Linguistic hedges for ant-generated rules

Shen, Qiang; Galea, Michelle

DOI:
10.1109/FUZZY.2006.1681974

Publication date:
2006

Citation for published version (APA):
Shen, Q., & Galea, M. (2006). *Linguistic hedges for ant-generated rules*. 9105-9112. https://doi.org/10.1109/FUZZY.2006.1681974

2006 IEEE International Conference on Fuzzy Systems
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Linguistic Hedges for Ant-Generated Rules

Michelle Galea, *Student Member, IEEE*, and Qiang Shen

***Abstract*— *FRANTIC*, a system inspired by insect behaviour for inducing fuzzy IF-THEN rules, is enhanced to produce rules with linguistic hedges. *FRANTIC* is evaluated against an earlier version of itself and against several other fuzzy rule induction algorithms, and the results are highly encouraging. Rule comprehensibility is maintained while an improvement in the accuracy of the rulebases induced is observed. Equally important, the increase in computation expense due to the improved richness in the hypothesis language is acknowledged, and several ways of resolving this are discussed.**

I. Introduction

In industry there is often a requirement to not only monitor, control and predict operational systems, but to also understand the conditions giving rise to their various possible states. This necessitates that any model used to explain and predict their operation must be human-comprehensible. However, it is often the case that model comprehensibility is obtained at a cost to its predictive capability, and vice versa.

This paper builds on earlier work by Galea & Shen to induce models from empirical data that are both accurate and human-comprehensible. In [1] and [2] the authors introduced *FRANTIC* (Fuzzy Rules from ANT-Inspired Computation)—a system for inducing fuzzy rulebases that uses Ant Colony Optimization (ACO) [3] as the rule discovery mechanism. The system was compared against several fuzzy induction algorithms and the results obtained highlight its ability to balance the tradeoff encountered between classification accuracy on the one hand, and rulebase comprehensibility on the other—rules induced by *FRANTIC* aid human comprehensibility in that they are very short and do not make use of any numerical quantifiers, while the accuracy of the induced rulebases was in most cases comparable or superior to the accuracy achieved by rulebases produced by the other induction algorithms.

This work suggested two avenues for improving the accuracy of *FRANTIC* rulebases whilst maintaining their high comprehensibility level. The first arises out of the system's original strategy, an iterative rule learning approach that runs several ACO algorithms in succession with each contributing a fuzzy rule that is added to the final rulebase. Fuzzy rules are generated and evaluated independently of each other, and when a choice is made as to which specific rule from an ACO is to be added to the final rulebase, no consideration is taken of other rules already present, or of future rules that may be added, Fig. 1. This instantiation of the *FRANTIC* system is called *FRANTIC-IRL* (-Iterative Rule Learning). As highlighted in [1], it can lead to situations where a case requiring classification is closely matched by two or more fuzzy rules in the final rulebase that have different

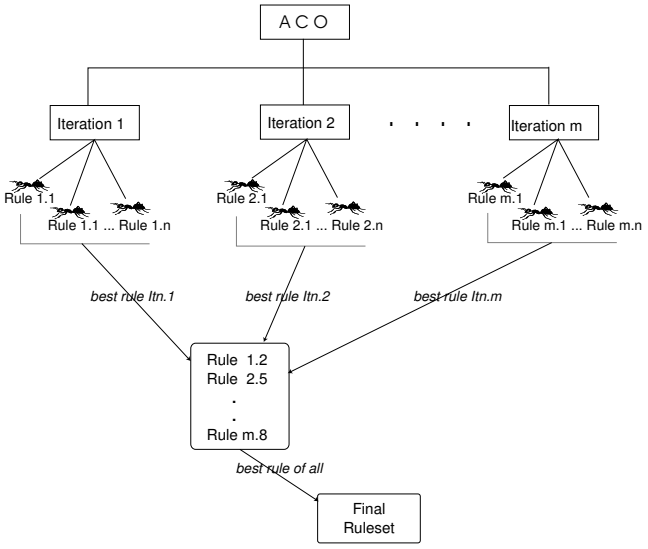


Fig. 1. Overview of a basic rule-inducing ACO algorithm

conclusions. The system was therefore further developed so that the ACO algorithms may be run simultaneously instead of in succession. This meant that rules describing different classes could be combined during the training and evaluation process, and only those rules that interacted optimally were chosen for reinforcement. This version is called *FRANTIC-SRL* (-Simultaneous Rule Learning) and is described in detail in Section III. It was introduced in [4], compared with *FRANTIC-IRL* and the results indicated both comparable or improved classification accuracy, and an increased robustness to parameter value changes.

