View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Kent Academic Repository

Kent Academic Repository

Full text document (pdf)

Citation for published version

Helal, Ayah and Brookhouse, James and Otero, Fernando E.B. (2018) Archive-Based Pheromone Model for Discovering Regression Rules with Ant Colony Optimization. In: 2018 IEEE Congress on Evolutionary Computation, 8-13 July 2018, Rio de Janeiro, Brazil. (In press)

DOI

Link to record in KAR

http://kar.kent.ac.uk/67178/

Document Version

Author's Accepted Manuscript

Copyright & reuse

Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (eg Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

Versions of research

The version in the Kent Academic Repository may differ from the final published version.
Users are advised to check **http://kar.kent.ac.uk** for the status of the paper. **Users should always cite the published version of record.**

Enquiries

For any further enquiries regarding the licence status of this document, please contact:
researchsupport@kent.ac.uk

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at **http://kar.kent.ac.uk/contact.html**

KAR Kent Academic Repository

University of Kent

Archive-Based Pheromone Model for Discovering Regression Rules with Ant Colony Optimization

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

James Brookhouse
School of Computing
University of Kent
Chatham Maritime, UK
Email: jb765@kent.ac.uk

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

***Abstract*—In this paper we introduce a new algorithm, called Ant-Miner-Reg$_{\mathrm{MA}}$ to tackle the regression problem using an archive-based pheromone model. Existing regression algorithms handle continuous attribute using a discretisation procedure, either in a preprocessing stage or during rule creation. Using an archive as a pheromone model, inspired by the ACO for Mixed-Variable (ACO$_{\mathbf{MV}}$), we eliminate the need for a discretisation procedure. We compare the proposed Ant-Miner-Reg$_{\mathrm{MA}}$ against Ant-Miner-Reg, an ACO-based regression algorithm that uses a dynamic discretisation procedure, inspired on M5 algorithm, during rule construction process. Our results show that Ant-Miner-Reg$_{\mathrm{MA}}$ achieved a significant improvement in the relative root mean square error of the models created, overcoming the limitations of the dynamic discretisation procedure.**

I. Introduction

Data mining is the automated process of extracting useful and usable patterns from data [1]. The field of data mining comprises of a number of tasks which can be roughly divided into descriptive tasks (e.g., association rule mining, clustering) and predictive tasks (e.g., regression, classification). In this paper we will focus on the task of discovering regression rules. The regression task involves the creation of a model that predicts a continuous dependent attribute based on a number of independent attributes or regressors. Unlike the classification task, which aims to predict the value of a nominal dependent variable (i.e., a variable that has a number of predefined categories), regression aims to predict a continuous value with no predefined categories or boundaries. Similarly to classification rules, regression rules can be represented using the form *IF-THEN*, where the antecedent of the rule represented by *IF* contains logical tests involving the independent attributes, while the *THEN* is the consequent of the rule and makes the rules prediction if the tests in the antecedent are satisfied. When combined as a list, regression rules provide a comprehensible prediction model.

Ant Colony Optimization (ACO) [2] has been adapted to solve data mining problems such as classification, clustering and regression. In most cases, these approaches use a graph-based pheromone model, which is used to guide the ants in a discrete search space. The solution components of the problem are represented by nodes of a graph and ants traverse this graph to produce a solution. Most data mining problems contains continuous attributes, therefore these approaches have to be adapted to handle continuous attributes. The majority of ACO-based algorithms that handle continuous attributes use a discretisation procedure which take place in a preprocessing stage or during rule creation.

Recently, Laio et al. [3] proposed a new approach for ACO-based algorithms to handle mixed variable (continuous, ordinal and discrete) optimisation problems, called Ant Colony Optimization for Mixed Variable (ACO$_{\mathbf{MV}}$). ACO$_{\mathbf{MV}}$ replaces the graph-based pheromone model with an archive-based pheromone model to guide ants in mixed variables search space using different sampling strategies according to the variable type. Helal and Otero [4], [5] presented the first data mining approach, to the best of our knowledge, that used an archive-based pheromone model instead of a graph-based pheromone model for classification problem.

