FRANTIC - A system for inducing accurate and comprehensible fuzzy rules

Galea, Michelle; Shen, Qiang

Publication date:
2004

Citation for published version (APA):
Galea, M., & Shen, Q. (2004). *FRANTIC - A system for inducing accurate and comprehensible fuzzy rules*. 136-143. http://hdl.handle.net/2160/430

FRANTIC - A System for Inducing Accurate and Comprehensible Fuzzy Rules

Michelle Galea
School of Informatics
University of Edinburgh
m.galea@sms.ed.ac.uk

Qiang Shen
Department of Computer Science
University of Wales, Aberystwyth
qqs@aber.ac.uk

Abstract

An approach to fuzzy rule induction inspired by the foraging behaviour of ants is presented. The implemented system - *FRANTIC* - is tested on a real classification problem against two other fuzzy rule induction algorithms, one with an emphasis on rule comprehensibility, and the other on rule accuracy. The results obtained highlight *FRANTIC*'s ability to balance the tradeoff often encountered between predictive accuracy on the one hand, and ruleset comprehensibility on the other. *FRANTIC*'s actual and potential strength when applied to real-world large datasets is highlighted, while its limitations and the possible ways of overcoming them are also discussed.

1 Introduction

The activities of social insects have inspired successful applications in many areas. For instance, the cemetery organisation and brood sorting activity of ants has led to new clustering algorithms (e.g. graph colouring and partitioning [9]), while their foraging behaviour has resulted in a suite of optimisation algorithms - called ant algorithms - for solving problems such as bin packing and the travelling salesman problem [5]. It is these optimisation algorithms that have been adapted for rule induction.

The appeal of ant algorithms lies in several factors: they provide a simple effective mechanism for conducting global search by simultaneously constructing multiple solutions that investigate diverse areas of the solution space; a simplicity of implementation that requires minimum understanding of the problem domain; the problem-specific elements - such as the fitness function and heuristic - which may be readily borrowed from existing literature on rule induction; and, an explicit heuristic embedded in the solution construction mechanism that makes for easy insertion of domain knowledge.

The application of ant algorithms to rule induction is an under explored research area. This paper further develops work done in [6] for inducing linguistic fuzzy IF-THEN rules, making a simple modification to the original version of *FRANTIC* that allows it to better deal with real-world datasets that contain an imbalanced number of instances representing different classes. The implementation is tested on a classification problem, but this approach is equally applicable to fuzzy modelling for other tasks, e.g. prediction.

In the next section the motivation and details behind the Ant Colony Optimisation (ACO) metaheuristic, instantiations of which result in different ant algorithms, are introduced. The existing literature on the application of ACO-based algorithms to rule induction is also briefly reviewed. In Section 3 the *FRANTIC* system is described, while Section 4 presents the initial findings of the system, comparing it with the results obtained from teo other fuzzy rule induction algorithms. Section 5 highlights the advantages and limitations of the current research, which in turn suggest avenues for future work.

2 Ant Algorithms

Ant algorithms is a collective term for algorithms motivated by the way ants forage for food. Experiments with real ant colonies have been conducted to determine how ants are able to find the shortest path between their nest and a food source. In making decisions about which path to take ants are guided by the amount of pheromone (a chemical substance laid by ants) on a path - the greater the amount the higher the probability an ant will follow that path.

When a new food source is first located there is no pheromone to guide ants and so each will have made a random decision when presented with different paths. Several paths by different ants may therefore have been taken to reach the same food source. On their return trip to the nest ants will lay more pheromone and those that have found the shortest path will get back to the nest more quickly. Pheromone, however, evaporates so unless the amount on a path is maintained by ants that continue to use it, the path will be chosen less frequently. The shortest path will however continue to

accrue more pheromone as ants are able to travel faster along this path and can lay pheromone more quickly to replace that which is evaporating. It has been observed that foraging ants usually converge on the shortest path to a food source.

2.1 Ant Colony Optimisation

ACO is an agent-based meta-heuristic motivated by these foraging strategies of real ants. In ACO, each artificial ant is considered a simple agent, communicating with other ants indirectly by effecting changes to a common environment. A high-level description of an ACO-based algorithm is:

```
(1)    while termination condition false
(2)        each ant constructs a new solution
(3)        evaluate new solutions
(4)        update pheromone levels
(5)    output best solution
```

The pheromone levels and a problem-specific heuristic are what guide artificial ants in their construction of a solution. Any combinatorial problem for which the following listed elements can be defined may be solved by an ACO algorithm [2]. These elements are introduced briefly here in the context of rule induction, with more detail provided in Section III. The first four elements relate to line (2) of the high-level ACO description above, the fifth relates to line (3), and the sixth to line (4):

1. An appropriate *problem representation* is required that allows an artificial ant to incrementally build a solution using a *probabilistic transition rule*. The main idea is to model the problem as the search for a best path through a graph. In the context of rule induction a solution is a rule antecedent and each node of the graph represents a condition that may form part of it, such as OUTLOOK=Sunny, or OUTLOOK=Cloudy.

