Using an Ant Colony Optimization Algorithm for Monotonic Regression Rule Discovery

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

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

ABSTRACT

Many data mining algorithms do not make use of existing domain knowledge when constructing their models. This can lead to model rejection as users may not trust models that behave contrary to their expectations. Semantic constraints provide a way to encapsulate this knowledge which can then be used to guide the construction of models. One of the most studied semantic constraints in the literature is monotonicity, however current monotonically-aware algorithms have focused on ordinal classification problems. This paper proposes an extension to an ACO-based regression algorithm in order to extract a list of monotonic regression rules. We compared the proposed algorithm against a greedy regression rule induction algorithm that preserves monotonic constraints and the well-known M5' Rules. Our experiments using eight publicly available data sets show that the proposed algorithm successfully creates monotonic rules while maintaining predictive accuracy.

CCS Concepts

•**Computing methodologies** → **Supervised learning by regression;** *Rule learning; Continuous space search;*

Keywords

ant colony optimization; semantic constraints; monotonic; data mining; regression rules; sequential covering

1. INTRODUCTION

Data mining is a research area focused on automating the search for useful patterns in data [9]. Many algorithms concentrate on producing accurate models, however while accuracy is an important feature other parameters of a model are also important including a model's comprehensibility and its ability to preserve pre-existing domain knowledge. Both of these features can aid the acceptance of a model amongst its users who are normally experts in the domain being investigated.

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908896

This paper aims to investigate adding domain knowledge in the form of semantic constraints to the learning process of an Ant Colony Optimization (ACO)-based regression rule learner to produce accurate models that conserve existing domain knowledge.

ACO-based algorithms are stochastic algorithms that use colonies of artificial ants that mimic the path finding behaviour of real ants to solve hard optimisation problems. The artificial ants traverse a construction graph to build full solutions [6]. In data mining, the ACO-based algorithm Ant-Miner [20] and its derivatives have been successfully used to create lists of classification rules while Ant-Miner-Reg [4] — inspired by Ant-Miner — has tackled the regression task, producing regression rules that outperform a classical greedy search approach.

Semantic constraints allow algorithms to capture and use existing domain knowledge to guide the construction of models. Algorithms that do not pay attention to these constraints may build models that break this relationship. This failure can lead to decreased model acceptance by domain experts due to their counter intuitiveness [12]. Monotonic constraints are one form of semantic constraints, e.g., house prices display monotonic properties as it is expected that as the size of a house increases so will its price.

The rest of the paper is structured as follows. Firstly, we present work from the literature that has been completed in related areas, including regression, ant colony optimization and semantic constraints. Section 3 details the proposed extension to Ant-Miner-Reg. Next, Section 4 presents our results followed up by a discussion, before drawing our conclusions and suggesting possible future work in Section 6.

2. BACKGROUND

There are three areas of related work, regression, ant colony optimization (ACO) and semantic constraints. First we will discuss regression and regression rule models followed by a description of ACO algorithms and Ant-Miner-Reg — the initial application of an ACO-based rule learner for the regression task. Finally, a summary of the literature surrounding semantic constraints and more specifically monotonic constraints.

2.1 Regression

Regression algorithms aim to construct models that will produce real values predictions for each instance they are given. The prediction of a target value is made based on a set of values of the predictor attributes. This is in contrast to classification problems where the aim of a model is to

Algorithm 1: Generic ACO pseudocode

```
1 while Not Terminated do
2 |   ConstructSolutions()
3 |   LocalSearch() // Optional step
4 |   PheromoneUpdate()
5 end
6 return BestSolution
```

classify each instance into a number of predetermined classes [8]. Regression models can take a number of different forms including linear and non-linear algebraic expressions, e.g., GP and SVR [1, 24], regression trees as produced by CART and M5 [24, 23]. In this paper we will focus on the generation of regression rules, following the structure:

$$IF\ att_1 \geq value_1\ AND\ att_2 = value_2\ THEN\ target \quad (1)$$

where att_1 and att_2 are belong to the set of predictor attributes and $target$ is the mean value of instances covered by the rule in the training data. Instances are covered when they satisfy the conditions present in the antecedent – the *IF ...* part of the rule.

Regression rules consist of (attribute, operator, value) tuples which are joined together with logical ANDs. If all terms are satisfied by an instance the value after the $THEN$ (the consequent) is used as a prediction. Regression rules are used by many algorithms, two of which are M5' Rules [11] and SeCoReg [13]. Ant-Miner-Reg [4] is an ACO-based regression algorithm that also constructs lists of regression rules. Ant-Miner-Reg algorithm will be discussed in detail in Section 3.1.

