Automatic Design of Ant-Miner Mixed Attributes for Classification Rule Discovery

Ayah Helal
School of Computing
University of Kent
Chatham Maritime, UK
amh58@kent.ac.uk

Fernando E. B. Otero
School of Computing
University of Kent
Chatham Maritime, UK
F.E.B.Otero@kent.ac.uk

ABSTRACT

Ant-Miner Mixed Attributes (Ant-Miner$_{\text{MA}}$) was inspired and built based on ACO$_{\text{MV}}$, which uses an archive-based pheromone model to cope with mixed attribute types. On the one hand, the use of an archive-based pheromone model improved significantly the runtime of Ant-Miner$_{\text{MA}}$ and helped to eliminate the need for discretisation procedure when dealing with continuous attributes. On the other hand, the graph-based pheromone model showed superiority when dealing with datasets containing a large size of attributes, as the graph helps the algorithm to easily identify good attributes. In this paper, we propose an automatic design framework to incorporate the graph-based model along with the archive-based model in the rule creation process. We compared the automatically designed hybrid algorithm against existing ACO-based algorithms: one using a graph-based pheromone model and one using an archive-based pheromone model. Our results show that the hybrid algorithm improves the predictive quality over both the base archive-based and graph-based algorithms.

CCS CONCEPTS

•Computing methodologies → Supervised learning by classification;

KEYWORDS

Ant Colony Optimization; data mining; classification rules; sequential covering

ACM Reference format:
Ayah Helal and Fernando E. B. Otero. 2017. Automatic Design of Ant-Miner Mixed Attributes for Classification Rule Discovery. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071306

1 INTRODUCTION

Ant Colony Optimization (ACO) [2] was originally designed to solve optimization problems in discrete search spaces, where solutions are defined by the path taken by each ant. A graph-based pheromone model is used to guide the ants in a discrete search space, where solution components are represented by nodes of the graph. Recently, Liao et al. [7] proposed the Ant Colony Optimization for Mixed Variables (ACO$_{\text{MV}}$), a new approach for ACO to handle mixed variables (continuous and discrete) optimization problems. ACO$_{\text{MV}}$ uses an archive-based pheromone model to guide the ants in the mixed variables search space, employing different sampling strategies according to the variable type.

Ant-Miner [18] uses the principles behind the ACO meta-heuristic to build classification rules in data mining context. The best quality ant deposits pheromone on each used node as an indication of the current best attributes. This allows ants to learn and select the most effective attributes to use in rule creation. Originally proposed to handle data mining problems with discrete attributes, Ant-Miner was subsequently extended to handle continuous attributes in [15, 16]. Recently, this work has been further extended to handle continuous, categorical and ordinal attributes in the Ant-Miner$_{\text{MA}}$ algorithm [4]. Ant-Miner$_{\text{MA}}$ achieves this feature by using an archive-based model to cope with the various types of attributes presented by the classification problem at hand, improving significantly the runtime when dealing with continuous attributes.

In this paper we propose a framework to incorporate the graph-based model of Ant-Miner in the rule creation process, along with the archive-based model. The graph-based model allows the algorithm to select the best attributes when creating classification rules, while the archive allows the algorithm to deal with different attributes types.

The remainder of this paper is organised as follows. We begin by reviewing the Ant-Miner algorithm in Section 2. We then present our automatically configurable approach in Section 3. Computation results are presented in Section 4, and conclusions and future directions are discussed in Section 5.

2 BACKGROUND

In this section, we will discuses the two main approaches for applying ACO in classification problems. We first review the original graph-based approach employed in Ant-Miner algorithm. Then, we present an overview of the archive-based algorithm Ant-Miner$_{\text{MA}}$.

2.1 Graph-based Pheromone Model

Graph-based approaches started with Ant-Miner [18], which was limited to discrete datasets only. Continuous attributes were discretised in a pre-processing stage and used as nominal attributes. Ant-Miner uses a graph-based approach to extract `IF-THEN` classification rules from data. Let r be a rule, each rule is a n-dimensional vector of terms t_n that are joined by `AND`s, such as `IF` t_1 `AND` t_2

GECCO '17, Berlin, Germany

978-1-4503-4920-8/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071306

. . . AND t_n THEN *class*. Each term t_i consists of a tuple (attribute, operator, value) and the *class* is the value predicted by the rule.

The Ant-Miner construction graph consists of a fully connected graph. Let a_i be a nominal attribute and v_{ij} be the j-th value of a_i attribute. For $j = 1, ..., D_i$, where D_i is the number of values in the domain of attribute a_i. Each v_{ij} is added as a node (a_i, =, v_{ij}) to the graph. Suppose an ant l is creating a rule r_l. It starts with an empty rule at node x and probabilistically chooses to visit a node y based on the amount of pheromone and heuristic information. This process is repeated until the ant cannot add more nodes to the rule.

Martens et al. [13] reviewed several extensions of Ant-Miner. Ant-Miner2 [10] added a somewhat simpler heuristic function using density estimation, while Ant-Miner3 [9] modified the rule update mechanism and increased exploration by means of a different transition rule. Ant-Miner+ [14] extended Ant-Miner with a class-based heuristic information, where an ant pre-selects the predicted class value and extracts a rule accordingly. It also employs a different pheromone initialization and update procedure based on $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [21], and copes with ordinal attributes by using a simplified construction graph.