The second avenue suggested for further work aimed at improving accuracy was the inclusion of linguistic hedges [5] in the IF-THEN rules. This essentially provides the system with a richer hypothesis language with which to build rules, and results in more accurate, yet still very human-comprehensible rules to describe the data. This paper presents initial findings resulting from this development to *FRANTIC*'s rule construction mechanism, demonstrating the increased accuracy of the rules constructed using linguistic hedges, yet also discussing the resultant computational expense due to the richer hypothesis language, and indicating several ways in which this may be resolved.

Since the application of ACO to rule induction is still relatively unexplored, the next section introduces this topic. Section III describes *FRANTIC-SRL* and how linguistic hedges are utilised. It should be emphasised that the extension to linguistic hedges is applicable to both *FRANTIC-IRL* and *FRANTIC-SRL* since the rule construction mechanism is identical in both instantiations of the system—they differ in the overall strategy and evaluation process. Due to the

Michelle Galea is with the Centre for Intelligent Systems and their Application, School of Informatics, University of Edinburgh, Edinburgh EH8 9LE, UK (email: m.galea@sms.ed.ac.uk).

Qiang Shen is with the Department of Computer Science, University of Wales, Aberystwyth SY23 3DB, UK (email: qqs@aber.ac.uk).

0-7803-9489-5/06/$20.00/©2006 IEEE

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

Fig. 2. Generic ACO algorithm

advantages reported in [4] *FRANTIC-SRL* was used to carry out the work reported here. However, for ease of reference, from now on the system is often referred to as *FRANTIC*.

Section IV describes the datasets used for experiments in this work, and the algorithms against which *FRANTIC* is compared. The main results are discussed in Section V, whilst conclusions and avenues for future work are presented in Section VI.

II. Rule Induction via Ant Colony Optimization

ACO is an agent-based heuristic for combinatorial optimization motivated by the ability of real ants to find the shortest path between their nest and a food source. This is attributed to the fact that ants lay a chemical substance, called a pheromone, along the paths they take, and when presented with a choice between alternative paths, they tend to choose the one with the greatest amount of pheromone. Pheromone, however, evaporates so that over time the shortest path accrues more pheromone as it is traversed more quickly.

A high-level description of an ACO-based algorithm is given in Fig. 2. An appropriate problem representation is required that allows an artificial ant to incrementally build a solution using a probabilistic transition rule. The problem is modelled as a search for a best path through a graph (here referred to as a problem graph). In the context of rule induction a solution may be 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.

The probabilistic transition rule determines which node an ant should visit next, and this is dependent on the heuristic value and the pheromone level associated with a node. The heuristic provides local guidance to an ant in choosing the next node for the path (solution) it is building, and a fitness function determines the quality of the solution built. A pheromone update rule then specifies how to modify the pheromone levels of each node in the graph between iterations of an ACO algorithm. More detail about how an ACO is used for rule induction is provided in Section III.

The general appeal of such insect-inspired 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—may be readily borrowed from existing literature on rule induction; and, an explicit heuristic embedded in the solution construction mechanism makes for easy insertion of domain knowledge.

There are additional advantages specific to rule induction. As will be demonstrated in later sections, the constructionist nature of ACO allows for simple effective mechanisms within the rule discovery process that enable it to cope with imbalanced datasets and prevent over-fitting to the training data, whilst the strategy of running several ACO algorithms simultaneously encourages a fuzzy rulebase optimized for rules that are complementary to, rather than competitive with, each other.