2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a stochastic meta-heuristic that has been used to approximate solutions to many NP-hard combinatorial problems. A classical ACO application is the traveling salesman problem (TSP) [6]. The TSP is a problem where a salesman has a number of cities he visits, which are all interconnected. The salesman wishes to choose the shortest route that allows him to visit all the cities at least once.

The ACO meta-heuristic was first proposed by Marco Dorigo [6] based on the methods used by real ants to search for food sources and then communicate the location of food to the whole colony. Algorithm 1 shows a generic overview of the ACO meta-heuristic.

An ACO algorithm is an iterative process. During the construction phase (line 2 in algorithm 1) each ant in the colony generates a candidate solution by traversing a construction graph, where the vertices represent the components of a solution. At each vertex, a decision is made by an ant to choose the vertex that should be visited next, this choice is probabilistic. The probability of selection is proportional to the current pheromone level on the edge and a problem specific heuristic information. The probability an edge E_{ij} is selected by an ant is given by:

$$P(E_{ij}) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in \text{ allowed j}} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} \quad (2)$$

where τ is the pheromone level of an edge, η is its heuristic value, α and β are constants that alter the importance of each component. This equation was used by Dorigo in the original ant system [6]. Other ACO derivatives may use other methods for calculating probabilities.

Table 1: Simple house rental data set.

Target Attribute	Predictor Attributes	
Rental Value	Floor Area	Location
£300	45	2
£600	80	1
£250	33	3
£400	65	2
£450	70	1
£350	54	2

Once candidate solutions have been generated an optional step can be performed where a local search can be used to optimise the verticies chosen by the ants. In classification this local search takes the form of pruning strategies that follow rules to remove terms that improve the quality of a rule [20].

Finally the quality of the generated solutions is used to update the pheromone values for the construction graph's edges. First edges used in good solutions have their pheromone level increased to strengthen the probability they are selected in the next iteration — the increase in pheromone value is proportional to the solution quality. Edges that remain unused go through a process of evaporation where their pheromone levels are decreased to suppress their selection. Pheromone evaporation allows colonies to forget poor decisions they made in the past and explore new areas of the problem domain. The pheromone gives the ant colony an implicit memory, allowing future ants to improve their solutions based on the successes of those before them [6].

These steps are then repeated until set stopping criteria are reached, typically these are a fixed number of iterations or the ants converge on a solution. At which point the best solution is returned by the algorithm.

Ant-Miner [20] and its extensions have been successfully used to generate classification rules in data mining applications, as the ants traverse the graph they generate a rule by adding the terms that they visit on their path while the pheromone is increased based on the quality of the best rule generated by the colony in each iteration. cAnt-Miner [18] extended Ant-Miner by allowing the use of continuous attributes which are discretised at run time rather than a pre-processing step. Ant-Miner-Reg [4] is inspired by both Ant-Miner and cAnt-Miner however it focuses on the construction of regression rules, like Ant-Miner it constructs rules by traversing a graph adding individual terms and can also handle continuous attributes by using a dynamic discretisation.

2.3 Semantic Constraints

Semantic constraints incorporate existing domain knowledge into the construction of new models. For example, when you consider house rent the price can depend on features such as the location, floor area. Table 1 shows a simple hypothetical dataset. An obvious relationship in this data

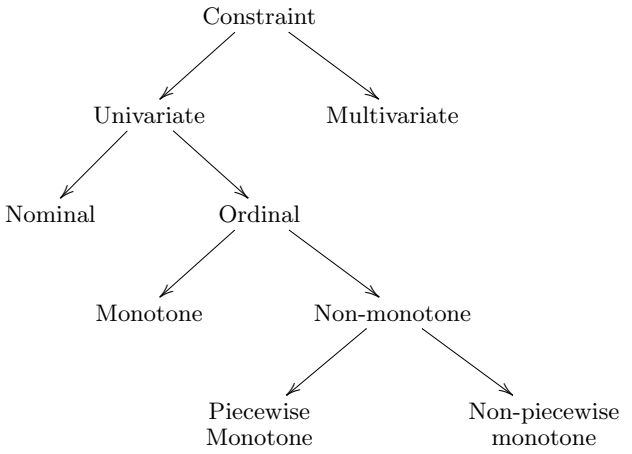

Figure 1: Taxonomy of constraints, all constraints can exist in soft and hard variants. Univariate Ordinal Monotone constraints are the most common found in the literature [17].

set is that as the floor area increases so does the rental price for all possible pairs in the data set.

