses V instances based on the pheromone information associated with each solution component. The chosen instances represent a medoid-based clustering solution, where the k-th element (instance) represents the medoid of the 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.

Once a medoid-based candidate solution has been constructed by an ant, each instance i in the *learningSet* is assigned to the k-th cluster for which the similarity between instance i and the k-th medoid is the highest (compared to the other medoids). The similarity measure is discussed in the following subsection.

In this paper, we extend the medoid-based method to allow the ACO procedure to optimize the number of clusters. The algorithm receives K as the maximum number of clusters that can be produced. Every integer within the inclusive range 2 to K has an associated decision component in the construction graph, and an associated pheromone amount. Each training instance also has an associated decision component and associated pheromone amount. Thus, the construction graph consists of $(N + K)$ decision components.

When constructing a candidate solution, an ant first probabilistically selects the number of clusters V (between 2 and K) by selecting one of these decision components based on their pheromone amounts. It then selects V instances to act as cluster medoids. The V instances are selected one by one; each selection is made probabilistically based on pheromone amounts and under the constraint that an instance cannot be selected more than once.

3.4 Quality Evaluation and Pheromone Update

The AntClust-Miner algorithm evaluates the quality of a candidate solution as a classifier, rather than a clustering solution. That is, we use predictive accuracy, a widely-used measure in the context of classification, as an evaluation measure for the quality of the produced cluster-based classification system as a whole, rather than the cohesion/separation measure of the clusters per se. The *validationSet* is used to calculate the predictive accuracy of a candidate classification system as follows. First, each instance is assigned to its nearest cluster. Then, the local classifier of that cluster is used to predict the class of the validation instance. The predictive accuracy is computed as the ratio of the number of correctly classified instances to the total number of instances in the validation set.

The nearest cluster for instance i is the one that has the maximum total similarity between instance i and each instance j in the cluster. The similarity between any two instances i and j is computed according to:

$$\textbf{Similarity}(i,j) = 1 - \sum_{v=1}^{a} \sum_{l=1}^{|C|} |P(C_l|i_v) - P(C_l|j_v)|, \tag{1}$$

where a is the number of attributes and $P(C_l|i_v)$ is the conditional probability of the class value C_l given the attribute value i_v. This conditional probability is measured empirically as the ratio of the number of instances with attribute value i_v and the class value C_l to the number of instances with attribute value i_v in the *learningSet*. Accordingly, if a class value occurs frequently with two attribute values, then these two values are considered similar; the same applies if the class value does not occur frequently with both of the attribute values. On the other hand, if a class value occurs frequently with one attribute value, but does not occur frequently with the other attribute value, then these two attribute values are considered dissimilar. Note that the continuous attributes in the *learningSet* are discretized using the supervised C4.5-disc algorithm [28].

As for pheromone update, pheromone level τ_x is increased on each solution component x of the construction graph included in the clustering solution, based on the quality of the iteration-based cluster-based classification solution ($system_{tbest}$), using the following formula:

$$\tau_x(t+1) = \tau_x(t) + \tau_x(t) \cdot quality_{tbest}(t) \tag{2}$$

To simulate pheromone evaporation, normalization is then applied; each τ_x is divided by the total pheromone amounts in the construction graph.

4 Experimental Methodology and Results

In our experiments, we used three classification algorithms to learn the local classifiers of the cluster-based classification systems constructed by our AntClust-Miner algorithm, namely the k-nearest neighbour lazy learning algorithm, the Ripper classification rule induction algorithm, and the C4.5 decision tree construction algorithm [26,28]. We used the WEKA implementation for these algorithms: k-NN, JRip, and J48, and used WEKA's default parameter settings in each case. We evaluated the performance of the two versions of the AntClust-Miner algorithm (IB-AntClust-Miner and MB-AntClust-Miner), using each algorithm g to build the local classifiers, against several methods. Specifically, for each classification algorithm g, we evaluated IB-AntClust-Miner (with 3, 5, and 8 clusters) and MB-AntClust-Miner, using g as the underlying local classification algorithm. We compared these to the base algorithm g (using the whole training set to build the classifier without any data clustering). We further compared performance to a cluster-based classification system with the c-means algorithm as the clustering algorithm and g as the underlying local classifier (without any

ACO)—this was evaluated with 3, 5, and 8 clusters. All of these evaluations used 30 benchmark datasets from the University of California at Irvine (UCI) Machine Learning repository. In all, 24 algorithms were evaluated and compared in this study.