2. A local *heuristic* provides guidance to an ant in choosing the next node for the path (solution) it is building. Possible examples may be based on fuzzy subsethood values, or a measure of the vagueness in a fuzzy set.

3. The *probabilistic transition rule* determines which node an ant should visit next. The transition rule is dependent on the *heuristic* value and the *pheromone* level associated with a node.

4. A *constraint satisfaction method* forces the construction of feasible rules. For instance, if simple propositional IF-THEN rule antecedents are being constructed, then only one fuzzy linguistic term from each fuzzy variable may be selected.

5. A *fitness function* determines the quality of the solution built by an ant.

6. The *pheromone update rule* specifies how to modify the pheromone levels of each node in the graph. For instance, between iterations of an ACO algorithm, the nodes (conditions) contained in the best rule antecedent created get their pheromone levels increased.

2.2 Rule Induction via Ant Colony Optimisation

The application of ant-inspired algorithms to rule induction is an unexplored research area.

A first attempt was made by Casillas *et al* in [3]. However, the ACO algorithm is not used for generating fuzzy rules, but for assigning rule conclusions. In their problem graph the fixed number of nodes are fuzzy rule antecedents found by a deterministic method from the training set. An ant goes round the problem graph, visiting each and every node in turn and probabilistically assigns a rule conclusion to each.

In [11] an ACO algorithm is used for generating crisp IF-THEN rule antecedents. In the problem graph each node represents a condition that may be selected as part of the crisp rule antecedent being built by an ant. An ant goes round the graph selecting nodes according to a constraint satisfaction method, building its rule antecedent. The rule conclusion is assigned afterwards by a deterministic method. The overall strategy used is one of iterative rule learning - starting with a full training set an ACO algorithm is run and the best rule created by an ant is added to a final rule set. Instances in the training set that are covered by this best rule are removed before a new ACO algorithm is run. This process is re-iterated until only a few (as pre-determined by the user) instances remain in the training set, when a default rule is created to cover them. The final result is an ordered rule list with the rules being applied in the order in which they were created, when classifying a new instance.

FRANTIC originated by developing this work on inducing crisp rules [**?**], modifying the overall strategy used and extending it for the induction of fuzzy rules.

3 The *FRANTIC* System

FRANTIC (Fuzzy Rules from ANT-Inspired Computation) implements a class-dependent iterative rule learning stategy whereby for each class that requires descriptive rules to be learnt, a number of ant algorithms are run with each one outputting one such rule. Note that lines (4)-(8) are equivalent to the ACO-based algorithm previously introduced:

```
(1)    for each class
(2)        reinstate full training set
(3)        while classInstRem>classInstUncovered
(4)            for noIterations
(5)                each ant constructs rule
(6)                evaluate all rules
(7)                update pheromone levels
```

```
(8)          add best rule to finalRuleSet
(9)          remove covered class instances
(10)   output finalRuleSet
```

A simplified version of this strategy is to run just one ACO algorithm for each class, with the assumption being that one rule is sufficient to describe a class. *FRANTIC* may be run in this simplified mode, or carry out a full class-dependent iterative strategy.

If a full version is run, then more than one ACO algorithm may be run per class. From each the best rule constructed is determined and added to the final rule set. However, before the next ACO is run to find another rule describing the same class, the instances belonging to that class that are covered by the previous best rule are removed from the training set. This process goes on until there are fewer class instances remaining in the training set than a value pre-defined by the user, line (3). This parameter provides a simple and effective mechanism for controlling over-fitting to the training data, since continuing to run ACO algorithms to find rules describing the last few class instances may not necessarily produce rule sets with better classification accuracy. Instances belonging to classes other than the one currently being described are left in the training set, as this helps in the evaluation of the rules constructed.

The following subsections explain the ACO-specific elements in more detail.

3.1 Rule Construction

FRANTIC has been designed with the flexibility to create simple propositional rules, propositional rules with internal disjunction (e.g. OUTLOOK=Cloudy OR Sunny), or propositional rules that include negated terms (e.g. OUTLOOK=NOT_Sunny). The problem graph for each is similar and differences in the rule construction mechanism from one knowledge representation to the other are highlighted where appropriate.