A model that does not conserve these patterns would seem counter intuitive and may lead to model rejection by domain experts. Hoover and Perez [12] state that the economic field distrusts data mining as a technique to search for models due to the discovery of accidental correlations. They say "*Data mining is considered reprehensible largely because the world is full of accidental correlations, so that what a search turns up is thought to be more a reflection of what we want to find than what is true about the world.*" [12, p. 197]. Semantic constraints provide a method for guiding searches by providing information on real correlations present within the data.

There are many different possible constraints, Martens et al. [17] presents a taxonomy of constraints. This taxonomy can be seen in Figure 1, where the constraints featuring in the taxonomy can be implemented in either *hard* or *soft* variants. Hard constraints are enforced rigidly guaranteeing compliance in the final model, while soft constraints bias a preference towards compliance but will not enforce it if model accuracy would be badly affected. So far the literature has focused on implementing monotonic constraints when tackling the classification task. Many real-world problems contain monotonic properties such as house prices, customer credit ratings [3]. This paper explores the implementation of monotonic constraints to the data mining regression task.

2.4 Monotonicity

Monotonicity is found in many different fields including house prices, medicine, finance and law. Taking the first example of house prices, it is expected that as the total floor area of a property increases the value of the property will also increase, this is illustrated in the example data shown in Table 1 where the rental value is always monotonically increasing with respect to the floor area. From the rental price data set shown in Table 1 we could extract the rules:

```
IF floor area ≤ 65 THEN 325 ELSE
IF floor area ≥ 65 AND location = 1 THEN  525
```

where we can see that the rules have a monotonic relationship between floor area and rental price with respect to each another, as no prediction can be made where the floor area decreases and the price would increase. Incidentally the model is also monotonic w.r.t. location as the second rule does not constrain this attribute so can be ignored.

Many data mining algorithm do not enforce this relationship when constructing models and still produce good models. However, if models violate these constraints they may not be accepted by experts as valid, conforming to monotonicity constraints improves model acceptance [7, 10].

Monotonicity can be defined formally in the following manner. Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_i$ be the instance space of i attributes, $\mathcal{Y}$ be the target space, and function $f : \mathcal{X} \to \mathcal{Y}$, it is also assumed that both the instance space and target space have an ordering. A function can then be considered monotone if:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : \mathbf{x} \leq \mathbf{x}' \implies f(\mathbf{x}) \leq f(\mathbf{x}') \tag{3}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two vectors in instance space, $\mathbf{x} = (x_1, x_2, \cdots, x_p)$ [21]. In other words, $f(\mathbf{x})$ is monotonic if and only if all the pairs of examples $\mathbf{x}$, $\mathbf{x}'$ are monotonic with respect to each other.

Monotonicity constraints can be enforced in a number of different stages in the data mining process. The first is in the pre-processing stage where the training data is manipulated so that is becomes monotonic in nature. This method does not enforce constraints in the model so it will be discussed no further. In the model construction stage models are constructed in a monotonic fashion. Finally constraints could be enforced in a post processing stage which modifies constructed models so that they are monotonic. Table 2 presents previous work identified in the literature review into these categories.

Constraints can be implemented as *hard* or *soft* variants. Hard constraints are enforced rigidly and reject any model or change to a model that would cause a violation to occur. This method can cause the rejection of good models due to small violations in their monotonicity. The second method, soft constraint, is to balance the monotonicity of a model against other model quality measures.

2.4.1 Model Construction

Soft constraints have been implemented in the model construction stage by Ben-David [2]. The approach attempts to assign a non-monotonicity index to each decision produced. The index is the ratio between the number of non-monotonic leaf node pairs and the maximum number of pairs that could have been non-monotonic. First a non-monotonicity matrix m is constructed which has dimensions k (the number of branches in the tree). This matrix is used to find the number of violations in the current tree, given by:

$$\begin{aligned} W &= \sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij} \\ m_{ij} &= \begin{cases} 1 & \text{if } i,j \text{ is non-monotonic} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where i and j denote the current cell being referenced in

Table 2: Monotonicity implementations found in the literature categorised by their implementation stage.