Brookhouse and Otero [6] have successfully used an ACO-based algorithm, called Ant-Miner-Reg, to create regression rules. Ant-Miner-Reg uses a sequential covering strategy to create a rule list using an ACO rule creation procedure with a graph-based pheromone model. Ant-Miner-Reg uses a M5 [7] inspired dynamic discretisation procedure to handle continuous attributes during the rule creation rather than requiring the discretisation of continuous values as a pre-processing step. Ant-Miner-Reg significantly outperformed SeCoReg [8], a greedy sequential covering algorithm, without increasing the average number of terms required to classify an instance.

In this paper we propose the use of an archive-based pheromone model to better handle continuous values in regression problems. By incorporating a similar strategy as ACO$_{\mathbf{MV}}$, different attributes types (categorical and continuous) can be handled directly, without requiring a discretisation procedure. We compared our proposed algorithm against Ant-Miner-Reg in nineteen publicly available datasets and used the Wilcoxon signed-rank test to determine the significance between the difference in performance.

The remainder of this paper is organised as follows. We begin by reviewing the literature of existing regression algorithms, archive-based ACO algorithms, and Ant-Miner-Reg in Section II. Section III present our proposed Ant-Miner-Reg$_{\mathrm{MA}}$ algorithm. The computational results are presented in Section IV, and finally conclusion and direction for future work are discussed in Section V.

II. Background

There are three main areas of related work, existing regression rule learners, the existing archive-based ACO algorithms, and the existing ACO-based algorithm for regression Ant-Miner-Reg.

Conventional regression models take the form of linear and non-parametric equations [9], however, we will be focusing on regression rule learners. One of the classical regression rule learners is M5' Rules [10] which builds on the model tree learner M5 [7]. M5' Rules uses sequential covering to build a list of rules, each iteration of the sequential covering algorithm produces a complete M5 tree which is then flattened into rules and the best rule generated is added to the partial rule list. M5 uses an interesting strategy to cope with continuous attributes: it chooses the split points in its dynamic discretisation step by trying to maximise the expected error reduction. In this case the error is considered to be the standard deviation of the dependent attribute in the generated subsets.

Before Ant-Miner-Reg, the only known swarm intelligence rule miner was Minnaert and Martens' PSOminer [11]. Like M5' Rules, PSOminer uses a sequential covering strategy to generate a rule list however instead of using M5 as the rule generating procedure, a Particle Swarm Optimisation (PSO) is used. The attributes in a dataset are encoded onto a particle as the particles position in the attribute space. PSOminer showed promising initial results however its development was not continued.

Ant Colony Optimization has been successfully used to generate classification rules, most notably Ant-Miner [12] and the suite of algorithms developed as extensions to the original. While the majority use a graph-based model, an archive-based model has been successful in creating classification rules.

Ant-Miner$_{\mathrm{MA}}$ proposed by Helal and Otero [4] is the first Ant-Miner classification algorithm to use an archive-based pheromone model instead of a graph-based pheromone model to the best of our knowledge. The archive was used to sample conditions to create rules, instead of ants traversing a construction graph. This approach showed competitive results compared to cAnt-Miner, a graph-based ACO classification algorithm. The archive-based pheromone model significantly improved the runtime, since it eliminated the need for a discretisation procedure. cAnt-Miner uses a Minimum Description Length (MDL) procedure as a discretisation procedure, which is proposed by Otero et al. [13]. The MDL procedure was used to find the best possible split point, with respect to the class value of the rule. This process became expensive in terms of runtime when the number of instances available increased in a dataset. Ant-Miner$_{\mathrm{MA}}$ removed this expensive process with a faster implementation of the archive, where the archive would be used to find the split points instead, improving the runtime in classification problems while not compromising the accuracy. One limitation of Ant-Miner$_{\mathrm{MA}}$, was as the number of attributes increased over 50, the runtime increased compared to cAnt-Miner.