When creating a rule antecedent an ant traverses a problem graph where each node represents a term that may be added e.g OUTLOOK=Sunny. In the case of constructing rules with negated terms, the graph has double the number of nodes - one extra for each original linguistic term, e.g. OUTLOOK=NOT_Sunny. The choice of the next node to visit (next term to be added to the current partial rule antecedent) depends on both a heuristic value and the pheromone level associated with the node. The choice is made probabilistically but is biased towards terms that have relatively higher heuristic and pheromone values.

However, after selection and before a term is added to a rule antecedent, a check is made - this ensures that the resultant rule antecedent covers a minimum number of instances from the training set (set by a parameter called `minInstPerRule`), and is another simple way of avoiding over-fitting to the training. With fuzzy sets all fuzzy rules cover all training instances, but to varying degrees, and so what constitutes coverage of an instance by a fuzzy rule needs clarifying. This is defined in subsection *B*.

For simple propositional rules, or rules with negated terms, if an ant does add a term to its rule antecedent then it will not consider other linguistic terms belonging to the same linguistic variable. For example, if the linguistic variable OUTLOOK has terms Sunny, Cloudy, Rain, and the term OUTLOOK=Sunny has just been added to the rule antecedent, then the remaining terms are not considered further. If this restriction is removed, then it is possible for ants to add more than one linguistic term from each variable, with the interpretation being of a disjunctive operator between the terms added.

3.1.1 Heuristic

The heuristic used to guide ants when selecting terms is based on fuzzy subsethood values [8], giving a degree to which one fuzzy set *A* is a subset of another fuzzy set *B*:

$$S(A,B) = \frac{M(A \bigcap B)}{M(A)} = \frac{\sum_{u \in U} Min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)}$$

where in this case *u* is an instance from the training set *U*, *A* represents a class label and *B* a term that may be added to a rule antecedent. The heuristic value of a term *j* (η_j) therefore gives a measurement of how important that term is in describing a specific class. Since *FRANTIC* finds descriptions for each class in turn, heuristic values for terms are calculated at the start of each major iteration, line (1), using only those instances in the training set which belong to the class under consideration.

If fuzzy rules with negated terms are being constructed, the heuristic value for the negated term is the complement of the heuristic value for the non-negated term, i.e. $\eta_{NOT\_j} = 1 - \eta_j$

3.1.2 Pheromone Updating

At the start of an ACO run, line (4), all nodes in the graph have an equal amount of pheromone which is set to the inverse of the number of nodes. The pheromone level of individual nodes, however, changes between iterations, line (7). Towards the end of each iteration, line (6), rules created by all ants are evaluated. The terms in the best rule, say *R*, then get their pheromone levels increased:

$$\tau_j(t+1) = \tau_j(t) + \tau_j(t) \cdot Q, \forall j \in R$$

i.e. at time *t+1* each term *j* in rule *R* gets its pheromone level increased in proportion to the the quality of the

rule *Q*. A normalisation of pheromone levels of *all* terms further results in a decrease of the pheromone levels of terms *not* in *R*.

The pheromone updating process is a reinforcement mechanism - both positive and negative - for ants constructing new rules in successive iterations: terms that have had their pheromone levels increased have a higher chance of being selected, while those that have had their levels decreased have a lower chance.

3.1.3 Transition Rule

Ants select terms while constructing a rule antecedent according to a transition rule that is probabilistic but biased towards terms that have higher heuristic and pheromone levels. The relative importance of the heuristic and pheromone may be controlled by adjusting the values of the parameters α and β, though they have been kept constant and equal in these experiments. The probability that a term *j* is selected by an ant is given by:

$$P_j = \frac{[\eta_j]^\alpha \cdot [\tau_j]^\beta}{\sum_{i=1}^{n}(\eta_i \cdot \tau_i), \forall i \in I}$$

where *i* is any term within the set *I* of all terms, and *n* is the number of terms in the graph.

The probabilistic nature of the rule is a way of introducing exploration into the search for a solution, in the expectation that a more optimal solution may well be found rather than by adhering strictly to terms with the highest values.

3.2 Rule Evaluation

After a rule has been constructed, it needs to be evaluated. This is done by assessing how accurate the rule is in classifying the training instances. Before the fitness function is discussed, what constitutes coverage (or matching) of a fuzzy instance by a fuzzy rule needs to be defined.

3.2.1 Fuzzy Rule Matching

A fuzzy rule is said to cover or match a fuzzy instance if their degree of match is equal to or greater than a pre-defined value, here called a threshold value.