Implementation Stage	Algorithm Names	Reference
Pre Processing	Nearest Neighbour with Monotonicity Constraints	[7]
	Relabelling data to ensure monotonicity	[5]
Model Construction	Decision Trees with Modified Entropy Calculation	[2]
	Ordinal learning model	[3]
	Fused Monotonic Decision Trees	[22]
	AntMiner+ with Constraints	[16]
Post Processing	Decision Trees with Monotonic pruning	[10]

the matrix. W can then be used to find a tree's non-monotonicity index, given by:

$$I_{a_1 \ldots a_v} = \frac{W_{a_1 \ldots a_v}}{k^2_{a_1 \ldots a_v} - k_{a_1 \ldots a_v}} \tag{5}$$

where $a_1 ... a_v$ are the attributes being constrained. The $I_{a_1 \ldots a_v}$ index can be converted to an ambiguity score A and then incorporated with a tree accuracy score, given by:

$$A_{a_1 \ldots a_v} = \begin{cases} 0 & \text{if } I_{a_1 .. a_v} = 0 \\ -(log_2(I_{a_1 \ldots a_v}))^{-1} & \text{otherwise} \end{cases} \tag{6}$$

$$T_{a_1 \ldots a_v} = E_{a_1 \ldots a_v} + RA_{a_1 \ldots a_v} \tag{7}$$

where R is the importance given to the monotonicity of trees produced. It should be noted that an entropy based accuracy method was used in this example, which is a logarithmic function hence the ambiguity value is also made logarithmic. This modification is performed to ensure that both measures used to calculate the total T scale at the same rate without one dominating unduly. If the error measure E is altered then care should be taken to modify A based on the new behaviour to ensure the algorithm remains well behaved. It was found that this method of using a combined metric produced fewer models that breached monotonicity constraints while not significantly degrading the accuracy of the trees generated [2].

Ben-David [3] also investigated the effects of monotonicity constraints on ordinal classifiers, with the conjecture that adding monotonicity constraints to learning algorithms will impair their accuracy against those that do not. Ordinal classifiers are classifiers that are aware that there can be an order to discrete categories, for example credit rating may have the categories poor, acceptable and good that has an obvious order. The results presented contain two unexpected results. The authors found that ordinal classifiers did not significantly improve over non-ordinal classifiers. Secondly, the monotonicity algorithms were not able to significantly outperform a majority-based classifier. It is theorised that these results were due to noisy data sets: the monotonic classifiers enforced hard constraints, in the presence of noisy data a softer approach may lead to better results [3].

Qian et al [22] have explored the possibility of fusing monotonic decision trees to improve the accuracy of the final model. This is achieved by reducing the original data set to create sets that maintain the monotonicity of the original. From these new reduced data sets, monotonic trees can be constructed. Each leaf node of a reduced tree then contain probabilities of the correctness of the prediction based on the reduced training set. When a prediction is required, the probabilities at each leaf nodes is averaged with the highest average being the class predicted by the model.

2.4.2 *Post-Processing*

Feelders [10] has suggested that using non-monotonic criteria in tree construction is not beneficial as splits later in the construction process can transform a tree from a state of non-monotonicity to one that is. Therefore pruning methods have been developed to make the minimal number of changes to make a tree monotonic in a post-processing phase [10].

The first method proposed is the Most Non-monotone Parent (MNP) method, which aims to prune the node that gives the most number of monotone pairs. This method has the disadvantage of possibly creating new non-monotonic pairs. The second method proposed is the best fix method, this prunes the node that gives the biggest reduction in non-monotonicity. The authors have also combined these pruning methods with existing complexity pruning methods and found that the monotonic trees produced no significant difference in performance compared to trees produced without monotonic pruning. However it was observed that the trees produced were smaller, which aids the comprehensibility of the models produced [10].

3. DISCOVERING MONOTONIC REGRESSION RULES

In this section we present the proposed extension to Ant-Miner-Reg. Ant-Miner-Reg is a sequential covering algorithm that uses an ACO procedure to construct regression rules. Sequential covering — also known as *separate-and-conquer* — is commonly used to construct lists of rules. The iterative procedure generates a single rule in each cycle while removing any instances covered by the new rule. The extension proposed incorporates monotonicity measures into pruning process, both during rule construction and post rule construction. This allows soft constraints to be used during rule construction enableing the ACO procedure to fully explore the search space while using hard constraints post-construction to enforce the constraints rigourously.