An automatic algorithm design approach for Ant-Miner$_{\mathrm{MA+G}}$ [5] was created to overcome this problem, which combines the graph-based and archive-based pheromone models. This automatically configured algorithm outperformed the original cAnt-Miner algorithm to a significant level, and solved the problems Ant-Miner$_{\mathrm{MA}}$ faced when dealing with a large number of attributes. The improvement came from the graph-based pheromone model which allowed the algorithm to quickly identify irrelevant attributes and ignore them during the rule creation process.

Brookhouse and Otero introduced the first Ant-Miner algorithm for regression, Ant-Miner-Reg [6]. Ant-Miner-Reg uses the same sequential covering approach adopted by Ant-Miner and couples this with the dynamic discretisation procedure used in Quinlan's M5 [7].

Ant-Miner-Reg creates a rule list as follows. First n rules are created by the colony, where each ant traverses a graph of attribute nodes and values to build the antecedent of a rule. If the ant discovers a node representing a continuous attribute a value is generated via a dynamic discretisation method that attempts to find the optimal split points for that attribute in the set of uncovered instances. When the antecedent of a rule is created the prediction is generated by the calculating the mean value of any instances covered by the new rule. Once all the rules are created the best rule generated is used to update the pheromone matrix of the colony. This procedure is repeated until the maximum number of iterations is reached, at which point the best rule is then returned and added to the list of rules under construction removing any newly covered instances from the dataset. The colony is then reset and the ACO process repeated on the still uncovered set of instances until all instances are covered by the rule list.

Ant-Miner-Reg generated comprehensive regression rules that significantly outperformed SeCoReg [8]. They identified the need to include a better continuous attribute processing technique, which would enable the optimisation of the numeric values chosen by fully integrating them inside the pheromone matrix [6].

III. Archive-based Ant-Miner-Reg

As discussed in Section II, most Ant Colony Optimization approaches to create rules are based on a graph pheromone model, which can directly cope with categorical attributes while continuous attributes require either a pre-processing or dynamic discretisation step. Liao et al., [3] introduced a new algorithm, called Ant Colony Optimization for Mixed-Variable (ACO$_{\mathbf{MV}}$), to deal with mixed-variable optimisation problems. ACO$_{\mathbf{MV}}$ uses an archive-based pheromone model and sampling procedures to create a new solution, allowing the algorithm to cope directly with categorical or continuous (real-valued) attributes. The archive-based pheromone model is implemented as a solution archive (A), which contains previously generated k best solutions, to derive a probability distribution to bias the search. Each ant starts generating a new candidate solution. During the construction of a solution, a probabilistic solution construction method is used to sample new values from the archive according to the type of each

Algorithm 1: High-level pseudo code of Ant-Miner-Reg$_{\mathrm{MA}}$

Data: training data
Result: list of rules

```
1  RuleList ← {}
2  while |TrainingData| < MaxUncovered do
3      A ← Generate Random Rules
4      while t < MaxIterations and not Restarted do
5          A_t ← {}
6          while i ¡ number of ants do
7              R_i ← Create New Rule
8              R_i ← Prune(R_i)
9              R_i ← Set Consequent(R_i)
10             i ← i + 1
11             A_t ← R_i
12         end
13         A ← UpdateArchive(A_t)
14         t ← t + 1
15         if stagnation() then
16             Restart(A)
17             Restarted ← True
18         end
19         if stagnation() and Restarted then
20             Break
21         end
22     end
23     R_best ← BestRule(A)
24     RuleList ← RuleList + R_best
25     TrainingData ← TrainingData − covered(R_best)
26 end
27 return RuleList
```

attribute. After *m* (colony size) solutions are created, they are added to the archive and the archive is then sorted. At the end of an iteration, the best *k* solutions are selected and a new iteration starts.