When rule *R* is applied to instance *u* it is necessary to determine how well the attributes of *u* match the condition part of *R*, and how the class of *u* matches the conclusion of *R*. An example follows.

Consider a rule *R*=(1,1,0; 0,0,1; 1,1; 0,1) that represents a rule with four attributes, the last being the class attribute with two possible values. Terms that are present in the rule are denoted by 1, others by 0. These rules may only classify instances into one class. However, the condition attributes may take more than one value (i.e. propositional rules with internal disjunction).

Consider a fuzzy instance *u*=(0.1,0.6,0; 0.1,0,0.8; 0.3,0.4; 0.2,0.7) where the representation is the same as for rule *R*, though the conclusion attribute values may be greater than 0 for more than one class.

The degree of match between *R* and *u* is given by

$$mRule(R,u) = Min(mCond(R,u), mConc(R,u))$$

where the degree of condition match between *R* and *u* is

$$mCond(R,u) = Min_k(mAtt_k(R,u))$$

and the degree of conclusion match is

$$mConc(R,u) = Max_{1 \leq L}(Min(\mu_{Class_j}(R), \mu_{Class_j}(u)))$$

with *L* being the number of class labels, i.e. the number of terms for the class attribute. In the above definitions, $mAtt_k$ measures the degree of match between an attribute *k* in *R* and the corresponding attribute in *u*:

$$mAtt_k = Max_j(Min(\mu_{k_j}(R), \mu_{k_j}(u)))$$

where *j* is a term within the domain of attribute *k*. However, if the terms for an attribute are all present in a rule, then the corresponding attribute match is equal to 1, the interpretation being that the attribute is irrelevant. From the rule and instance examples above the attribute matches are: $mAtt_1 = 0.6$, $mAtt_2 = 0.8$, $mAtt_3 = 1$, with a condition match $mCond(R,u) = 0.6$. The conclusion match is $mConc(R,u) = 0.7$ and the resulting rule match is $mRule(R,u) = 0.6$. If the threshold value is set at 0.6 or below, then this rule is considered to cover the instance. If the threshold value is set above 0.6, then this rule is considered to not sufficiently match the instance.

3.2.2 Fitness Function

The fitness function evaluates an individual rule on the basis of how accurately it classifies all instances in the training set. It combines a measure of the sensitivity of a rule (its accuracy among instances of the same class as the rule) with a measure of the specificity of the rule (its accuracy among instances of different classes):

$$Q = \frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}$$

where

- TP (True Positives) is the number of instances covered by the rule that have the same class label as the rule
- FP (False Positives) is the number of instances covered by the rule that have a different class label from the rule

- FN (False Negatives) is the number of instances that are not covered by the rule but have the same class label as the rule, and
- TN (True Negatives) is the number of instances that are not covered by the rule and do not have the same class label as the rule.

Whether a rule covers an instance or not is determined by a threshold value, pre-defined by the user, for the degree of match between the rule and an instance.

3.3 Classification by Fuzzy Rules

Once a complete fuzzy rule set has been generated, it needs to be tested for classification accuracy. The method adopted to test the rules produced by *FRANTIC*, and those produced by the algorithms against which it is compared ([4, 12]), follows:

1. For each rule, calculate the condition match for instance u, and set the conclusion match equal to the condition match;
2. If two or more rules classify instance u into the same class, choose the highest conclusion match as the degree for that class;
3. Finally, select the class with the highest membership degree as the class for instance u.

4 Results and Analyses

4.1 The Dataset and Other Algorithms

The problem on which *FRANTIC* is tested is the Water Treatment Plant Database [1]. The database contains the daily observations of 38 sensors monitoring the operation of an urban waste water treatment plant at various stages throughout the process, with the objective being to predict faults in the process. Observations were taken over 527 days and are real-valued. The database is ill-conditioned, having 13 possible classifications for each daily set of observations, but with most classifications being assigned to only a few records in the database. Furthermore, when faults are reported these are generally fixed very quickly so that the database is heavily biased by containing a disproportionate number of records indicating correct operation of the plant, versus faulty operation.

The 13 classifications have therefore been collapsed to two: *OK* and *Faulty*, as in [13]. Records that have no assigned classification, and others with missing values have been removed, leaving 377 records for training and testing the rule induction algorithms (289 *OK*, 88 *Faulty*). Other pre-processing steps included fuzzification of the features using trapezoidal functions, and a feature subset selection process [7] to reduce the number of features ([13, 14] indicated better accuracy results in their work when a reduced water treatment dataset was used). A description of the retained features is shown in Table 1. Each feature is described by three linguistic terms (*low, high, normal*), except for the last which uses only two (*low, high*).