3.1 Ant-Miner-Reg

Algorithm 2 shows the sequential covering pseudocode. During each iteration of the algorithm the ACO returns a single rule (line 5). The rule is then added to the current list of rules and any instances covered by the new rule are removed (lines 7 and 9). Once the number of uncovered instances drop below a preset threshold the default rule is added to the list and the completed list returned. The de-

fault rule is an empty rule whose consequent is the mean of the uncovered instances.

The rule construction process is based on the *c*Ant-Miner algorithm [18]. The high-level pseudocode is shown in Algorithm 3, where *ant_interations* control the number of iterations performed while searching for a new rule and *colony_size* represents the number of ants that will traverse the construction graph in each iteration. The *CreateRule()* function builds each rule using the pheromone levels to determine the terms to be added to each rule — conventionally the probability of a term being selected also relies on a problem specific heuristic, however Ant-Miner-Reg does not employ heuristic information.

3.2 Rule Pruning

The original Ant-Miner-Reg had a single rule pruning method applied after the rule creation (line 9 of Algorithm 3). This has been replaced by a soft pruning method that is applied after each ant has constructed its rule. The soft pruning method removes a single term from the rule, and if this produces an improvement to its quality the new shorter rule is retained. This is repeated until no improvement is measured or there are no terms left in the rule. The rule quality measure uses a combination of factors including a measure of monotonicity and therefore favours these monotonic rules, however its does not prohibit non-monotonic rules if they are shown to have other good features. Section 3.3 presents a detailed explanation of the quality measure.

A second hard rule pruning is also performed (line 28 of Algorithm 3) in a post-construction pruning phase. The monotonicity of the best rule generated is checked, if it is found to be non-monotonic a single term is removed from the rule and the monotonicity re-evaluated. This is repeated until the rule is monotonic and it is then returned; if no terms remain the default rule is returned and the algorithm will return the rule list that has been constructed so far.

The decision to use two monotonic pruning strategies allows the ACO to fully explore the search space. During rule construction soft constraints allow the search to be guided towards the goal of creating monotonic models. After construction hard pruning ensures that the model returned is strictly monotonic.

3.3 Rule Quality Measure

The original Ant-Miner-Reg used the rule quality measure described by Janssen and Furnkranz [13]. This combines both relative coverage and relative root mean squared error (RRMSE), given by:

$$Q = \alpha \cdot (1 - RRMSE) + (1 - \alpha) \cdot relCov \tag{8}$$

where *relCov* represents the relative coverage of the rule, RRMSE is the relative root mean squared error and α is the weighting applied to each component. *relCov* is given by:

$$relCov = \frac{1}{m} \cdot coverage(Rule) \tag{9}$$

where m is the total number of instances which is used to constraint the coverage between 0 and 1. RRMSE is given by:

$$RRMSE = \frac{RMSE}{\sqrt{\frac{1}{m} L_{default}}} \tag{10}$$

Algorithm 2: SequentialCovering(Instances)

```
Data: Instances
Result: RuleList
1  RuleList ⟵ ∅
2  Rule ⟵ ∅
3  while Instances > Threshold do
4  |  // Creates the next rule
5  |  Rule ⟵ ACOLearnOneRule(Instances)
6  |  // Adds rule to list
7  |  RuleList ⟵ RuleList ∪ Rule
8  |  // Removes covered instances
9  |  Instances ⟵ Instances − Covered(Rule, Instances)
10 end
11 // Adds the default rule
12 RuleList ⟵ RuleList ∪ DefaultRule
13 return RuleList
```

$$RMSE = \sqrt{\frac{1}{m} \cdot \sum_{i=1}^{m} (y_i - \bar{y_i})^2} \tag{11}$$

$$L_{default} = \sum_{i=1}^{m} (y_i - y')^2 \tag{12}$$

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

The semantic constraint extensions to Ant-Miner-Reg requires a modification of the rule quality measure, including the addition of a third factor that measures a model's compliance with the required constraints. To measure the non-monotonicity index of a newly constructed rule, the rule is added to the partial list of rules that was created in previous iterations. The non-monotonicity index of this partial model can then be calculated. The partial model will always have a non-monotonicity index of 0 at the beginning of the ACO procedure – this is enforced by hard monotonic pruning before a rule is committed to the rule list. Therefore, any increase in the non-monotonicity index is attributed to the new rule that has been added. The non-monotonicity index is given by:

$$\begin{aligned} NMI &= \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij}}{k^2 - k} \\ m_{ij} &= \begin{cases} 1 & \text{if } m_{ij} \text{ is a non-monotonic pair} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

where m_{ij} defines if the pair of rules $rule_i$ and $rule_j$ violate the constraint, k is the number of rules in the model. The NMI of a model is constrained between 0 and 1. The non-monotonicity index calculates the ratio of monotonic violating pairs over the total possible number of prediction pairs present in the model being tested, the lower a non-monotonicity index is the better a model is considered. If this is the first rule in the partial model it will be automatically designated monotonic and be assigned a non-monotonicity index of 0. A pair is found to be monotonic if the constrained attributes and prediction satisfy the criteria, this differs from the literature where the monotonic test is