*c*Ant-Miner was proposed by Otero et al. [15, 16], where a dynamic entropy-based discretisation method is proposed for handling continuous attributes during the rule construction process. The use of the minimum description length principle in *c*Ant-Miner allow construction of discrete intervals with lower and upper bounds. *c*Ant-Miner's pheromone model added a node for each continuous attribute. When a node representing a continuous attribute is selected, the dynamic discretisation procedure is used to choose an operator and value to create a term in the form ($a_i \leq v_{ij}$), ($a_i > v_{ij}$) or ($v_{1ij} < a_i \leq v_{2ij}$).

*c*Ant-Miner$_{\text{PB}}$ was proposed by Otero et al. [17], which included a new strategy to create a list of rules. Each ant would create a complete list of rules at each iteration, rather than just one rule at a time. One important characteristic of the new strategy is that ants are guided by the quality of a complete list of rules, allowing the algorithm to cope better with the problem of rule interation. Ant-Miner$_{\text{PAE}}$ proposed by Yang et al. [22] extended *c*Ant-Miner$_{\text{PB}}$by incorporating a principal of attraction and exclusion of pheromone.

The graph-based pheromone model has been heavily researched since the introduction of Ant-Miner. An alternative approach of extending Ant-Miner is the implementation of a hybrid approach, such as PSO/ACO2 [5]. The main motivation is to combine the strength points of both PSO and ACO approaches: PSO is known to cope well with continuous-valued attributes, while ACO naturally handles nominal-valued ones.

2.2 Archive-based Pheromone Model

Ant-Miner$_{\text{MA}}$ proposed by Helal and Otero [4], is an Ant-Miner mixed-attributes approach for classification rule discovery. Ant-Miner$_{\text{MA}}$ was inspired by the ACO$_{\text{MV}}$ algorithm to handle mixed attributes types at the same time, eliminating the need of a discretisation—either as a pre-processing or dynamically—when handling a continuous attribute, and also coping with ordinal attributes. An archive is used to sample conditions for the creation of the rules, instead of using ants to traverse a construction graph.

Ant-Miner$_{\text{MA}}$ starts by initializing the solution archive with R random generated rules (solutions). Each solution S_j is associated with weight w_j related its quality $Q(S_j)$, where w_j is calculated using a Gaussian function. In each iteration, an ant creates a single rule using an ACO$_{\text{MV}}$ procedure. Once m new rules have been created, where m is the number of ants, they are added into the solution archive. The R and m rules are sorted based on quality of solutions and the m worst ones are removed from the archive. The procedure to create a new rule is repeated until the maximum number of iterations has been reached, or a restart procedure is applied to avoid early stagnation.

During rule creation, the procedure probabilistically decides which terms to include using a discrete sampling procedure on the archive to sample an *enable* flag to determine if the term should be included or not. If the term is enabled, a discrete sampling procedure is used to choose to choose an operator. Finally, the value of each attribute is then determined: if the attribute type is continuous, a continuous sampling procedure is used; if the attribute is ordinal, a continuous sampling procedure is used and the result is rounded up to the nearest index to preserve the natural order of the attribute values; categorical terms are determined using a discrete sampling procedure.

The sampling of continuous values is handled by ACO$_{\text{MV}}$ using ACO$_{\mathbb{R}}$ [19], where each ant i probabilistically chooses one solution from the archive based on their weight and samples a new solution around the chosen solution using a normal probability density function. The ordinal sampling works exactly as the continuous, with the difference that the real value is rounded to the nearest valid index. The categorical sampling procedure was first proposed in ACO$_{\text{MV}}$ [19]. Given a categorical variable x that has t possible values, each ant has to choose v_i—where $v_i \in \{v_1, v_2, \ldots, v_t\}$—according to a weight calculated based on two components: the first component biases the sampling towards values that are used in high-quality solutions but do not occur very frequently in the archive; the second component biases the sampling towards unexplored values for the given attribute.

The Ant-Miner$_{\text{MA}}$ approach, which is the only Ant-Miner variation using an archive-based pheromone model to the best of our knowledge, has shown competitive results with *c*Ant-Miner algorithm. The archive-based pheromone model improved significantly the runtime, since it eliminates the need for a discretisation procedure. On the other hand, the graph-based pheromone model showed superiority when dealing with datasets that have a large number of attributes, as the graph allows the algorithm to easily identify good attributes to use.

3 AUTOMATIC DESIGN OF ANT-MINER$_{\text{MA}}$

As discussed in the previous section, both graph-based and archive-based pheromone models have their merits. Combining concepts from both approaches could potentially lead to improved runtime and a better capacity to handle datasets with a large number of attributes.

There are a number of design questions when building a framework to combine both archive-based and graph-based pheromone models. Ant-Miner$_{\text{MA}}$ uses the archive pheromone model to sample rule term components, such as the attribute, the operator and

Algorithm 1: High-level pseudocode of Ant-Miner$_{\text{MA+G}}$

```
Data: training data
Result: list of rules
RuleList ← {}
while TrainingData.Size() <MaxUncovered do
    $A$ ← Generate Random Rules
    while $t$ ¡ MaxIterations and not Restarted do
        $A_t$ ← {}
        while $i$ ¡ number of ants do
            $R_i$ ← Create New Rule (Section 3.1)
            $R_i$ ← Prune($R_i$) (Section 3.2)
            $i \leftarrow i + 1$
            $A_t \leftarrow R_i$
        end
        $A$ ← UpdateArchive($A_t$) (Section 3.3)
        $A$ ← UpdateGraph($A_t$) (Section 3.3)
        $t \leftarrow t + 1$
        if stagnation() then
            Restart($A$) (Section 3.3.3)
            Restarted ← True
        end
    end
    $R_{best}$ ← BestRule($A$) (Section 3.2)
    RuleList ← RuleList + $R_{best}$
    TrainingData ← TrainingData − covered($R_{best}$)
end
return RuleList
```

the value; ants in *c*Ant-Miner traverse the graph-based pheromone model to create rules, using the pheromones deposited on each node as an indication of the current best attributes-value pairs, and in the case of continuous attributes, they use a dynamic discretisation procedure based on the entropy measure.