Table 1. Predictive accuracy (%) results using k-NN as the local classifier.

Dataset	k-NN	c-means-3	c-means-5	c-means-8	IB-Ant-3	IB-Ant-5	IB-Ant-8	MB-Ant
audiology	50.00	78.00	36.42	54.43	65.00	62.50	58.07	**79.17**
automobile	48.36	71.64	68.85	**74.62**	56.02	56.69	60.03	65.05
breast-l	61.09	63.99	64.20	63.93	67.04	63.18	64.69	**74.89**
breast-p	65.24	65.18	64.88	63.68	**70.21**	67.55	65.48	68.58
breast-w	93.85	93.61	94.26	95.39	95.09	94.31	**95.88**	95.22
car	66.67	59.81	65.78	66.31	72.75	74.62	63.85	**74.77**
chess	78.18	82.53	76.49	72.33	**86.07**	79.25	79.39	85.60
credit-a	77.68	79.02	79.32	78.92	80.29	**81.01**	79.77	79.57
credit-g	65.40	69.50	68.30	68.91	68.00	68.80	66.98	**69.98**
cylinder	55.92	66.75	72.41	72.29	60.38	50.37	72.60	**72.91**
dermat	82.21	92.53	87.55	91.53	**92.62**	88.77	86.12	89.92
ecoli	71.82	79.31	77.67	77.41	**81.88**	77.72	79.34	80.11
heart-c	53.18	50.52	53.36	53.16	52.44	51.23	**53.77**	53.16
heart-h	59.30	44.55	60.81	50.81	56.57	58.16	62.61	**62.76**
horse	71.07	77.05	75.24	78.01	77.87	78.89	**79.15**	77.30
ionospere	73.66	85.38	79.42	78.09	69.68	79.97	80.67	**85.81**
liver	53.34	58.87	63.21	61.22	62.05	**65.20**	62.01	64.58
lymphography	75.00	77.10	81.09	**81.21**	77.00	76.48	78.95	79.10
monks	58.18	54.73	58.93	60.48	60.91	59.27	**69.58**	61.64
pima	65.63	68.31	67.58	67.31	65.24	68.98	67.66	**71.11**
s-heart	78.52	71.33	80.79	79.10	77.78	74.07	82.19	**82.52**
segmentation	78.48	**93.60**	81.42	84.87	88.74	82.46	85.82	93.17
soybean	32.07	**86.62**	84.50	56.81	86.55	84.97	58.99	86.55
thyroid	85.89	94.23	**96.42**	95.43	93.48	89.31	95.84	96.41
transfusion	62.85	59.55	63.42	62.55	62.84	64.33	**65.20**	64.74
ttt	70.53	65.37	65.23	70.17	72.11	**73.79**	70.40	72.32
vertebral-2c	80.00	78.97	80.22	82.50	77.10	**84.65**	83.61	82.35
vertebral-3c	73.87	76.71	71.59	79.47	75.81	79.29	**79.93**	79.35
voting	86.85	86.73	87.29	**88.35**	88.28	86.84	86.91	88.17
wine	88.17	92.94	95.48	96.55	96.63	**97.52**	97.21	97.49
Rank (avg.)	6.7	5.2	5.0	4.7	4.3	4.3	3.5	**2.3**

The experiments were carried out using the *stratified* 10-fold cross validation procedure. The results (accuracy rate on the test set) are averaged and reported as the accuracy rate of the classifier. Because ACO is a stochastic method, we ran our AntClust-Miner algorithms ten times—using a different random seed to initialize the search each time—in each iteration of the cross-validation procedure, and took the average accuracy as the iteration result. The c-means algorithm

was also run ten times in each cross validation iteration with different cluster initializations, and the average accuracy was taken as the iteration result.