The application of ant-inspired algorithms to rule induction is a relatively recent area of research, but is gaining increasing interest. A first attempt to apply ACO to fuzzy modelling is found in [7], and in this work the ACO algorithm is used for assigning rule conclusions to pre-determind rule antecedents that act as nodes of the problem graph—an ant traverses the graph, visiting each and every node and probabilistically assigns a rule conclusion to each.

Parpinelli *et al*. [8] introduced *Ant-Miner*, a system using ACO algorithms 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 walks round the graph selecting nodes and building its rule antecedent. The rule conclusion is assigned afterwards by a deterministic method. The strategy used is one of iterative rule learning and the final result is an ordered rule list. Interest in *Ant-Miner* has resulted in various modifications to it, and applications to different problem domains (e.g. [9], [10]).

A recent application of ACO to fuzzy modelling is [11], where simple propositional fuzzy rules are pre-determined and act as nodes of the problem graph. Each ant then attempts to build a compact rulebase by selecting some of the fuzzy rules and making them more general by including additional attribute values in the antecedent (e.g. TEMPERATURE $=$ $Mild$ might become TEMPERATURE $\geq$ $Mild$).

FRANTIC follows a rule construction mechanism similar to that of *Ant-Miner*, but the knowledge representation used has been greatly enriched, and the strategy specifically developed for the induction of fuzzy rules.

III. *FRANTIC-SRL*

FRANTIC-SRL runs several ACO algorithms in parallel, with each maintaining its own problem graph, pheromone levels and heuristic values. The ACO algorithms are run simultaneously in principle, i.e. this is not as yet a parallel implementation running on multiple processors.

An overview of the system is provided in Figure 3. After each class has had its rules created for a particular iteration, all possible combinations of rules (one from each class) are formed into a rulebase and this is tested on the training set. The rules in the best performing rulebase are used to update the pheromone levels, with the rule describing a specific class being used to update the pheromone levels of the associated ACO. The following subsections detail the rule construction and rule evaluation processes.

A. Rule Construction

FRANTIC was implemented with the flexibility to create simple propositional rules (e.g. *IF* TEMPERATURE *is Cool AND* WIND *is Windy THEN Weightlifting*), propositional rules with internal disjunction (e.g. *IF* TEMPERATURE *is Cool* OR *Mild AND* WIND *is Windy THEN Weightlifting*), and propositional rules that include negated terms (e.g. *IF*

```
(1) for numIterations
(2)   for each class
(3)      each ant constructs rule
(4)   for each combined rulebase
(5)      evaluate rulebase
(6)   update pheromone levels
(7) output best rulebase
```

Fig. 3. Simultaneous rule learning for fuzzy rules—*FRANTIC-SRL*

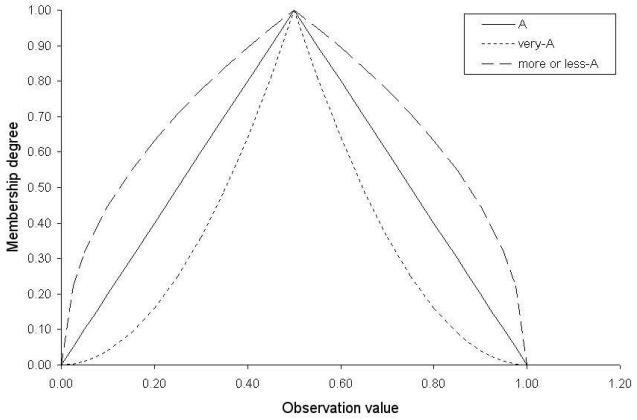


Fig. 4. Impact of linguistic hedges on a fuzzy set A

TEMPERATURE *is* Not *Hot AND* WIND *is Windy THEN Weightlifting*).