Algorithm 3: ACOLearnOneRule(Instances)

```
Data: Instances
Result: BestRule
BestEval ⟵ ∞, BestRule ⟵ null
PheromoneInitialization()
for i = 0 to ant_iterations do
    // Each ant creates a rule, remembering
    // the best for later
    MaxEval ⟵ −∞, MaxRule ⟵ null
    for j = 0 to colony_size do
        Rule ⟵ CreateRule()
        PruneRuleSoft(MaxRule)
        Eval ⟵ EvaluateRule(Rule, Instances)
        if Eval > MaxEval then
            MaxEval ⟵ Eval
            MaxRule ⟵ Rule
        end
    end
    // Adds the best rule and update the
    // pheromone levels with the best rule
    // produced by the colony
    UpdatePheromone(MaxRule)
    // If the max rule is better than the best
    // rule found from all iterations update
    // the best rule
    if MaxEval > BestEval then
        BestEval ⟵ MaxEval
        BestRule ⟵ MaxRule
    end
end
PruneRuleHard(BestRule)
return BestRule
```

only required if all others attributes are equal. This conservative approach to monotonicity is required as continuous attributes are rarely equal in real data sets.

The three separate measures are incorporated into a single equation, given by:

$$Q_{sem} = \alpha \cdot (1 - L_{RRMSE}) + \beta \cdot relCov + \gamma \cdot (1 - NMI) \quad (14)$$

where α, β and γ are used to weight the importance of the three individual measures and can be optimised for the best performance and create a single quality measure that is constrained between 0 and 1.

4. RESULTS

Ant-Miner-Reg has been previously compared to SeCoReg [14], a greedy sequential covering algorithm. In this paper we have compared the proposed Ant-Miner-Reg+MC to SeCoReg+MC (SeCoReg modified to prune rules to enforce monotonic constraints) and the non-constraint aware M5' Rules [11]. Table 3 contains the configuration of Ant-Miner-Reg+MC and SeCoReg+MC, the values chosen are based on those used by the original SeCoReg [13] and have not been optimised for either Ant-Miner-Reg or the addition of monotonic constraints.

Eight publicly available data sets from the UCI Machine Learning Repository [15] were chosen as they contain attributes that have a low non-monotonicity index. These

Table 3: Parameters used for SeCoReg+MC and Ant-Miner-Reg+MC algorithms.

General Parameter	Value
Minimum Covered Rule	10
Minimum Uncovered Theory	0.1
Split Points	3
Error Weighting (α)	0.4
Coverage Weighting (β)	0.3
Constraint Weighting (γ)	0.3
Cross Validation Folds	10
ACO Parameter	**Value**
Iterations	500
Colony Size	10

data sets are shown in Table 4, which summarises the number of instances in each data set, the make up of the attributes, and constraint information. Experiments were run using tenfold cross-validation, both SeCoReg+MC and M5' Rules were ran once however Ant-Miner-Reg+MC was ran 5 times across the same folds and an average taken to help avoid spurious results due to the stochastic nature of ACO.

The average RRMSE of each model produced by the 3 algorithms are shown in Table 5 with the smallest error for each data set shown in bold. Ant-Miner-Reg+MC managed to achieve lower RRMSE values than M5' Rules 2 out of 8 times and SeCoReg+MC in seven of the data sets. M5' Rules achieved lower RRMSE value than SeCoReg+MC in all 8 data sets and was the algorithm with the lowest RRMSE values overall.

Further statistical analysis was performed on the experiment results where the Friedman test was applied with a Holm post-hoc test. This showed that M5' Rules did not statistically outperform Ant-Miner-Reg+MC at the 5% confidence level as shown in Table 6.