Tables 1, 2 and 3 report the predictive accuracy results of our experiments, one for each base algorithm: k-NN, Ripper, and C4.5, respectively. In each table, the first column shows the results for the base algorithm (i.e., using the whole training set to build the classifier without any data clustring). The second, third and fourth columns show the results for the c-means algorithm, where the number of clusters are 3, 5 and 8 respectively. The fifth, sixth and seventh column show the results for the IB-AntClust-Miner algorithm, with 3, 5 and 8 clusters, respectively. The last column shows the results for the MB-AntClust-Miner algorithm—note that the latter optimizes the number of clusters, and the maximum number of clusters was set to 8. The last row of each table shows the average rank of each algorithm over the 30 datasets in terms of the predictive accuracy results. Note that in the

Table 2. Predictive accuracy (%) results using Ripper as the local classifier.

Dataset	Ripper	c-means-3	c-means-5	c-means-8	IB-Ant-3	IB-Ant-5	IB-Ant-8	MB-Ant
audiology	**77.17**	49.17	43.33	43.92	51.67	44.48	49.17	56.83
automobile	66.69	62.14	66.07	60.12	72.12	68.96	68.43	**80.91**
breast-l	67.44	69.13	67.54	69.42	69.92	69.85	68.24	**70.87**
breast-p	73.79	**78.82**	64.71	66.85	74.58	66.96	72.63	70.06
breast-w	92.20	90.82	87.89	**94.89**	92.39	93.94	85.23	93.79
car	**85.66**	78.30	69.82	65.54	78.60	70.56	66.50	74.77
chess	**97.00**	93.18	87.39	85.61	92.61	87.70	86.10	91.58
credit-a	83.51	82.93	81.59	75.23	84.03	85.98	**86.96**	86.17
credit-g	70.20	69.20	69.00	66.80	**70.50**	69.39	68.45	70.36
cylinder	62.29	56.72	52.56	65.55	72.97	**75.80**	72.68	73.38
dermat	86.01	86.91	31.66	88.17	86.51	91.75	79.21	**93.80**
ecoli	79.86	65.50	64.00	80.58	81.18	81.82	81.44	**82.67**
heart-c	52.15	47.49	49.78	52.48	52.47	54.56	54.74	**54.81**
heart-h	61.72	61.60	57.83	63.77	64.70	65.13	62.68	**65.34**
horse	81.54	74.92	79.59	75.97	82.19	81.00	80.19	**82.32**
ionospere	**87.66**	66.95	69.99	73.46	70.05	72.93	76.46	73.96
liver	64.34	56.81	60.02	60.34	**64.71**	63.31	64.70	63.22
lymphography	**77.95**	57.24	64.86	68.86	74.57	76.44	73.89	74.78
monks	58.36	61.64	61.09	61.51	62.36	61.88	60.18	**62.94**
pima	71.55	68.88	69.78	62.63	71.61	**73.07**	72.28	72.41
s-heart	76.52	62.22	78.52	83.29	77.30	83.65	81.11	**83.81**
segmentation	**92.10**	76.54	84.81	74.22	88.88	83.84	79.98	90.05
soybean	**81.10**	42.41	42.07	49.47	50.34	40.00	48.62	50.35
thyroid	90.53	75.80	79.83	73.23	91.70	93.45	91.92	**93.80**
transfusion	71.71	67.94	71.00	69.41	**72.48**	71.89	70.06	72.06
ttt	**96.00**	72.11	61.05	52.41	69.05	60.40	51.16	69.83
vertebral-2c	80.58	81.29	80.00	78.55	**83.87**	79.04	76.13	82.52
vertebral-3c	78.32	74.19	79.35	78.71	79.68	81.08	80.03	**82.34**
voting	91.66	92.95	90.53	89.29	92.93	92.37	90.48	**93.14**
wine	90.19	79.38	83.63	82.55	92.65	95.07	92.91	**96.05**
Rank (avg.)	4.1	5.8	6.4	6.0	3.2	3.5	4.9	**2.0**