Table 1: Water Treatment Plant Features

Name	Sensor Description
Q-E	Input to plant - flow
PH-E	Input to plant - pH
DBO-E	Input to plant - biological demand of oxygen
DBO-P	Input to primary settler - biological demand of oxygen
SSV-P	Input to primary settler - volatile suspended solids
PH-D	Input to secondary settler - pH
DQO-D	Input to secondary settler - chemical demand of oxygen
SSV-D	Input to secondary settler - volatile suspended solids
PH-S	Output - pH
SSV-S	Output - volatile suspended solids
RD-SED-G	Global performance, input - sediments

The fuzzy rule sets generated by *FRANTIC* are compared against those produced by two different methods that are however both based on subsethood values [4, 12]. The first uses subsethood values to determine a fuzzy quantifier for each possible condition in the rule, whilst the second uses subsethood values to select a small number of conditions to formulate a rule. Both algorithms are deterministic, however, [4] requires the setting of two parameters and so may produce different rulesets depending on these parameters. Each algorithm produces only one rule to describe each class, and because of this, the focus here will be on *FRANTIC* rulesets generated using the simplified iterative rule learning approach, i.e. only one rule per class is created.

4.2 *FRANTIC* Parameters

FRANTIC parameters that require setting are listed in Table 3, together with a description of these parameters and the values given in order to obtain the results reported here. Very little parameter tuning has been done and these values are based on a few exploratory runs of the system that indicated reasonable results would be obtained. The parameter values given as 'various' are detailed where appropriate.

It is worth noting that the concept of matching between a rule and an instance is used several times by *FRANTIC* (rows 6-8). These thresholds have been implemented separately for maximum flexibility. Initial findings suggest rule sets with greater classification accuracy are found if the threshold during rule construction is greater than the threshold during fitness evaluation and for removal of class instances between ACO runs (the latter threshold only relevant if full iterative rule learning is used). Further investigation is required to understand the dynamics between these three param-

Table 2: *FRANTIC* I vs *FRANTIC* II Predictive Accuracy

Construction Threshold	Same number of instances			Same proportion of instances		
	50 % (+/-)	60 % (+/-)	70 % (+/-)	60% % (+/-)	70% % (+/-)	80% % (+/-)
0.50	44.79 (10.9)	50.88 (8.00)	55.66 (8.30)	67.60 (9.35)	70.31 (8.06)	76.12 (7.05)
0.55	53.56 (8.26)	51.96 (9.42)	60.46 (10.8)	72.44 (8.97)	73.72 (8.59)	76.12 (6.72)
0.60	55.90 (9.95)	60.43 (14.6)	60.40 (9.53)	73.22 (10.1)	71.58 (11.0)	72.96 (5.57)
0.65	65.81 (9.82)	63.65 (11.1)	63.62 (10.6)	74.30 (10.2)	*76.93* (7.62)	66.84 (8.54)
0.70	59.40 (8.60)	64.44 (8.84)	67.12 (8.63)	75.38 (9.26)	76.37 (7.57)	59.17 (9.81)
0.75	63.61 (10.1)	64.70 (6.99)	67.39 (8.54)	75.35 (8.72)	75.58 (7.98)	67.67 (7.28)
0.80	62.39 (8.71)	67.93 (9.15)	63.22 (7.88)	75.64 (10.1)	75.58 (7.98)	64.99 (6.53)
0.85	64.45 (10.0)	*69.46* (6.57)	62.39 (6.48)	75.62 (7.35)	76.37 (7.57)	64.99 (6.53)

eters, and it may well be possible to merge two or three of them.

4.3 Classification Accuracy

The middle column of Table 4 indicates the average accuracy obtained by the algorithms after performing stratified 10-fold cross-validation (the same folds of the dataset were used for each algorithm). The figure in brackets is the standard deviation of the accuracies of the 10 rulesets produced. The right column gives the average number of terms per rule, with standard deviation in brackets.

Note, that since *FRANTIC* is a stochastic algorithm, the *FRANTIC* results presented in Table 4 are averages of *ten* 10-fold cross-validations.

Table 4: Comparison of Algorithms

	%Accuracy (+/-)	#Terms (+/-)
WSBA [12]	81.74 (0.08)	32.00 (0.00)
FRANTIC II	76.90 (7.72)	1.90 (0.21)
Fuzzy SH [4]	69.51 (0.07)	4.45 (0.37)
FRANTIC I	69.06 (6.86)	2.29 (0.40)

WSBA [12] achieves the highest classification accuracy on the water treatment data, for this particular partitioning of the data.