Finally, the monotonicity of the models was checked, both Ant-Miner-Reg+MC and SeCoReg+MC produced monotonic models — this is guaranteed by the hard monotonic pruner that is run before rules are added to the list. Table 7 shows the non-monotonicity index for the models produced by M5' Rules. As M5' Rules contain linear models it is possible for a rule to internally violate monotonicity if the coefficient for a constrained attribute has the incorrect sign. This means that another method is required to estimate its non-monotonicity. The models produced on each cross-validation fold were used to re-label the testing data to create a new data set, from this each pair in the data can be compared. This allows non-monotonicity index for each of the relabelled data sets to be measured.

5. DISCUSION

Ant-Miner-Reg+MC has been shown to produce models that preserve the monotonic constraints imposed upon it while not suffering a statistically significant decrease to its performance when compared to M5' Rules. It manages to outperform M5' Rules in two of the eight data sets. However M5' Rules has been shown to be significantly better than the greedy search algorithm SeCoReg+MC which has been

Table 4: Attribute makeup and constraint information of the eight UCI data sets used in the experiments [15]. In each data set a single attribute was constrained, the attribute name, whether it is monotonically increasing or decreasing and the non-monotonicity index of the attribute is given.

		Attributes		Constraint		
Name	Instances	Nominal	Continuous	Constrained Attribute	Direction	NMI
CCPP	9568	0	5	V	Decreasing	0.080
CPU	209	1	8	MMax	Increasing	0.074
Elevators	9516	0	6	ClimbRate	Increasing	0.080
Flare	1065	10	1	LargestSpot	Increasing	0.065
Housing	452	1	13	LSTAT	Decreasing	0.087
MPG	392	3	5	Horsepower	Decreasing	0.084
Red Wine	1599	0	12	Alcohol	Increasing	0.090
Yacht	308	0	7	Froude	Increasing	0.035

Table 5: RRMSE of the model produced by each algorithms in each of the eight data sets. The bold value indicates the smallest error of the three algorithms; the standard deviation is shown in square brackets.

Data Set	Ant-Miner-Reg+MC	SecoReg+MC	M5' Rules
CCPP	**0.1715 [0.0126]**	0.2853 [0.0150]	0.2375 [0.0136]
CPU	0.3564 [0.2269]	0.4194 [0.2074]	**0.1707 [0.1438]**
Elevators	0.8574 [0.1421]	1.0017 [0.0020]	**0.6014 [0.0134]**
Flare	**0.9480 [0.0283]**	1.0099 [0.0144]	1.0086 [0.0257]
Housing	0.5088 [0.2732]	0.8936 [0.2282]	**0.4396 [0.1154]**
MPG	0.4639 [0.1724]	0.6203 [0.0588]	**0.3723 [0.0455]**
Red Wine	1.0745 [0.5425]	1.0099 [0.0160]	**0.8068 [0.0354]**
Yacht	0.1811 [0.0385]	0.4597 [0.1168]	**0.0833 [0.0264]**

Table 6: Non-parametric Friedman test with Holm's post-hoc test results based on the average RRMSE of the three algorithms used in the experiments. Statistically significant results at the $\alpha = 0.05$ level are shown in bold.

Algorithm	Avg. Rank	p-value	Holm
M5' Rules	1.25	–	–
Ant-Miner-Reg+MC	2.0	0.1336	0.05
SeCoReg+MC	**2.75**	**0.0027**	**0.025**

Table 7: Estimation of the non-monotonicity index of the models produced by M5' Rules by relabelling the data sets and calculating their NMI.

Data Set	Model NMI	Data Set	Model NMI
CCPP	0.0707	Housing	0.0735
CPU	0.0994	MPG	0.0804
Elevators	0.1010	Red Wine	0.1221
Flare	0.2197	Yacht	0.0445

modified to preserve the same constraints as Ant-Miner-Reg+MC. While M5' Rules on average does produce better RRMSE values, all of the models produced by the algorithm are non-monotonic. Five of the eight models produced showed an increase in the non-monotonicity over the original data sets.

In this initial implementation the ACO-based Ant-Miner-Reg algorithm with monotonic constraints shows that future investigation is warranted to improve the performance against the current state-of-the-art data mining algorithms. The use of regression rules as a comprehensible model coupled with the conservation of monotonic constraints should provide an increase in model acceptance when compared with other black box models which may or may not conserve the monotonic properties of a data set.