The proposed Ant-Miner-Reg$_{\mathrm{MA}}$ algorithm uses ACO$_{\mathbf{MV}}$ pheromone model and search procedure to sample terms to create regression rules. The high level pseudo code of Ant-Miner-Reg$_{\mathrm{MA}}$ is shown in Algorithm 1. Ant-Miner-Reg$_{\mathrm{MA}}$ starts with an empty list of rules (line 1). At each iteration (lines 3 -25), a single rule is created. The rule creation process starts by initialising the archive with k randomly generated rules (line 3). At each iteration m new rules (lines 6-12) are generated, where m is the number of ants in the colony. Rules are added to the archive (line 12), and $k+m$ rules are sorted. The worst m rules are removed from the archive, limiting the archive to k best rules found so far. The procedure to create new rules is repeated until the maximum number of iterations has been reached or stagnation. Stagnation is the failure of the algorithm to find better rules for a number of iterations. In the first case of stagnation, a restart procedure is applied; if the algorithm reaches stagnation for a second time, the rule creation procedure stops.

A. Rule Structure

A rule R consists of an n-dimensional terms vector, where n is the number of attributes in the dataset. Each term t_i, for $i \in \{1, n\}$ in a R contains a flag to indicate if this term is enable or not, an operator and value. For continuous attribute terms, the operator can be either $\leq$ or $>$, representing conditions where the term's attribute value is $\leq x$ or $> x$, where x is a real value. Categorical attribute terms have a single operator $=$, representing conditions where the term's attribute value is $= y$, where y is a value in the domain of the nominal attribute.

The consequent of a rule—the rule's prediction—is a real value, calculated as the mean value of the instances covered by this rule in the training data.

B. Rule Quality

The quality of a regression rule is based on two factors, the first is the quality of the prediction measured using a Relative Root Mean Squared Error (RRMSE). The RRMSE of a rule is defined as

$$L_{RRMSE} = \frac{L_{RMSE}}{\sqrt{\frac{1}{m} L_{default}}} \tag{1}$$

where L_{RMSE} is the root mean square error and $L_{Default}$ is a normalising factor that will approximately bound the RRMSE between 0 and 1. Both L_{RMSE} and $L_{Default}$ are defined as

$$\begin{aligned} L_{RMSE} &= \sqrt{\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i - \bar{y_i})^2} \\ L_{default} &= \sum_{i=1}^{m} (y_i - y')^2 \end{aligned} \tag{2}$$

m is the total number of instances in the dataset, y is the value of the current instance, $\bar{y}$ is the predicted value of the current instance and y' is the mean over all instances.

The RRMSE approximately normalises the RMSE of a rule between 0 and 1, where a value less than 1 corresponds to a rule making a prediction better than the uncovered instances mean and a value greater than 1 is worse than the mean.

The second factor is a measure of how generalised the rule is, i.e., number of instances covered by the rule. Like RRMSE, the coverage of a rule is normalised so that 0 represents a rule covering no instances and 1 is a rule that covers all of the instances in the dataset. The relative coverage of a rule R is defined as

$$relCov = \frac{1}{m} \cdot coverage(R) \tag{3}$$

Both the RRMSE and relative coverage are combined into a single metric Q, which is used as a rule's quality, defined as

$$Q = \alpha \cdot (1 - L_{RRMSE}) + (1 - \alpha) \cdot relCov \tag{4}$$

where α sets the weighting between RRSME and relative coverage. Varying α between 0 and 1 will bias the rule quality towards either RRMSE or relative coverage.

C. Archive Structure and Initialisation

The archive consist of k rules sorted by their quality Q, so that $Q(R_1) \geq Q(R_2) \geq \ldots \geq Q(R_k)$. Each rule (solution) j is associated with a weight ω_j that is related to $Q(R_j)$, where ω_j is calculated using a Gaussian function given by

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(rank(j)-1)^2}{2q^2k^2}} \tag{5}$$

where q is used to control the influence of the top-ranked rules on the construction of a new rule. When a new rule is created, it probabilistically samples values around the rules with higher weights.

The archive is initialised with k random rules. Initialisation begins by randomly enabling each term in the vector of allowed terms. These enabled terms are then initialised according to their types.