following tables, IB-Ant refers to AntClust-Miner with the instance-based clustering method, while MB-Ant refers to AntClust-Miner with the medoid-based clustering method.

As shown in Table 1, MB-AntClust-Miner with k-NN obtained the best overall average rank of 2.3, and achieved the best results in 9 datasets. IB-AntClust-Miner using 8 clusters came in second place with an overall rank of 3.4, achieving the best results in 6 datasets, followed by IB-AntClust-Miner using 3 clusters, which obtained 4.2 as an overall rank and achieved the best results in 5 datasets. With k-NN, the Friedman test with the Holm *post hoc* test, at the conventional 0.05 significance level, indicates that MB-AntClust-Miner is significantly better than the other algorithms, except for IB-AntClust-Miner with 8 clusters.

Table 3. Predictive accuracy (%) results using C4.5 as the local classifier.

Dataset	C4.5	c-means-3	c-means-5	c-means-8	IB-Ant-3	IB-Ant-5	IB-Ant-8	MB-Ant
audiology	**80.50**	46.67	55.75	52.50	64.42	50.00	55.93	65.83
automobile	**79.36**	56.07	69.25	61.50	76.92	67.43	60.08	78.79
breast-l	**70.86**	69.49	68.17	65.74	69.42	69.48	67.20	67.12
breast-p	69.08	71.29	72.04	67.13	**74.44**	70.24	72.01	72.68
breast-w	92.91	90.49	94.59	86.48	94.46	91.18	**95.03**	94.04
car	**90.98**	72.28	67.67	63.63	72.10	68.07	64.05	73.74
chess	**97.47**	91.38	87.68	84.97	91.00	88.71	84.88	89.75
credit-a	83.80	78.84	82.44	77.10	**85.59**	84.10	83.90	82.03
credit-g	67.60	69.20	68.70	68.50	**70.08**	65.80	67.21	68.90
cylinder	72.50	53.95	67.29	56.74	70.15	58.51	68.19	**72.74**
dermat	**92.00**	43.39	86.62	84.02	86.94	82.80	86.71	91.01
ecoli	81.66	50.26	80.34	64.05	80.21	**81.82**	79.24	81.66
heart-c	49.21	50.80	53.96	49.92	54.93	51.82	53.71	**55.88**
heart-h	64.73	31.12	63.35	63.58	**65.54**	64.97	61.85	64.42
horse	**80.75**	78.94	80.03	76.25	80.43	79.34	78.83	80.36
ionospere	**86.26**	70.69	70.21	61.91	71.94	72.14	71.28	75.37
liver	62.56	61.42	58.25	58.05	**63.71**	62.82	63.22	63.39
lymphography	74.38	68.24	**76.98**	62.24	74.68	72.86	76.38	74.95
monks	59.46	60.55	60.67	58.55	61.25	61.64	60.11	**63.09**
pima	70.51	72.01	71.08	67.43	73.48	**74.52**	73.67	64.99
s-heart	73.56	75.56	75.19	77.41	**84.65**	83.78	83.07	75.93
segmentation	**93.59**	87.04	80.49	74.95	88.07	81.62	70.72	84.64
soybean	**81.11**	45.17	37.61	39.66	49.99	38.97	37.88	53.45
thyroid	89.15	79.03	94.09	93.31	**94.47**	93.01	92.92	93.48
transfusion	**71.78**	71.37	70.44	66.33	71.25	69.78	68.62	71.57
ttt	**83.58**	79.26	67.59	54.95	80.92	68.95	55.49	81.89
vertebral-2c	79.29	80.32	77.10	69.68	**80.63**	78.06	78.63	79.55
vertebral-3c	76.71	80.32	75.81	80.32	**81.22**	79.72	79.75	72.90
voting	92.48	**93.92**	92.01	90.53	93.50	92.80	90.63	92.68
wine	91.30	87.12	83.66	93.04	**96.15**	94.22	93.13	91.60
Rank (avg.)	3.6	5.2	5.1	6.7	**2.4**	4.5	5.1	3.3