Chen *et al*'s algorithm [4] requires the setting of two different parameters α and β. α is a threshold used to determine which linguistic terms should be present in a rule antecedent describing a specific class - terms with a subsethood value equal to or greater than α are selected. If the subsethood values for the linguistic terms associated with a particular class are all lower than α, then an explicit rule can not be created for this class. Instead, an indirect rule is formed and will fire if the membership value of the instance to be classified is greater than β (e.g. IF $Membership(\text{OK}) < \beta$ THEN OUTCOME is FAULTY). The algorithm was run using all combinations of the following values: 0.5,0.55,...,0.85 for α, and 0.5,0.6,...,1.0 for β. The result produced in Table 4 is the best obtained, with α=0.8 and β=0.5.

FRANTIC was first run as originally introduced in [6] (*FRANTIC* I in Table 4, `minInstPerRule=70`, `constructionThreshold=0.85`). The dataset used in that work was a small artificial one with a fairly equal distribution between the three classes. In the water treatment dataset the number of instances per class is very uneven and this was found to have a detrimental impact on *FRANTIC*'s ability to create accurate rules.

This is due to the way one of the parameters (`minInstPerRule`) has been implemented. The value of this parameter stipulates the minimum number of instances that a rule must cover in the training set. Once set the value is the same for all classes, with the maximum value being bound by the number of instances belonging to the smallest class. For the water treatment data, once the dataset had been partitioned into 10 folds, within the training set there were typically 260 instances in the *OK* class, and 80 in the *Faulty* class.

A simple modification was consequently made to *FRANTIC* - the user may now specify the *proportion* of class instances that a rule must cover (*FRANTIC* II in Table 4, `minInstPerRule=70%`, `constructionThreshold=0.65`). For instance, if `minInstPerRule=70` then a rule describing *OK* will cover at least 70 *OK* instances from the training set, and a rule describing *Faulty* will cover at least 70 *Faulty* instances. If, however, `minInstPerRule=70%` then a rule for *OK* will cover at least 70%*260=182 instances, while a rule for *Faulty* will cover 70%*80=56.

A few exploratory runs of *FRANTIC* suggested that rules with negated terms were more accurately descriptive, so the results presented were obtained using this form of knowledge representation. Table 2 presents some results of stratified 10-fold cross-validations before and after the change to `minInstPerRule` was effected. Although the figures are based on one 10-fold cross-validation, they clearly indicate a general improvement in accuracy when `minInstPerRule` is not bound by the number of instances in the smallest class, but may be tailored for individual classes.

A possible reason for the improvement is sug-

Table 3: *FRANTIC* Parameters

Parameter Name	Description	Value
`noAnts`	Number of ants constructing a solution within an iteration, (line (5)).	10
`noIterations`	Number of iterations per ACO run, (line (4)).	150
`minInstPerRule`	Required during rule construction - minimum number instances in training set that rule must cover (section 3.1).	various
`constructionThreshold`	Used during construction of a rule - sets the value for the threshold below which a rule is considered not to cover an instance in the training set.	various
`fitnessThreshold`	Used during evaluation of a rule - sets the value for the threshold below which a rule is considered not to cover an instance in the training set.	0.5
`classInstUncovered`	Maximum number of class instances that may be left uncovered by a rule, before descriptions for a new class are found, (line (3)). Not applicable for simplified iterative rule learning.	n/a
`removalThreshold`	Used during removal of class instances from the training set between ACO runs, (line (9)) - sets the value for the threshold below which a rule is considered not to cover an instance in the training set. Not applicable for simplified iterative rule learning.	n/a

Table 5: *FRANTIC* I vs *FRANTIC* II Predictive Accuracy - Detailed Results

	FRANTIC I		*FRANTIC* II	
	Rule1 (*OK*)	Rule2 (*Faulty*)	Rule1 (*OK*)	Rule2 (*Faulty*)
%Recall	0.68	0.74	0.81	0.62
%Precision	0.87	0.47	0.88	0.49

gested in Table 5. This provides a breakdown on two of the best accuracies obtained by *FRANTIC* before and after the modification (results in italic in Table 2, which are specific examples from the ten 10-fold cross-validations averaged in Table 4). Recall ($TP/(TP + FN)$) measures how accurate a rule is in classifying instances of its *own* class. Precision ($TP/(TP + FP)$) measures a rule's ability to avoid classifying instances of *other* classes.