If the term is continuous, then an unbiased random probability is used to set the operator from the set $\{\leq, >\}$. The value of the continuous term is a random value generated in the range found in the training data for that attribute. For categorical terms, the only operator $=$ is added and the value set randomly to one of the values in the domain of the attribute.

Rules are then pruned to disable irrelevant terms that might be enabled by the stochastic nature of the initialisation. If the number of instances covered by a rule is greater or equal to a user-defined minimum limit, the rule is added to the archive, if it doesn't a new rule is generated instead. Finally, rules are sorted according to their quality.

D. Sampling Procedures

There are two types of sampling procedures used in Ant-Miner-Reg$_{\mathrm{MA}}$ to select values for rule terms: categorical and continuous sampling.

1) Categorical sampling: The categorical sampling is implemented using the same approach as ACO$_{\mathbf{MV}}$. Given a categorical attribute i that has t_i possible values, an ant chooses probabilistically a value v_l^i of the available $\{v_1^i, \ldots, v_{t_i}^i\}$ values. The probability of selecting a value v_l^i is given by

$$p_l^i = \frac{\alpha_l}{\sum_{j=1}^{t_i} \alpha_j} \tag{6}$$

where α_l is the weight associated to each value of the categorical attribute, calculated as

$$\alpha_l = \begin{cases} \frac{\omega_{jl}}{u_l^i} + \frac{q}{\eta} & , \text{if}(\eta > 0, u_l^i > 0) \\ \frac{\omega_{jl}}{u_l^i} & , \text{if}(\eta = 0, u_l^i > 0) \\ \frac{q}{\eta} & , \text{if}(\eta > 0, u_l^i = 0) \end{cases} \tag{7}$$

where ω_{jl} is the weight of the highest rule that uses the value v_l^i for attribute i in the archive, u_l^i is the number of rules that use the value v_l^i for attribute i in the archive ($u_l^i = 0$ corresponds to the special case where v_l^i is not used by the rules in the archive), η is the number of values from t_i that are not used in the archive ($\eta = 0$ corresponds to the special case where all values are used), and q is the same parameter used in Equation (5). The categorical sampling procedure allow an ant to consider two components when sampling a new value. The first component biases the sampling towards values that are used in high-quality rules, but do not occur very frequently in the archive. The second component biases the sampling towards unexplored values in that attribute.

2) Continuous sampling: Continuous sampling implemented using the same approach as ACO$_{\mathbb{R}}$ [14], which is used in ACO$_{\mathbf{MV}}$. First, an ant chooses probabilistically a rule from the archive, before the rule creation process. This rule is used to sample continuous values around it for all continuous attributes. The probability of choosing rule j is given by

$$p_j = \frac{\omega_j}{\sum_{l=1}^{k} \omega_l} \tag{8}$$

where ω_j is the weight associated with the j-th rule in the archive calculated according to Equation (5). Let R_i denote a new solution sampled by ant i around the chosen solution R_j for continuous attribute a, the Gaussian probability density function (PDF) is given by

$$R_{ia} \sim N(R_{ja}, \sigma_{ja}) \tag{9}$$

$$\sigma_{ja} = \xi \sum_{l=1, j \neq l}^{K} \frac{|R_{la} - R_{ja}|}{K - 1} \tag{10}$$

where R_{ja} is the value of the variable a in the selected rule j of the archive, σ_{ja} is the average distance between the value of the variable a in the rule j and the value of a in all the other rules in the archive (given by Equation 10), and ξ is a user-defined value representing the convergence speed of the algorithm.

E. Rule Creation

Rule creation starts by choosing probabilistically whether to include each term or not. The decision is handled using a categorical sampling to choose a $\{$`TRUE`, `FALSE`$\}$ value. If the term is enabled (`TRUE` value), we set the operator according to the attribute type. If the attribute is categorical, it is set to $=$. If it is continuous, the decision is handled using a categorical sampling to choose an operator from the set $\{\leq, >\}$, with the only difference being only the subset of rules that have this term enabled are considered in Equation (7).