It is this design flexibility within the rule construction process that has been exploited further to introduce linguistic hedges in rules, e.g. *IF* WIND *is* Very *Windy THEN Weightlifting*. The use of the linguistic hedge 'Very' here, acts as a function that is applied to the fuzzy term 'WIND *is Windy*' in order to increase its precision. This and other linguistic hedges that may be used during the rule construction process provides an artificial ant with a richer knowledge representation language, enabling it to more accurately describe the underlying patterns in the data. It should be noted though, that generally, the more expressive the knowledge representation language, the larger is the search space. The impact of the use of linguistic hedges on the computation performance is discussed in Section V-B.

As a first step in testing the use of hedges in improving classification accuracy, two of the more common ones have been utilised in this paper: the hedges 'Very' and 'More or less'. These are based on the fuzzy set operators of concentration and dilation respectively [5, Part I:226, Part II:322]:

$$CON(A) = A^2, \quad DIL(A) = \sqrt{A}$$

where A is a fuzzy set and the CON and DIL operators cause the degrees of membership to decrease or increase respectively. The impact on a simple triangular membership function of these hedges is depicted in Fig. 4.

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. If linguistic hedges are made available to the artificial ants in creating their rules, then additional nodes are present in the graph, e.g. OUTLOOK=Very-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.

After selection but before a term is added to a rule antecedent, a check is made—this ensures that the resultant partial rule antecedent covers a minimum number of instances from the training set (set by a parameter called `minInstPerRule`), and is a simple and effective way of avoiding over-fitting to the training data. 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 done in Section III-C.

For simple propositional rules, and rules with negated terms and/or linguistic hedges, 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 (e.g.OUTLOOK=*Rain* OR *Cloudy*).

1) Heuristic: The heuristic used to guide ants when selecting terms is based on fuzzy subsethood values [12], giving a degree to which one fuzzy set *A* is a subset of another fuzzy set *B*:

$$(A, B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u \in U} min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)}$$

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—gives a measurement of how important that term is in describing a *specific* class. If there are n class labels in a dataset, j will therefore have n heuristic values associated with it in total. An ACO finding rules to describe a particular class will use the appropriate term heuristic values, i.e. those associated with the class.

The heuristic value for a negated term is the complement of the heuristic value for the non-negated term, i.e. $\eta_{NOT\_j} = 1 \quad \eta_j$, whilst the heuristic values for terms with linguistic hedges are $\eta_{V\ ry\_j} = {\eta_j}^2$ and $\eta_{Mor\ \ or\ l\ ss\_j} = \sqrt{\eta_j}$.

2) Pheromone Updating: Unlike most other ACO implementations, the pheromone here is deposited on the nodes, and not the edges of the graph. This is because it is the actual nodes (terms) themselves that are important in constructing the rule antecedent, and not the order in which they are selected (as opposed to, say, the travelling salesman problem where the order in which the cities are visited is relevant). For instance, the rule

$$IF\ \text{TEMPERATURE}\ is\ Mild\ AND\ \text{WIND}\ is\ Windy\ THEN\ Weightlifting$$

is equivalent to the rule

IF WIND $is\ Windy\ AND$ TEMPERATURE $is\ Mild$
$THEN$ *Weightlifting*

At the start of an ACO run 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. At the end of each iteration rules created by all ants are evaluated. The terms in the best rule, say rule *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 quality of the rule *Q* (defined in section III-B). 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 therefore 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) 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 probability that ant *m* selects term *j* when building its rule during iteration *t* is given by:

$$P_j^m(t) = \frac{[\eta_j] \times [\tau_j(t)]}{\sum_{i \in I_m} [\eta_i] \times [\tau_i(t)]}$$

where I_m is the set of terms that may still be considered for inclusion in the rule antecedent being built by ant *m*.

If propositional rules with internal disjunction are being created, then I_m will exclude terms that are already present in the current partial rule antecedent, and terms that have already been considered but found to decrease coverage of the training set below the required number of instances (as set by `minInstPerRule`). If simple propositional rules, or rules that include negated terms or terms with linguistic hedges are being created, then I_m will further exclude other values within the domain of linguistic variables that already have a term present in the rule antecedent.

The probabilistic nature of the transition 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.