It can be seen that there is a significant improvement in the recall value for Rule1. The improvement may be explained by noting that when `minInstPerRule=60`, a rule is required to cover only approximately 21% of the training instances. However, it is reasonable to suppose that if only one rule is used to describe a class, then a more general rule (such as one covering 70% of the class instances) may perform better. This may also partly explain why the recall value of Rule2 decreased - when `minInstPerRule=60`, at least 75% of the class instances were being covered by that singe rule, while only 70% are reequired to be covered when `minInstPerRule=70%`. However, since there are considerably more *OK* instances to classify in a test set than *Faulty* ones, the overall result is improved accuracy. Similar changes to the recall and precision values of the rules were found when analysing the other results in Table 2.

Table 6: *FRANTIC* Ruleset (89.47% Accuracy)

R1	IF SSV-D is NOT Low THEN OUTCOME is OK
R2	IF PH-E is NOT High AND SSV-P is Low AND RD-SED-G is High THEN OUTCOME is FAULTY

Table 7: Fuzzy SH Ruleset (81.08% Accuracy)

R1	IF Q-E is NOT Low AND RD-SED-G is Low THEN OUTCOME is OK
R2	IF Q-E is NOT High AND PH-E is NOT Low AND SSV-P is High AND DQO-D is NOT Low AND SSV-D is NOT Low AND SSV-S isNOT Low AND RD-SED-G is Low THEN OUTCOME is FAULTY

4.4 Rule Comprehensibility

Rule and ruleset comprehensibility may be measured by the number of rules in a ruleset, and the number of conditions in a rule. Generally, the fewer the better on both counts.

All the algorithms tested here produce just one rule per class, so that the focus here is on the number of conditions within a rule. There is a considerable difference in the length of the rules produced by each algorithm, with *FRANTIC* producing the most comprehensible rulesets (see Table 6).

Table 7 illustrates a ruleset produced by [4]. The rules are fairly comprehensible, though not as short as *FRANTIC* rules. It should also be remembered that this algorithm may produce rules described in terms of other rules in the ruleset (Section 4.3), thereby reducing the explanatory power of individual rules.

Table 8: WSBA Ruleset (89.47% Accuracy)

R1	IF Q-E is (0.31*Low OR 1.0*Normal OR 0.44*High) AND PH-E is (0.80*Low OR 1.0*Normal OR 0.54*High) AND DBO-E is (0.62*Low OR 0.47*Normal OR 1.0*High) AND DBO-P is (1.0*Low OR 0.84*Normal OR 0.96*High) AND SSV-P is (0.64*Low OR 1.0*Normal OR 0.73*High) AND PH-D is (1.0*Low OR 0.44*Normal OR 0.40*High) AND DBO-D is (1.0*Low OR 0.56*Normal OR 0.68*High) AND SSV-D is (1.0*Low OR 0.68*Normal OR 0.45*High) AND PH-S is (0.63*Low OR 0.91*Normal OR 1.0*High) AND SSV-S is (0.67*Low OR 1.0*Normal OR 0.87*High) AND RD-SED-G is (1.0*Low OR 0.44*High) THEN OUTCOME is OK
R2	IF Q-E is (0.51*Low OR 1.0*Normal OR 0.38*High) AND PH-E is (0.31*Low OR 1.0*Normal OR 0.60*High) AND DBO-E is (0.72*Low OR 0.57*Normal OR 1.0*High) AND DBO-P is (1.0*Low OR 0.59*Normal OR 0.71*High) AND SSV-P is (0.00*Low OR 0.08*Normal OR 1.0*High) AND PH-D is (1.0*Low OR 0.60*Normal OR 0.50*High) AND DBO-D is (0.25*Low OR 0.51*Normal OR 1.0*High) AND SSV-D is (0.24*Low OR 0.45*Normal OR 1.0*High) AND PH-S is (0.82*Low OR 1.0*Normal OR 0.87*High) AND SSV-S is (0.16*Low OR 0.36*Normal OR 1.0*High) AND RD-SED-G is (1.0*Low OR 0.35*High) THEN OUTCOME is FAULTY

Table 8 presents the ruleset produced by WSBA. The fuzzy quantifiers attached to each condition allow the rules to be highly accurate, but also results in rather long rules.

5 Conclusions and Future Work

This paper has demonstrated *FRANTIC*'s ability to find a good balance between ruleset predictive accuracy and ruleset comprehensibility. Although much analysis remains to be done, this ability may be partly due to the generalisation and specialisation capabilities both present within the rule construction mechanism. The parameter `minInstPerRule` may be used for generalisation (the higher the value, the more general a rule), whilst the parameter `constructionThreshold` may be used for specialisation (the higher the value the less likely many rule conditions will be added to the rule).