The design questions that we are interested in this work are:

(1) *Should the archive pheromone model be used only for continuous values, or should it also be used for nominal and ordinal values?*
(2) *Should the operator be selected using the archive pheromone model, or should it be added to the graph pheromone model?*
(3) *How should both pheromone models be updated?*

Instead of following a manual approach of testing each possible configuration of Ant-Miner$_{\text{MA}}$, which would require large amount of human and computational time, we propose the use of an automated algorithm configuration tool to obtain a high-performing Ant-Miner$_{\text{MA}}$ variant. We are inspired by the work of Lopez and Stützle [12], which used I/F-Race [1, 11] to deal with the automatic design and configuration of parameters to obtain a multi-objective ant colony optimization algorithm. In order to use an automatic configuration tool, we created a framework of design algorithmic components from which new variants of Ant-Miner$_{\text{MA}}$ could be generated.

On a high level, Ant-Miner$_{\text{MA+G}}$ starts with an empty list of rules and iteratively adds the best rule found along the iterative process while the number of uncovered training examples is greater than a maximum uncovered value. It uses the same rule creation loop as Ant-Miner$_{\text{MA}}$. Algorithm 1 shows the high-level pseudocode of the Ant-Miner$_{\text{MA+G}}$ algorithm.

Table 1: Algorithmic components of the proposed Ant-Miner$_{\text{MA+G}}$.

Design components	
Ordinal attributes:	1: Using ordinal with range condition 2: Using Ordinal without range condition 3: Not using ordinal
Operator selection:	2: Using archive for sampling conditions 1: Using graph for choosing conditions
Categorical attributes:	1: Archive sampling 2: Archive sampling and not equal condition 3: Using graph for choosing categorical value
Prune Quality Function:	See Table 2
Selection Quality Function:	See Table 2
Pheromone limits:	1: Max-Min limits 2: No limits
Archive top rule updates graph pheromone model:	1: First iteration after the archive is created 2: At the end of each iteration 3: Never updates
Updating graph pheromone model with:	1: Best iteration rule 2: All rules added to the archive
Value used to update graph pheromone model:	1: Weight of the rule in the archive 2: Quality of the rule
Pheromone restart:	1: Restart both the pheromone models 2: No restart

The following subsections present the different design approaches that were implemented in Ant-Miner$_{\text{MA+G}}$—a summary of the algorithmic components are shown in Table 1. We divided the design approaches into several algorithmic components and grouped them into three main categories: (1) rule construction; (2) pheromone model; and (3) quality function configurations.

3.1 Rule Construction

A crucial design decision when combining both pheromone models is defining each pheromone model's contribution in solution creation. This problem is exacerbated by the fact that there are two different ways to create a solution and we want to find the optimal approach between both construction methods. The following sections will show different algorithmic approaches for operator, ordinal and categorical attributes selection. Note that we do not consider using a graph-based model to select continuous attributes values, since this would involve using a discretisation procedure.

The basic framework of Ant-Miner$_{\text{MA+G}}$ consists of a fully connected construction graph. Let a_i be an attribute, $i = 1, ..., n$ where n is the number of attributes; each attribute is added as a node (a_i) to the graph. Suppose an ant l is generating a rule r_l. It starts with an empty rule at node i and probabilistically chooses to visit a node j based on the amount of pheromone on the edge E_{ij}. For the value selection, we will always use the archive model to sample the continuous value using a continuous sampling procedure on the subset of the archive rules that a term enabled using the same attribute and operator.

3.1.1 Ordinal attributes. Ant-Miner$_{\text{MA+G}}$ framework implements a procedure for ordinal attribute, where it uses the continuous sampling procedure from ACO$_{\text{MV}}$ on the indexes of the ordinal values. This was based on ACO$_{\text{MV}}$ approach to benefit from the natural order of the ordinal attribute values. The possible conditions used for Ant-Miner$_{\text{MA+G}}$ ordinal attributes are $\{a_i \leq v, a_i > v, v_1 < a_i \leq v_2\}$—the latter is referred to as RANGE. Ant-Miner$_{\text{MA+G}}$ framework implements three possible approaches to handle ordinal attributes:

1. Sampling from three possible operators {≤, >, RANGE};
2. Sampling from two possible operators {≤, >};
3. Handling ordinal attributes as categorical attribute without any special treatment—i.e., conditions are always in the form $a_i = v$.

3.1.2 Categorical attributes. Ant-Miner$_{\text{MA+G}}$ implements a procedure for categorical attribute, where it use the discrete sampling procedure from ACO$_{\text{MV}}$ on the indexes of the categorical value on the archive rules. The only possible condition used for Ant-Miner$_{\text{MA+G}}$ categorical attributes is $a_i = v$. Ant-Miner$_{\text{MA+G}}$ framework implements three possible approaches to handle categorical attributes:

1. The archive pheromone model is used to sample the value;
2. The archive pheromone model is used to sample the value and only of two possible operators {=, ≠};
3. The graph pheromone model is used for categorical values—each categorical node has the form (attribute, =, value) and there is a node for each value in the domain of the attribute (refer to section 2.1).