B. Rule Evaluation

Each constructed rule needs to be evaluated and this is done by assessing how accurate it is in classifying the training instances. However, instead of evaluating each rule separately, at the end of each iteration when each class has produced its set of rules, a rule describing one class is combined with one rule describing each of the other classes and together they classify the training set (Fig. 3 lines(4)–(5)).

The method of classification used during evaluation is a single winner-based method [13]. Specifically, for each instance u:

1) for each rule, determine the degree of match between the rule and u (defined in Section III-C);
2) assign to u the class of the rule with the highest degree of match.

The accuracy obtained by a rulebase on the training set is used as a measure of the quality, *Q*, of each rule within the rulebase. The rules in the rulebase obtaining the highest accuracy are the ones used for updating the pheromone levels in the various ACO algorithms before the next iteration is run.

Each rule from each class is combined with every other possible rule from the other classes, so that the total number of *ruleset* evaluations conducted during SRL is $(numItns * numAnts^{numClass\ s})$, where $numItns$ is the number of iterations run by an ACO, $numAnts$ is the number of ants used within an iteration, and $numClasses$ is the number of class labels in the dataset.

C. Fuzzy Rule Matching

Finding the degree of match between a fuzzy rule and an instance is required during rule construction and rule evaluation.

As previously stated, while an ant is constructing a rule it ensures that the rule covers a minimum number of training instances. However, what constitutes coverage of a fuzzy instance by a fuzzy rule needs defining. A fuzzy rule describing a specific class is said to cover a fuzzy instance if:

1) the rule and instance belong to the same class; and,
2) the degree of match between the condition parts of rule and instance is equal to or greater than a pre-defined value, here called a threshold value.

An example follows. Consider a rule *R* that describes the conditions leading to a decision to do *Weightlifting* (the underlying dataset is described in Section IV-A):

IF TEMPERATURE $is\ Cool$ OR $Mild\ AND$ WIND $is\ Windy$
$THEN$ *Weightlifting*

For the purpose of illustrating how a condition match may be determined, a more convenient representation of the rule is used: *R*=(0,0,0; 0,1,1; 0,0; 1,0; 0,0,1). This means that there are five attributes, the first four being condition attributes with three or two values (terms) in the domains, and the last representing the class attribute with three possible values (*Volleyball*, *Swimming* and *Weightlifting* respectively). Terms that are present in the rule are denoted by 1, others by 0. These rules may only classify instances into one class. However, there may be more than one specific attribute value present in a rule (i.e. propositional rules with internal disjunction).

Consider now a fuzzy instance *u*=(0.9,0.1,0.0; 0.0,0.3,0.7; 0.0,1.0; 0.9,1.0; 0.0,0.3,0.7). The representation is similar to rule *R*, though the value for each term represents the degree of membership and lies in the range [0,1]. Note that the conclusion attribute values may be greater than 0 for more than one class, that an instance is considered to belong to the class with the highest degree of membership, and in

this case, the class is *Weightlifting*. The rule and instance therefore belong to the same class and so condition 1) above is satisfied.

The degree of condition match between a rule *R* and an instance *u* is given by

$$mC\ nd(R,u) = Min_k(mAtt(R_k,u_k))$$

In the above $mAtt(R_k,u_k)$ measures the degree of match between an attribute *k* in *R* and the corresponding attribute in *u*:

$$mAtt(R_k,u_k) = \begin{array}{r} 1 \ : \ R_k\ empty \\ Max_j(Min(\mu_j(R_k),\mu_j(u_k))) \ : \ otherwise \end{array}$$

where $R_k\ empty$ indicates that no term from the domain of attribute *k* is present in rule *R*, and *j* is a specific term within the domain of attribute *k*. If the attribute is not represented at all in the rule, the interpretation is that it is irrelevant in making a particular classification.

From the rule and instance examples above the attribute matches are: $mAtt(R_1,u_1) = 1.0$, $mAtt(R_2,u_2) = 0.$, $mAtt(R_3,u_3) = 1.0$ and $mAtt(R_4,u_4) = 0.$, with the condition match therefore $mC\ nd(R,u) = 0.$. If the threshold value is set at 0.7 or below, then *R* is considered to cover *u*. If the threshold value is set above 0.7, then *R* is considered to not sufficiently match *u*.