The results shown in Table 2 indicate that MB-AntClust-Miner with Ripper obtained the best overall rank of 2.0, and achieved the best results in 13 datasets. In second place came IB-AntClust-Miner using 3 clusters with an overall rank of 3.1, achieving the best results in 4 datasets. IB-AntClust-Miner using 5 clusters came in third place by obtaining 3.5 as an overall rank and achieving the best results in 2 datasets. With Ripper, the Friedman test with the Holm *post hoc* test, at the conventional 0.05 significance level, indicates that MB-AntClust-Miner is significantly better than the other algorithms except for IB-AntClust-Miner with 3 and 5 clusters.

The results for C4.5 shown in Table 3 indicate that IB-AntClust-Miner using 3 clusters obtained the best overall rank of 2.4, and achieved the best results in 10 datasets. MB-AntClust-Miner came in the second place with an overall rank of 3.3, achieving the best results in 3 datasets. In third place came the baseline C4.5 algorithm, which obtained 3.6 as an overall rank and achieved the best results in 12 datasets. With C4.5, the Friedman test with the Holm *post hoc* test, at the conventional 0.05 significance level, indicates that IB-AntClust-Miner with 3 clusters is significantly better than c-means with 3, 5, and 8 clusters, and also significantly better than IB-AntClust-Miner with 5 and 8 clusters.

5 Conclusions and Future Work

In this paper, we have employed Ant Colony Optimization to create cluster-based classification systems. The AntClust-Miner algorithm produces the data clusters and learns the local models in an integrated fashion so that the clusters are optimized to maximize the predictive accuracy of the overall classification system. We used two ACO clustering methods, instance-based and medoid-based, and we extended the medoid-based method to automatically optimize the number of clusters. In our experiments, we used three widely-used classification algorithms, namely k-nearest neighbours, Ripper, and C4.5. We compared our ACO algorithm to the baseline classifiers and to the cluster-based classifiers using the c-means algorithm to produce the clusters (instead of the ACO algorithms), using 30 datasets. The results indicate that the ACO algorithms perform statistically significantly better than c-means in terms of the predictive accuracy of the produced classification models with k-NN, Ripper, and C4.5, and also significantly better than the base k-NN and Ripper algorithms (however, in the case of the C4.5 base algorithm, the difference was not statistically significant). It is not clear why the C4.5 algorithm benefited the least from the cluster-based classification approach, and we would like to explore this further in future work.

We would also like to consider allowing different classification algorithms to be used to learn the local models, based on the characteristics of each cluster, with the choice of the algorithm to use per data cluster being automatically optimized by the ACO procedures. Furthermore, we would like to investigate using all the constructed local models to decide on the class of an instance using a weighted voting ensemble method, where the weight would be computed based on the fuzzy membership of the instance in each cluster.

Acknowledgments. Partial support of a grant from the Brandon University Research Council is gratefully acknowledged.