3.1.3 Operator configurations. Ant-Miner$_{\text{MA+G}}$ uses the archive to sample operators using a discrete sampling procedure. The framework has two possible approaches to handle operator selection:

1. The archive pheromone model is used to select the operator according to the attribute type, similar to Ant-Miner$_{\text{MA}}$ procedure;
2. The graph pheromone model is used to select the operator. In this case, each node of the graph consists of a pair (attribute, operator). Let a_c be a categorical attribute, each attribute will be associated with the equal operator and added as a node to the graph, such as (a_c, =). Let a_r be a continuous attribute, each attribute will be associated with three operators and three nodes will be added to the graph, such as (a_r, ≤), (a_r, >) and (a_r, RANGE). Finally, let a_o be an ordinal attribute, each attribute will be associated with two operators and two nodes will be added to the graph, such as (a_o, ≤) and (a_o, ≥).

Table 2: Rule qualify function used for pruning and selection procedures.

Functions	Parameter
Precision (P) $\frac{TP}{TP+FP}$	-
Confidence Coverage (CC) $\frac{TP}{TP+FP} + \frac{TP}{SM}$	-
Cost measure (CM) $(c \times TP) - ((1 - c) \times FP)$	$c = 0.437$
Fmeasure (FM) $\frac{(1+\beta^2) \cdot \frac{TP}{TP+FN} \cdot \frac{TP}{TP+FP}}{\beta^2 \cdot \frac{TP}{TP+FN} \cdot \frac{TP}{TP+FP}}$	$\beta = 0.5$
Jaccard (J) $\frac{TP}{TP+FP+FN}$	-
Klosgen (K) $(\frac{TP+FP}{S})^{\omega} . (\frac{TP}{TP+FP} - \frac{TP+FN}{S})$	$\omega = 0.4323$
Laplace (L) $\frac{TP+1}{TP+FP+k}$	$k = \textit{number of classes}$
MEstimate (ME) $\frac{TP+m.\frac{TP}{S}}{TP+FP+m}$	$m = 22.466$
Relative Cost Measure (RCM) $cr \times \frac{TP}{TP+FN} - (1 - cr) \times \frac{FP}{TN+FP}$	$cr = 0.342$
Precision Inverted (PI) $\frac{(FP+TN)-FP}{(S)-(TP+FP)}$	-
MEstimate Inverted (MEI) $\frac{(FP+TN)+m\times\frac{TP+FN}{S}}{(S)-(TP+FP+m)}$	$m = 22.466$
Laplace Inverted (LI) $\frac{(FP+TN)-FP+1}{(S)-(TP+FP+k)}$	$k = \textit{number of classes}$
Sensitivity and Specificity(SS) $\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}$	-

3.2 Quality Function Configurations

The quality function configurations in the rule creation process typically represent a trade off between consistency and coverage—i.e., they prefer rules that cover as few negative and as many positive instances as possible [3, 6]. A quality function is used in two different places in this process: (i) in evaluating rule refinements in the pruning process, where it bias the selection of refinements of the current rule to be explored; (ii) in rule evaluation, where it bias the selection of the rules that are added to the list of rules.

Stecher et al. [20] argued that these tasks should be treated separately and be evaluated with separate functions. Rule refinements function in Ant-Miner$_{\text{MA+G}}$ are used in the pruning procedure, while rule selection functions are used in the archive sorting and selection of rules to added to the model. We implemented thirteen different functions, presented in Table 2. For the parametric rule quality functions, we used the default parameter values proposed in [6]—these are shown in the 'Parameter' column in Table 2. We use a series of shorthand to condense the equations, as below:

- **TP** (**T**rue **P**ositives): The number of instances covered by a rule that belong to the class predicted by the rule;

Table 3: I/F-Race configurations of Ant-Miner$_{\text{MA+G}}$, where the starred configurations values are found in Table 1.

Configurations	AntMinerMA					
	1	2	3	4	5	6
Local Search	0.893	0.412	0.487	0.118	0.112	0.964
Convergence	0.109	0.715	0.231	0.754	0.216	0.808
Archive Size	47	105	9	54	108	12
Max Iterations	1648	757	1442	719	704	1820
Uncovered Instances by model	16	10	14	19	7	9
Minimum covered by rule	11	10	15	10	17	8
Ant Colony Size	26	9	86	36	34	74
Stagnation Limit	26	78	32	82	15	18
Prune Quality	J	CC	CM	LI	SS	J
Selection Quality	SS	SS	SS	SS	SS	SS
Ordinal attributes*	1	2	3	2	1	1
Conditions*	1	1	1	1	1	1
Categorical Attributes*	2	3	2	2	3	2
Top Rule updating the Graph Pheromone*	2	2	1	3	2	2
Updating the Graph Pheromone*	1	2	2	1	1	2
The Value Graph Pheromone Model*	1	1	1	1	1	1
Restart*	2	2	2	2	2	2
Pheromone Type*	2	2	2	1	2	2
Pheromone Initial	NA	NA	NA	2.491	NA	NA
Evaporate Factor	NA	NA	NA	0.809	NA	NA
Pheromone Best	NA	NA	NA	0.917	NA	NA

Table 4: Training datasets.