The value of the new rule's term is then sampled. If the term is continuous, we use the continuous sampling procedure only considering the subset of rules that have this term enabled and use the same operator as the new term. If the attribute is categorical, we use the categorical sampling procedure only considering the subset of rules that have this term enabled.

After a term is created and added to the partial rule, we apply the rule to the training data. If the number of instances

TABLE I
PARAMETERS: ANT-MINER-REG$_{\mathrm{MA}}$ USES THE FIRST THREE PARAMETERS IN TABLE, WHILE REMAINING ARE USED BY BOTH ANT-MINER-REG$_{\mathrm{MA}}$ AND ANT-MINER-REG.

Parameters	Value
q	0.025495
ξ	0.6795
R	90
Minimum Covered	10
Max Uncovered	10
Max Iterations	1500
Number of Ants	60
Stagnation Test	10
α	0.59

covered by the rule after the addition of the new term is less than the minimum covered instances, the term is disabled. This process is repeated until all terms are considered.

Finally, a local search procedure is applied. The local search procedure is inspired by the *threshold-aware* pruner found in [13]. Firstly, the quality of the rule is calculated according to Equation (4). Then, the last term is then disabled and the quality re-calculated. If the quality of the (pruned) rule decreases, the term is re-enabled and the procedure stops; otherwise, the procedure is repeated until a decrease in quality is observed.

IV. RESULTS

We compared our proposed algorithm Ant-Miner-Reg$_{\mathrm{MA}}$ against Ant-Miner-Reg. The experiments are conducted using nineteen regression datasets publicly available from the UCI Machine Learning Repository [15]—details are shown in Table II. Ant-Miner-Reg$_{\mathrm{MA}}$ uses the first three parameters in Table I for the archive setting, while the remaining parameters are used by both algorithms. We ran both algorithms for five times with ten-fold cross-validation for a total of fifty runs each dataset and reported the average performance of the models produced by each algorithm—shown in Table III in terms of relative root mean square error (RRMSE). For statistical significance testing of the difference in RRMSE, we used Wilcoxon signed-rank test. The result of the statistical testing is shown in Table IV.

As shown in Table III, Ant-Miner-Reg$_{\mathrm{MA}}$ shows an improvement in RRMSE compared to Ant-Miner-Reg, outperforming Ant-Miner-Reg in sixteen of the nineteen datasets. Most notably, Ant-Miner-Reg$_{\mathrm{MA}}$ improved the RRMSE by 80% in the Yacht dataset: Ant-Miner-Reg's RRMSE is 1.0120 while Ant-Miner-Reg$_{\mathrm{MA}}$'s RRMSE is 0.2091. Based on our results, it is clear that the introduction of archive-based pheromone model in Ant-Miner-Reg$_{\mathrm{MA}}$ resulted in an improvement in the model creation. Ant-Miner-Reg uses the M5 dynamic discretisation procedure when creating terms for continuous attributes, while Ant-Miner-Reg$_{\mathrm{MA}}$'s archive-based pheromone model is responsible for generating and improving the values chosen for the continuous attributes terms.

TABLE II
NUMBER OF INSTANCES AND ATTRIBUTE MAKEUP OF THE NINETEEN DATASETS USED IN THE EXPERIMENTS.

		Attributes	
Name	Instances	Categorical	Continuous
WPBC_r	194	0	33
CPU	209	1	8
Yacht	308	0	7
MPG	410	2	5
Housing	452	1	13
Forest Fire	517	2	11
Istanbul	536	0	8
Efficiency	768	0	9
Stock	950	0	10
Concrete	1030	0	9
Flare	1066	10	1
Airfoil	1503	0	6
Red Wine	1599	0	12
Skill Craft	3338	0	20
Elevator	9517	0	7
CCPP	9568	0	5
Bike Share	17379	0	13
Energy Data	19735	0	25
Pm 25	41757	1	12

TABLE III
AVERAGE RRMSE OF THE MODEL PRODUCED BY EACH ALGORITHM OVER FIVE RUNS OF TENFOLD CROSS-VALIDATION.