Finally the non-monotonicity of all models generated was checked and it was found that as designed both AntMiner-Reg+MC and SeCoReg+MC produced monotonic models. M5' Rules was shown to produce non-monotonic models in all data sets and in cases the non-monotonicity of the relabeled data set was greater than that of the original. This shows that existing algorithms are not capable of extracting and conserving these constraints on their own and require additional information on them. To the best of our knowledge there are no existing regression rule monotonic algorithms in the literature.

6. CONCLUSION

This paper presented an extension to the existing ACO-based algorithm Ant-Miner-Reg that preserved the monotonic constraints imposed upon it while constructing models, called Ant-Miner-Reg+MC. The algorithm outperformed a greedy search strategy that also conserved monotonic constraints. Ant-Miner-Reg+MC was shown to not be significantly worse than M5' Rules at the 5% significance level in terms of RRMSE, while successfully creating monotonic rules.

It was also shown that as expected M5' Rules fails to preserve the monotonic constraints. The incorporation of existing domain knowledge and therefore the preservation of monotonic constraints may aid model acceptance among users so it a worth while property to optimise when constructing models. The parameters used by Ant-Miner-Reg+MC have not been optimised for the specific algorithm which may lead to performance improvements in the future.

Monotonicity is a global property of a model, currently Ant-Miner-Reg+MC builds a model in a *one-rule-at-a-time* fashion and does not modify previous rules. Further work is required to allow full rule lists to be generated in a single iteration (as proposed in [19]) which would allow more complex pruning strategies optimising monotonicity in a global manner. Ant-Miner-Reg has a number of possible future enhancements including the use of heuristic information during rule construction. Additionally the adoption of linear models as rule consequents in a similar fashion to M5' Rules may reduce the RRMSE of Ant-Miner-Reg although with a loss in comprehensibility when compared to simple single valued rule consequents.

7. REFERENCES

[1] D. Augusto and H. Barbosa. Symbolic regression via genetic programming. In *Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on*, pages 173–178. IEEE, 2000.

[2] A. Ben-David. Monotonicity maintenancs in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.

[3] A. Ben-David, L. Sterling, and T. Tran. Adding monoticity to learning algorithms may impair their accuracy. *Expert Systems with Applications*, 36:6627–6634, 2009.

[4] J. Brookhouse and F. Otero. Discovering regression rules with ant colony optimization. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference (GECCO 2015)*, pages 1005–1012, 2015.

[5] H. Daniels and M. Velikova. Derivation of montone decision models from noisy data. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(5):705–710, 2006.

[6] M. Dorigo and T. Stutzle. *Ant Colony Optimization*. A Bradford Book, 2004.

[7] W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *Machine Learning and Knowledge Discovery in Databases*, volume 5211 5211, pages 301–316. Springer Berlin Heidelberg, 2008.

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

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

[10] A. Feelders and M. Pardoel. Pruning for monotone classification trees. In *Advances in intelligent data analysis V*, pages 1–12. Springer, 2003.

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

[12] K. Hoover and S. Perez. Three attitudes towards data mining. *Journal of Economic Methodology*, 7(2):195–210, 2000.

[13] F. Janssen and J. Furnkranz. Seperate-and-conquer regression. In *Proceedings of the German Workshop on Lernen*, pages 81–89, 2010.

[14] F. Janssen and J. Furnkranz. Seperate-and-conquer regression. Technical Report TUD-KE-2010-01, Knowledge Engineering Group, Technische Universitat Darmstadt, 2010.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] D. Martens, M. D. Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen. Ant-based approach to the knowledge fusion problem. In *Ant Colony Optimization and Swarm Intelligence*, pages 84–95. Springer, 2006.

[17] D. Martens and B. Baesens. Building acceptable classification models. In *Data Mining*, pages 53–74. Springer, 2010.

[18] F. Otero, A. Freitas, and C. Johnson. *c*Ant-Miner: An ant colony classification algorithm to cope with continuous attributes. In *Ant Colony Optimization and Swarm Intelligence (Proc. ANTS 2008)*, 2008.

[19] 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, February 2013.

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

[21] R. Potharst, A. Ben-David, and M. van Wezel. Two algorithms for generating structured and unstructured monotone ordinal data sets. *Engineering Applications of Artificial Intelligence*, 22(4):491–496, 2009.

[22] Y. Qian, H. Xu, J. Liang, B. Liu, and J. Wang. Fusing monotonic decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 2015.

[23] J. Quinlan. Learning with continuous classes. In *Proceedings 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348. World Scientific, 1992.

[24] X. Wu, V. Kumar, J. Quanlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge Information Systems*, 14:1–37, 2008.