FRANTIC may have clear advantages when inducing fuzzy rules from real-world datasets. Only a minor modification was required to enable it to deal with imbalanced datasets, and it is expected that only a minor modification would be required to enable *FRANTIC* to use instances with missing attribute values. This would ensure that the best possible use is made of available data and may be accomplished by using such instances during rule evaluation, to provide a more accurate measure of how well a rule performs.

Another advantage is the ability to operate in full iterative rule learning mode which means that as many rules as necessary are created to describe a class. This is likely to be necessary with large datasets, or even imbalanced datasets, when the assumption that one rule to describe a class is adequate, may be an invalid one.

However, the majority of the work is expected to lie in adopting a two-part strategy aimed at improving predictive accuracy, while maintaining comprehensibility. The first part involves improving the richness and flexibility of the knowledge representation used. In a similar way to how the rule construction mechanism may be adapted to produce simple propositional rules or rules with negated terms. For instance, it may be extended to include linguistic hedges [10]. This may improve rule accuracy without negatively impacting on rule comprehensibility.

The second part arises out of a potential shortcoming of fuzzy rule induction using iterative rule learning, as highlighted in [6]. This is because a final ruleset may well have individual rules that classify correctly within their own class, but when combained with the rest of the ruleset may result in suboptimal classification. In fact, when each rule describing a particular class is created, no consideration is taken of other rules that may already be present in the ruleset, or even of future rules. This suggests that an approach whereby the fuzzy rules are evolved simultaneously, and evalutated together, might yield better results.

The authors will therefore also be working on extending *FRANTIC* to adopt an alternative strategy to that of iterative rule learning - by evolving the fuzzy rules of a ruleset simultaneously. This may be accomplished by running several ACO algorithms in parallel, with each finding rules for one class.

References

[1] C.L. Blake and C.J. Merz, *UCI Repository of Machine Learning Data* Deparatment of Computer Science, University of California, Irvine CA, 1998. `http://www.ics.uci.edu/m̃learn/MLRepositary.html`

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

[3] J. Casillas, O. Cordon, and F. Herrera, "Learning Fuzzy Rules using Ant Colony Optimization Algorithms" *Proc. 2nd International Workshop on Ant Algorithms* pp. 13–21, 2000.

[4] S-M. Chen, S-H. Lee, and C-H Lee, "A New Method for Generating Fuzzy Rules from Numerical Data for Handling Classifi cation Problems" *Applied Artificial Intelligence* vol. 15 pp. 645–664, 2001.

[5] M. Dorigo, V. Maziezzo, ad A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Ants" *IEEE Trans. Systems, Man*, and *Cybernetics B* vol. 26 pp. 29–46, 1996.

[6] M. Galea and Q. Shen, "Fuzzy Rules from Ant-Inspired Computation" *Proc. IEEE International Conf. on Fuzzy Systems*, Budapest, July 2004.

[7] R. Jensen and Q. Shen, "Fuzzy-Rough Attribute Reduction with Application to Web Categorization" *Fuzzy Sets and Systems* vol. 141 pp. 469–485, 2004.

[8] B. Kosko, "Fuzzy Entropy and Conditioning" *Information Sciences* vol. 40 pp. 165–174, 1986.

[9] P. Kuntz, P. Layzell, and D. Snyers, "A Colony of Ant-Like Agents for Partitioning in VLSI Technology" *Proc. 4th European Conference on Artificial Life* pp. 417–424, 1997.

[10] J.G. Marin-Blazquez and Q. Shen, "From Approximative to Descriptive Fuzzy Classifi ers" *IEEE Trans. Fuzzy Systems* vol. 10 pp. 484–497, 2002.

[11] R. Parpinelli, H. Lopes, and A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm" *IEEE Trans. Evol. Comput.* vol. 6 pp. 321–332, 2002.

[12] K. Rasmani and Q. Shen, "Modifying Weighted Fuzzy Subsethood-Based Rule Models with Fuzzy Quantifi ers" *Proc. IEEE International Conf. on Fuzzy Systems*, Budapest, July 2004.

[13] Q. Shen and A. Chouchoulas, "A Rough-Fuzzy Approach for Generating Classifi cation Rules" *Pattern Recognition* vol. 35 pp. 2425–2438, 2002.

[14] Q. Shen and R. Jensen, "Selecting Informative Features with Fuzzy-Rough Sets and its Application for Complex Systems Monitoring" *Pattern Recognition* vol. 37 pp. 1351–1363, 2004.