Dataset	Ant-Miner-Reg$_{\mathrm{MA}}$	Ant-Miner-Reg
WPBC_r	1.0356	**1.0224**
CPU	**0.5038**	0.8233
Yacht	**0.2091**	1.0120
MPG	**0.5374**	0.6419
Housing	**0.5986**	0.9782
Forest Fire	1.5326	**1.0334**
Istanbul	**0.7948**	0.8341
Efficiency	**0.2348**	0.4288
Stock	**0.3258**	0.7434
Concrete	**0.7239**	0.9636
Flare	**0.9956**	0.9987
Airfoil	**0.8165**	0.9715
Red Wine	0.9898	**0.9757**
Skill Craft	**0.8536**	0.8912
Elevator	**0.7585**	0.7882
CCPP	**0.3557**	0.4769
Bike Share	**0.6412**	0.9941
Energy Data	**0.9775**	0.9971
Pm 25	**0.9389**	0.9982

In terms of computational time, Ant-Miner-Reg$_{\mathrm{MA}}$ did not improve the runtime as seen in Table V. This is different than what was observed in classification problems, where the introduction of an archive-based pheromone model did significantly improve the runtime by eliminating the need for a discretisation procedure. Looking at the datasets where Ant-Miner-Reg$_{\mathrm{MA}}$ runtime was significantly higher—Bike Share (17379 instance), Energy Data (19735 instances) and Pm 25 (41757 instances)—we noticed that Ant-Miner-Reg produces very generalised rules with a RRMSE closer to the mean (0.9941, 0.9971, and 0.9982 respectively), while Ant-Miner-Reg$_{\mathrm{MA}}$ produces more specific rules for those dataset with an improved RRMSE (0.6412, 0.9775, and 0.9389 respectively).

We hypothesise that when the dataset is more complex, Ant-

TABLE IV
WILCOXON SIGNED-RANK TEST (AT THE α = 0.05 LEVEL) ON RRMSE. STATISTICALLY SIGNIFICANT DIFFERENCES ARE SHOWN IN BOLD.

	Size	W+	W-	Z	*p*
RRMSE	18	23	167	-2.8974	**0.00374**

TABLE V
AVERAGE RUNTIME IN SECONDS OF THE MODEL PRODUCED BY EACH ALGORITHM OVER FIVE RUNS OF TENFOLD CROSS-VALIDATION.

Dataset	Ant-Miner-Reg$_{\mathrm{MA}}$	Ant-Miner-Reg
WPBC_r	0.51	**0.25**
CPU	**0.23**	0.46
Yacht	**0.20**	0.31
MPG	0.26	**0.24**
Housing	0.43	**0.26**
Forest Fire	0.99	**0.42**
Istanbul	0.39	**0.38**
Efficiency	0.70	**0.39**
Stock	0.85	**0.40**
Concrete	1.29	**0.44**
Flare	0.84	**0.25**
Airfoil	0.92	**0.57**
Red Wine	0.94	**0.61**
Skill Craft	10.12	**0.93**
Elevator	4.65	**2.28**
CCPP	**2.29**	12.22
Bike Share	223.52	**2.13**
Energy Data	582.38	**4.62**
Pm 25	615.24	**6.81**

Miner-Reg struggles to find good split points and produces very simple overgeneralised rules that cover large sections of the search space. This can be seen in the RRMSE of models produced for large datasets, identifying a potential limitation of using M5 dynamic discretisation procedure to create regression rules. The dynamic discretisation procedures in classification and regression Ant-Miner algorithms operate differently. In regression, the dynamic discretisation procedure aims to find the optimal split point for a continuous attribute in the set of uncovered instances, without considering how other attributes will alter the final prediction the rule. This limits the interaction between the creation of condition and the rule's final consequent, which is unknown during rule creation. In classification, the dynamic discretisation procedure aims to find the optimal split points for an attribute in the set of uncovered instances taking into account maximisation of a known target class, which improves the rule prediction. The archive-based approach overcomes this difficulty as the values chosen for continuous attributes are optimised in conjunction with all attributes and not in isolation.