Dataset	Size	Ordinal	Categorical	Continuous
ionosphere	351	0	0	34
dermatology	366	33	0	1
cylinder-bands	540	2	14	19
annealing	898	0	29	9
credit-g	1000	11	2	7
MiceProtein	1080	0	3	77
HillValley	1212	0	0	100
eb	45781	0	1	2
adult	48842	0	8	6
SkinNonSkin	245057	0	0	3

Table 5: Testing datasets.

Dataset	Size	Ordinal	Categorical	Continuous
breast-tissue	106	0	0	9
iris	150	0	0	4
wine	178	0	0	13
parkinsons	195	0	0	22
glass	214	0	0	9
breast-l	286	4	5	0
heart-h	294	3	3	7
heart-c	303	3	3	7
liver-disorders	345	0	0	6
breast-w	569	0	0	30
balance-scale	625	4	0	0
credit-a	690	4	4	6
pima	768	0	0	8
MolecularBiology	3189	0	60	0
ChoralsHarmony	5665	0	13	1
Mushroom	8124	0	22	0
PenDigits	10992	0	0	16
Magic	19020	0	0	10
CardClients	30000	7	2	14
Nomao	34465	0	29	89
bank-additional	41188	0	10	10
connect4	67557	0	42	0
diabetes	101766	2	34	11
ForestType	581012	0	44	10
PokerHand	1025010	5	0	5

- **FP** (**F**alse **P**ositives): The number of instances covered by a rule that do not belong to the class predicted by the rule;
- **TN** (**T**rue **N**egatives): The number of instances not covered by a rule that do not belong to the class predicted by the rule;
- **FN** (**F**alse **N**egatives): The number of instances not covered by a rule that belong to the class predicted by the rule;
- **S** (TP + FP + TN + FN): The total number of training instances.

3.3 Pheromone Models Configurations

There are three configurations regarding how the pheromone models are used in Ant-Miner$_{\text{MA+G}}$.

3.3.1 Graph Pheromone Model. Ant-Miner extensions used different graph pheromone update procedures. One of the most common update procedures based on the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [21]. In the Ant-Miner$_{\text{MA+G}}$ framework, there are two approaches for updating the graph pheromone model:

1. $\mathcal{MM}$AS, where the parameters are shown in Table 8;
2. Ant-Miner, where the pheromone associated with each term occurring in the rule created by an ant is increased in proportion to the quality of the rule in question; the pheromone associated with each term that does not occur

Table 6: Average predictive accuracy (*average* [*standard deviation*]) over 15 runs of tenfold cross-validation. The last row of the table shows the average rank of the algorithm. The best value of a given dataset is shown in bold.