References

1. Abdelbar, A.M., Salama, K.M.: Clustering with the ACOR algorithm. In: Swarm Intelligence, LNCS, vol. 9882, pp. 210–222 (2016)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
3. Gan, G., Ma, C., Wu, J.: Data Clustering: Theory, Algorithms, and Applications. SIAM Press, Philadelphia (2007)
4. Jafar, M., Sivakumar, R.: Ant-based clustering algorithms: a brief survey. Int. J. Comput. Theor. Eng. **2**, 787–796 (2010)
5. Liao, T., Socha, K., de Montes Oca, M., Stützle, T., Dorigo, M.: Ant colony optimization for mixed-variable optimization problems. IEEE Trans. Evol. Comput. **18**(4), 503–518 (2014)
6. Liu, X.Y., Fu, H.: An effective clustering algorithm with ant colony. J. Comput. **5**, 598–605 (2010)
7. Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Trans. Evol. Comput. **11**(5), 651–665 (2007)
8. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: swarm intelligence for data mining. Mach. Learn. **82**(1), 1–42 (2011)
9. Otero, F.E., Freitas, A.A., Johnson, C.: A new sequential covering strategy for inducing classification rules with ant colony algorithms. IEEE Trans. Evol. Comput. **17**(1), 64–74 (2013)
10. Otero, F.E., Freitas, A.A., Johnson, C.G.: Handling continuous attributes in ant colony classification algorithms. In: IEEE Symposium on Computational Intelligence in Data Mining (CIDM 2009), pp. 225–231 (2009)
11. Otero, F.E., Freitas, A.A., Johnson, C.G.: Inducing decision trees with an ant colony optimization algorithm. Appl. Soft Comput. **12**(11), 3615–3626 (2012)
12. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
13. Salama, K.M., Abdelbar, A.M.: Learning neural network structures with ant colony algorithms. Swarm Intell. **9**(4), 229–265 (2015)
14. Salama, K.M., Abdelbar, A.M., Anwar, I.M.: Data reduction for classification with ant colony optimization. Intelligent Data Analysis (2016, to appear)
15. Salama, K.M., Abdelbar, A.M., Freitas, A.A.: Multiple pheromone types and other extensions to the ant-miner classification rule discovery algorithm. Swarm Intell. **5**(3–4), 149–182 (2011)
16. Salama, K.M., Abdelbar, A.M., Helal, A.Z., Freitas, A.A.: Instance-based classification with ant colony optimization. Intelligent Data Analysis (accepted, 2016)
17. Salama, K.M., Freitas, A.A.: Clustering-based Bayesian multi-net classifier construction with ant colony optimization. In: IEEE Congress on Evolutionary Computation (IEEE CEC), pp. 3079–3086 (2013)
18. Salama, K.M., Freitas, A.A.: Learning Bayesian network classifiers using ant colony optimization. Swarm Intell. **7**(2–3), 229–254 (2013)
19. Salama, K.M., Freitas, A.A.: ABC-Miner+: constructing Markov blanket classifiers with ant colony algorithms. Memetic Comput. **6**(3), 183–206 (2014)
20. Salama, K.M., Freitas, A.A.: Classification with cluster-based Bayesian multi-nets using ant colony optimization. Swarm Evol. Comput. **18**, 54–70 (2014)

21. Salama, K.M., Freitas, A.A.: Ant colony algorithms for constructing Bayesian multi-net classifiers. Intell. Data Anal. **19**(2), 233–257 (2015)
22. Salama, K.M., Otero, F.E.: Learning multi-tree classification models with ant colony optimization. In: 6th International Conference on Evolutionary Computation Theory and Applications (ECTA 2014), pp. 38–48 (2014)
23. Shelokar, P.S., Jayaraman, V.K., Kulkarni, B.D.: An ant colony approach for clustering. Anal. Chim. Acta **509**(2), 187–195 (2004)
24. Socha, K., Blum, C.: An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput. Appl. **16**, 235–247 (2007)
25. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**, 1155–1173 (2008)
26. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining, 2nd edn. Addison Wesley, Reading (2005)
27. Whitley, D., Dominic, S., Das, R., Anderson, C.: Genetic reinforcement learning for neurocontrol problems. Mach. Learn. **13**(2–3), 259–284 (1993)
28. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2010)
29. Xu, R., Wunsch, D.: Clustering. Wiley-IEEE Press, Hoboken (2009)