Although Ant-Miner-Reg$_{\mathrm{MA}}$ did not improve the runtime of the Ant-Miner-Reg, the improvement in RRMSE shows great promise for regression problems. This confirms the hypothesis that the archive-based pheromone model improves the values chosen for the continuous attributes conditions in regression problem reaching better rules with overall lower RRMSE—Table IV shows Ant-Miner-Reg$_{\mathrm{MA}}$ achieved a statistically significant improvement with a value of $p = 0.00374$ with respect to Ant-Miner-Reg using Wilcoxon signed-rank test (at the α = 0.05 level) on RRMSE.

V. CONCLUSION

This paper presented an ACO-based regression algorithm, called Ant-Miner-Reg$_{\mathrm{MA}}$, which uses an archive-based pheromone model to handle both categorical and continuous attributes. The proposed algorithm significantly outperforms Ant-Miner-Reg, producing models with better relative root mean square error. The results showed an interesting comparison on how the archive-based pheromone model affected regression problems differently to classification problems. While Ant-Miner$_{\mathrm{MA}}$ improved the runtime of the graph-based algorithm in classification problems, improvements on the quality of the rule lists were not observed [4]. The opposite was observed in Ant-Miner-Reg$_{\mathrm{MA}}$, where the archive-based algorithm improved the relative root mean square error of the model while increasing the runtime. The results confirm that the proposed Ant-Miner-Reg$_{\mathrm{MA}}$ improved the dynamic discretisation for continuous attributes, reinforcing the significant benefits of using an archive-based pheromone model in regression problems.

Future investigation is required to realise the full potential of adding the archive-based pheromone model to rule discovery in regression algorithms. Using an archive-based pheromone improved the quality of the models created in ACO-based regression algorithm. It would be interesting to further investigate the effect of incorporating a graph pheromone model in combination with an archive-based pheromone model, where the graph pheromone model is responsible for selecting attributes and the archive pheromone model for optimising their values. Generating rule lists in each colony iteration, instead of a single rule, to allow the rule interactions to be optimised is also an interesting research directions worth further exploration.

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smith, "From data mining to knowledge discovery: an overview," in *Advances in Knowledge Discovery & Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, Eds. MIT Press, 1996, pp. 1–34.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr 1997.

[3] 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*, vol. 18, no. 4, pp. 503–518, 2014.

[4] A. Helal and F. E. Otero, "A mixed-attribute approach in ant-miner classification rule discovery algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. New York, NY, USA: ACM, 2016, pp. 13–20. [Online]. Available: http://doi.acm.org/10.1145/2908812.2908900

[5] A. Helal and F. E. B. Otero, "Automatic design of ant-miner mixed attributes for classification rule discovery," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 433–440. [Online]. Available: http://doi.acm.org/10.1145/3071178.3071306

[6] J. Brookhouse and F. E. Otero, "Discovering regression rules with ant colony optimization," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1005–1012. [Online]. Available: http://doi.acm.org/10.1145/2739482.2768450

[7] J. Quinlan, "Learning with continuous classes," in *Proceedings 5th Australian Joint Conference on Artificial Intelligence*. World Scientific, 1992, pp. 343–348.

[8] F. Janssen and J. Fürnkranz, "Seperate-and-conquer regression," in *Proceedings of the German Workshop on Lernen*, 2010, pp. 81–89.

[9] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx, *Regression: Models, Methods and Applications*. Springer, 2013.

[10] G. Holmes, M. Hall, and E. Frank, "Generating rule sets from model trees," in *Proceedings 12th Australian Joint Conference on Artificial Intelligence*. Springer, 1999, pp. 1–12.

[11] B. Minnaert and D. Martens, "Towards a particle swarm optimization-based regression rule miner," in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, 2012, pp. 961–963.

[12] R. Parpinelli, H. Lopes, and A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 321–332, Aug 2002.

[13] 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)*, March 2009, pp. 225–231.

[14] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221706006333

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