	Ant-Miner$_{MA+G}$							
Dataset	1	2	3	4	5	6	Ant-Miner$_{MA}$	*c*Ant-Miner
breast-tissue	64.81 [0.94]	63.35 [0.83]	**66.07 [0.76]**	65.83 [0.72]	65.84 [0.63]	66.93 [0.68]	60.24 [0.97]	64.24 [0.24]
iris	94.98 [0.22]	94.80 [0.27]	94.40 [0.26]	95.16 [0.20]	95.02 [0.17]	**95.33 [0.13]**	93.60 [0.31]	94.27 [0.11]
wine	92.61 [0.50]	93.05 [0.39]	92.00 [0.32]	92.69 [0.39]	92.72 [0.49]	93.34 [0.47]	90.78 [0.40]	**93.52 [0.07]**
parkinsons	83.77 [0.58]	86.06 [0.49]	84.71 [0.72]	82.40 [0.57]	84.39 [0.48]	84.09 [0.57]	**86.29 [0.64]**	85.22 [0.40]
glass	67.52 [0.39]	**68.72 [0.56]**	65.04 [0.57]	66.42 [0.66]	63.15 [0.77]	68.27 [0.49]	63.24 [0.50]	59.18 [0.32]
breast-l	71.95 [0.39]	73.73 [0.42]	72.24 [0.26]	72.94 [0.41]	70.23 [0.31]	72.54 [0.38]	71.46 [0.34]	**76.17 [0.11]**
heart-h	60.90 [0.39]	61.81 [0.30]	60.08 [0.38]	59.47 [0.48]	59.67 [0.43]	60.39 [0.51]	64.37 [0.29]	**64.81 [0.33]**
heart-c	55.59 [0.53]	55.99 [0.30]	55.97 [0.25]	55.64 [0.38]	55.99 [0.39]	55.36 [0.36]	56.94 [0.54]	**57.42 [0.32]**
liver-disorders	61.44 [0.48]	63.76 [0.61]	**64.41 [0.53]**	62.47 [0.41]	62.97 [0.22]	61.89 [0.60]	63.13 [0.49]	62.26 [0.18]
breast-w	93.58 [0.26]	93.22 [0.33]	93.29 [0.22]	91.67 [0.20]	93.75 [0.22]	93.48 [0.19]	93.53 [0.22]	**94.28 [0.11]**
balance-scale	73.56 [0.31]	74.38 [0.28]	73.30 [0.23]	73.34 [0.32]	74.97 [0.35]	74.00 [0.32]	**80.10 [0.22]**	68.34 [0.08]
credit-a	85.29 [0.19]	85.59 [0.15]	85.08 [0.18]	85.10 [0.20]	85.30 [0.16]	85.01 [0.17]	85.19 [0.22]	**85.74 [0.11]**
pima	73.81 [0.39]	73.51 [0.27]	74.45 [0.20]	72.80 [0.30]	73.63 [0.37]	72.75 [0.42]	**75.30 [0.21]**	67.45 [0.07]
MolecularBiology	84.09 [0.37]	69.92 [0.65]	83.46 [0.39]	79.46 [0.58]	83.66 [0.49]	**84.52 [0.36]**	56.09 [0.20]	81.31 [0.74]
ChoralsHarmony	61.42 [0.12]	60.44 [0.12]	**62.18 [0.11]**	61.31 [0.12]	60.16 [0.13]	62.51 [0.10]	64.14 [0.15]	60.66 [0.11]
Mushroom	97.46 [0.15]	97.05 [0.20]	98.89 [0.08]	96.82 [0.23]	93.98 [0.17]	98.52 [0.10]	**99.71 [0.02]**	95.10 [0.27]
PenDigits	82.15 [0.16]	81.07 [0.31]	**86.18 [0.23]**	79.21 [0.25]	85.76 [0.15]	86.28 [0.14]	71.56 [0.30]	56.92 [0.03]
Magic	80.65 [0.06]	81.35 [0.14]	81.74 [0.08]	80.10 [0.19]	81.31 [0.06]	80.61 [0.17]	**82.67 [0.05]**	70.41 [0.01]
CardClients	81.42 [0.12]	81.18 [0.10]	81.44 [0.04]	80.82 [0.12]	81.55 [0.05]	81.07 [0.11]	**81.80 [0.02]**	80.63 [0.00]
Nomao	88.74 [0.28]	89.76 [0.26]	**90.99 [0.23]**	86.84 [0.28]	88.68 [0.26]	90.77 [0.10]	87.77 [0.05]	90.66 [0.02]
bank-additional	90.60 [0.02]	90.32 [0.03]	**90.74 [0.02]**	90.41 [0.02]	90.57 [0.04]	90.62 [0.03]	89.49 [0.02]	89.87 [0.01]
connect4	67.90 [0.04]	67.42 [0.06]	**69.51 [0.08]**	67.72 [0.06]	67.49 [0.04]	68.37 [0.06]	68.18 [0.02]	67.83 [0.01]
diabetes	56.01 [0.03]	54.09 [0.03]	55.87 [0.05]	**56.10 [0.03]**	54.24 [0.05]	55.89 [0.07]	55.92 [0.09]	54.17 [0.13]
ForestType	68.92 [0.17]	67.25 [0.27]	**69.51 [0.53]**	67.02 [0.21]	69.09 [0.21]	69.15 [0.33]	68.38 [0.07]	63.07 [0.07]
PokerHand	50.23 [0.01]	50.25 [0.01]	51.39 [0.04]	51.56 [0.04]	51.74 [0.03]	50.24 [0.00]	**52.01 [0.04]**	50.20 [0.00]
Average Rank	4.52	4.6	**3.68**	5.52	4.44	3.88	4.08	5.28

in the rule is decreased by normalizing all the pheromones values after the update.

3.3.2 Updating Pheromone Models. The level of interaction between the two pheromone models could range from no interaction at all, to close interaction between them, as below:

(i) Archive top rule updates graph pheromone model:
  (1) In the first iteration after the archive is created;
  (2) At the end of each iteration;
  (3) Never updates the graph pheromone.
(ii) The graph pheromone model is updated with:
  (1) The best iteration rule;
  (2) All rules that have been added to the archive.
(iii) The value used to update the graph pheromone model:
  (1) The weight of the rule in the archive;
  (2) A value proportional to the quality of the rule.

3.3.3 Restart procedure. The restart procedure resets both pheromone models to the start point without forgetting the best-so-far solution in the archive. It is used to avoid premature stagnation of the algorithm. The reset procedure is triggered (only once) by observing a number of consecutive iterations without improvement on the quality of the best rule so far. It works by randomly initializing the archive and resetting graph pheromone values to their initial value.

4 COMPUTATIONAL RESULTS

Our computational experiments were computed using 35 publicly available dataset from the UCI Machine Learning Repository [8]. The datasets were divided into two sets: a training set (shown in Table 4) and a testing set (shown in Table 5).

In the first part the experiments, our goal is automatically design a better variant for Ant-Miner$_{MA}$ algorithm using the proposed configurable framework Ant-Miner$_{MA+G}$ discussed in Section 3 using automatic configuration method I/F-Race. I/F-Race is a state-of-the-art automatic configuration method to deal with continuous, categorical, and discrete parameters. I/F-Race generates new candidate configurations and performs races to discard the worst-performing ones. Within a single race of I/F-Race, candidate configurations run on one instance at a time and a Friedman test followed by a post-test analysis is applied to discard configurations that show a sufficient statistical evidence that they perform worse than the remaining ones. After only a small number of configurations remain in the race, the race stops. A new race starts with the best configurations previously found and with new candidate configurations generated from the best configurations using a simple probabilistic model. The automatic configuration process stops after reaching a given maximum budged, usually specified as a maximum number of runs or a time limit.

López-Ibáñnez and Stutzle [12] showed that fully configuring the design components and ACO parameter settings has an advantage. This may be due to interaction between the design components and

Table 7: Average runtime over 15 runs of tenfold cross-validation. The last row of the table shows the average rank of the algorithm. The best value of a given dataset is shown in bold.

	Ant-Miner$_{MA+G}$							
Dataset	1	2	3	4	5	6	Ant-Miner$_{MA}$	*c*Ant-Miner
breast-tissue	1.19	1.34	1.41	1.97	1.36	0.86	**0.38**	0.67
iris	0.48	0.94	0.86	1.04	0.93	0.75	**0.28**	0.49
wine	1.13	1.19	1.13	1.46	1.16	0.85	**0.56**	**0.56**
parkinsons	1.18	1.51	1.68	1.73	1.58	1.36	**0.78**	2.87
glass	1.51	2.11	2.15	3.09	1.93	2.00	**0.50**	2.66
breast-l	1.17	1.12	1.82	1.90	1.55	1.11	**0.54**	1.28
heart-h	1.66	2.37	3.02	3.70	2.33	2.47	**0.72**	12.61
heart-c	2.05	2.60	3.21	3.44	2.60	3.03	**0.76**	10.91
liver-disorders	1.36	1.87	2.21	2.59	1.81	1.52	**0.47**	1.81
breast-w	3.04	2.64	4.25	4.07	3.31	2.64	**2.31**	5.40
balance-scale	1.12	1.33	2.12	2.09	1.55	1.34	0**.50**	5.95
credit-a	2.44	2.24	4.43	4.06	3.16	2.56	**1.10**	11.57
pima	2.46	2.59	3.61	3.43	2.74	2.60	**0.93**	3.69
MolecularBiology	23.96	52.34	32.47	23.29	63.77	26.95	**12.50**	345.10
ChoralsHarmony	148.98	102.07	258.18	284.83	100.44	229.80	**11.56**	6320.00
Mushroom	11.08	4.91	25.36	22.28	4.60	19.60	**9.09**	11.56
PenDigits	173.83	107.12	294.15	176.00	164.38	251.87	**23.92**	135.79
Magic	198.34	145.74	199.70	197.71	161.93	158.76	**25.56**	155.07
CardClients	458.64	188.10	830.95	409.17	445.07	546.12	**106.17**	1060.10
Nomao	564.90	**310.67**	846.17	804.15	391.73	433.15	2440.88	779.77
bank-additional	240.03	**121.20**	338.18	307.17	242.78	355.86	185.27	1970.72
connect4	1263.02	**878.05**	4304.96	1405.23	1475.85	2950.35	1259.82	14380.04
diabetes	4043.66	4723.35	**7208.30**	2470.67	9669.81	6363.74	4374.27	313685.51
ForestType	49580.33	**29274.07**	72685.69	35223.81	64891.29	59500.34	40367.47	52746.55
PokerHand	6109.91	3415.63	11183.90	10218.68	6755.00	7442.81	**2647.72**	27872.59
Average Rank	3.32	3.32	6.6	5.92	4.6	4.4	**1.6**	6.24

ACO parameter settings. We therefore followed a similar approach, where I/F-Race optimises both the design components and ACO parameter settings—these are shown in Table 8. The configuration budget is set to 10000 runs. We perform five independent repetitions of the configuration process using the accuracy as the evaluation criterion. The best configuration found in each of the five runs were then used as seed candidates for a final I/F-Race configuration process. Therefore, we created six different configurations through six independent I/F-Race processes. The datasets used by the I/F-Race is shown in Table 4. It is important to note that the testing datasets are not used to evaluate the configurations.

The six configurations found by the independent runs of I/F-Race are shown in Table 3. The configuration values are presented in Table 1 and the keys are used in Table 3 to describe the configurations found by I/F-Race.

The resulting configurations did show the impact of using a graph pheromone model, since the option of sampling operators using the graph was used in every winning configuration. This provides evidence for our first assumption that the graph pheromone model works well with nominal values. Categorical attributes showed interesting results as two options surfaced: (1) using the graph to select the categorical value, which is the expected behaviour; and (2) sampling from archive was used when we added the not equal (≠) operator. Also, using the value of the rule weight of the archive proved to produce better configurations. The configuration of the quality functions showed an interesting behaviour. While different functions were considering when pruning rules, the sensitivity and specificity dominated the configuration for evaluating rules for selection, providing a good indication of the benefit of using this function—it is the same function used in the original Ant-Miner.

Those six configurations are evaluated on the testing datasets (shown in Table 5) by running them 15 times in a tenfold cross-validation (a total of 150 individual runs) on every dataset. The average results are shown in Table 6. For comparison, we ran the previous version of Ant-Miner$_{MA}$ [4] and *c*Ant-Miner [16]. We also measured the average runtime of the algorithms, shown in Table 7.

Two of configurations of Ant-Miner$_{MA+G}$ showed a higher average ranking than the original Ant-Miner$_{MA}$, while most of the configurations showed a higher average rank than *c*Ant-Miner. One of the interesting datasets to look at is Nomao, which is one of the largest datasets with 34465 instances and 118 attributes: *c*Ant-Miner achieved a 90.66% accuracy, higher than Ant-Miner$_{MA}$ (87.77% accuracy); the runtime of *c*Ant-Miner was 779 seconds, better than Ant-Miner$_{MA}$(2440 seconds). This case was noted in [4], where it was believed that the number of attributes did affect the Ant-Miner$_{MA}$ performance. Notably, the proposed Ant-Miner$_{MA+G}$ framework did show an improvement in the configuration Ant-Miner$_{MA+G}$ (3), where it achieved a better performance (90.99% accuracy) and a runtime of 846 seconds, which is still lower than Ant-Miner$_{MA}$.

We performed a Wilcoxon signed rank test (shown in Table 9) to compare the accuracy results of Ant-Miner$_{MA+G}$ (3), Ant-Miner$_{MA}$

Table 8: Range of parameter values in Ant-Miner$_{\text{MA+G}}$.

Parameters	range of Values
Local search	[0.001, 1]
Convergence	[0.001, 1]
Archive size	[5, 150]
Maximum iterations	[500, 2500]
Uncovered instances	[5, 25]
Minimum covered by rule	[5, 25]
Ant colony size	[5, 90]
Stagnation limit	[10, 100]
Initial pheromone	[1, 10]
Evaporate factor	[0.001, 1]
Best pheromone	[0.001, 1]

Table 9: Accuracy comparison according to a Wilcoxon signed rank test. Significant differences at $\alpha = 0.5$ are shown in bold.

Wilcoxon Signed Rank Test	p
Ant-Miner$_{\text{MA+G}}$ (3) *vs* *c*Ant-Miner	**0.0139**
Ant-Miner$_{\text{MA}}$ *vs* *c*Ant-Miner	0.1936

and *c*Ant-Miner. Ant-Miner$_{\text{MA+G}}$ (3) shows a significant improvement in terms of accuracy over *c*Ant-Miner (p value **0.0139**), while the baseline Ant-Miner$_{\text{MA}}$ could not show significant improvement in terms of accuracy over *c*Ant-Miner. The results also shows that the graph pheromone model improved the performance of the Ant-Miner$_{\text{MA+G}}$ algorithm in datasets with large number of attributes.

5 CONCLUSION

In this paper we introduced the concept of combining the graph pheromone model and the archive pheromone model, based on Ant-Miner$_{\text{MA}}$ and *c*Ant-Miner algorithms. The use of the solution archive allows the algorithm to deal with continuous attributes without requiring a discretisation procedure, while using the graph pheromone model improves the performance of algorithm for datasets containing large number of attributes.

Instead of manually designing a new algorithm to combine both pheromone models, we used a fully configurable framework Ant-Miner$_{\text{MA+G}}$ using an automatic design process. I/F-Race, which is a state of the art automatic configuration tool, was used to generate five different configurations for the Ant-Miner$_{\text{MA+G}}$ algorithm. Each one of those automatically designed preformed competitively well against the baseline algorithms.

Our experimental results have shown that such an automatically configured design outperforms the *c*Ant-Miner algorithm to a significant level, and solved the problems Ant-Miner$_{\text{MA}}$ faced when dealing with datasets with a large number of attributes. The automatic framework also provides the flexibility to design an algorithm specific a given dataset. As a future research direction, we would investigate the automatic design of Ant-Miner$_{\text{MA+G}}$ for each dataset individually. We expect that this could further improve the predictive accuracy of the algorithm.

REFERENCES

[1] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. *F-Race and Iterated F-Race: An Overview*, pages 311–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[2] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66, 1997.

[3] J. Furnkranz. *From Local to Global Patterns: Evaluation Issues in Rule Learning Algorithms*, pages 20–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[4] A. Helal and F. Otero. A mixed-attribute approach in ant-miner classification rule discovery algorithm. In *Genetic and Evolutionary Computation Conference (GECCO 2016)*, pages 13–20. ACM Press, April 2016.

[5] N. P. Holden and A. A. Freitas. A hybrid pso/aco algorithm for classification. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '07, pages 2745–2750, New York, NY, USA, 2007. ACM.

[6] F. Janssen and J. Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379, 2010.

[7] T. Liao, K. Socha, M. Montes de Oca, T. Stützle, and M. Dorigo. Ant colony optimization for mixed-variable optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(4):503–518, 2014.

[8] M. Lichman. UCI Machine Learning Repository, 2013. Irvine, CA: University of California, School of Information and Computer Science. [http://archive.ics.uci.edu/ml].

[9] B. Liu, H. Abbas, and B. McKay. Classification rule discovery with ant colony optimization. In *IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003)*, pages 83–88, Oct 2003.

[10] B. Liu, H. A. Abbass, and B. McKay. Density-Based Heuristic for Rule Discovery with Ant-Miner. In *The 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System*, pages 180–184, 2002.

[11] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Caceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[12] M. López-Ibáñez and T. Stützle. The automatic design of multiobjective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875, Dec 2012.

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

[14] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens. Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11(5):651–665, Oct 2007.

[15] F. Otero, A. Freitas, and C. Johnson. *c*Ant-Miner: an ant colony classification algorithm to cope with continuous attributes. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, editors, *Proceedings of the 6th International Conference on Swarm Intelligence (ANTS 2008), Lecture Notes in Computer Science 5217*, pages 48–59. Springer-Verlag, 2008.

[16] F. Otero, A. Freitas, and C. Johnson. Handling continuous attributes in ant colony classification algorithms. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pages 225–231, March 2009.

[17] F. Otero, A. Freitas, and C. Johnson. A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):64–76, 2013.

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

[19] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[20] J. Stecher, F. Janssen, and J. Furnkranz. Separating rule refinement and rule selection heuristics in inductive rule learning. In T. Calders, F. Esposito, E. Hullermeier, and R. Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8726 of *Lecture Notes in Computer Science*, pages 114–129. Springer Berlin Heidelberg, 2014.

[21] T. Stützle and H. H. Hoos. Max-min ant system. *Future Gener. Comput. Syst.*, 16(9):889–914, 2000.

[22] Q. Yang, W. N. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, and J. Zhang. Adaptive multimodal continuous ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 21(2):191–205, 2016.