Finding the condition match for a fuzzy instance and a fuzzy rule is also necessary during rule evaluation, as described in Section III-B.

IV. Experimental Preliminaries

A. The Datasets and Other Algorithms

The first dataset is the Saturday Morning set originally used for the induction of crisp decision trees [14]. A fuzzified version of this dataset has been used by several fuzzy induction algorithms and so allows a direct comparison between these algorithms and *FRANTIC*. The dataset has 16 instances, 4 condition attributes and 1 class attribute called PLAN:
OUTLOOK={Sunny,Cloudy,Rain},
TEMPERATURE={Hot,Cool,Mild},
HUMIDITY={Humid,Normal},
WIND={Windy,Not-Windy},
PLAN={Volleyball,Swimming,Weightlifting}.

The second dataset is more challenging and is the Water Treatment Plant database [15]. The database contains the daily observations of 38 sensors monitoring the operation of an urban waste water treatment plant, with the objective being to predict faults in the process. Observations were taken over 527 days and are real-valued. There are 13 possible classifications for each daily set of observations, with many assigned to only a few records in the database. When faults are reported these are generally fixed very quickly and so the database contains a disproportionate number of records indicating correct operation of the plant, versus faulty operation.

The 13 classifications have been collapsed to two: *Normal* and *Faulty*, as in [17]. 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 (98% *Normal*, 2% *Faulty*). Other pre-processing steps included fuzzification of the features using trapezoidal functions into two (*low, high*) or three (*low, high, normal*) linguistic terms, and a feature subset selection process [16] to reduce the number of features (better accuracy was indicated in [17] with the reduced dataset). A description of the retained features is shown in Table I.

TABLE I
WATER TREATMENT PLANT DATABASE FEATURES

Name	Sensor Description
COND-E	Input to plant – conductivity
PH-D	Input to secondary settler – pH
DBO-D	Input to secondary settler – biological demand of oxygen
SED-S	Output from plant – sediments
RD-SED-G	Global performance, input – sediments

TABLE II
FRANTIC-SRL PARAMETERS

Parameter	SM	WT
`numIterations` – number of iterations per ACO run	25	30
`numAnts` – number of ants within an iteration constructing a solution	4	10
`minInstPerRule` – minimum number of instances in training set that a rule must cover	4	70%
`constructionThreshold` – value for the threshold below which a rule is considered not to cover an instance in the training set	–various–	

The fuzzy rule sets generated by *FRANTIC* are compared against those produced by a fuzzy decision tree algorithm (*FDT*) [18], a fuzzy genetic algorithm (*FGA*) [19], and two methods based on fuzzy subsethood values (*QSBA* [20], *FSBA* [21])—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, *FSBA* requires the setting of two parameters and so may produce different rulesets depending on their setting. Apart from *QSBA*, the algorithm acronyms are not the names given to the algorithms by the original authors, but are introduced here for ease of reference.

Examples of the rulebases produced by these algorithms are provided in Section V.

B. FRANTIC-SRL Parameters

Parameters that require setting are listed in Table II, together with a brief description and the values given in order to obtain the results reported here for the two datasets—the Saturday Morning Problem dataset (SM) and the Water Treatment Plant database (WT). The values for the SM dataset are as in [1] while those for the WT database are as in [4]—little parameter tuning has been done so it is quite possible that better results may be obtained.

The `minInstPerRule` parameter mentioned in Section III-A is flexible enough so that different values may be given to different classes. This is particularly useful in imbalanced datasets (such as the WT one) where stipulating the same number of instances that a rule must cover for a small class as for a large class is impractical. The value '4' therefore means that for each class a rule must cover at least 4 class

instances from the training set, whilst the value '70%' means that a rule should cover at least 70% of the class instances.

Both `minInstPerRule` and `constructionThreshold` have been implemented so that their values may change automatically, if necessary, during the running of an experiment. For instance, it is generally the case that the actual number of instances belonging to a particular class in the training set is equal to or greater than the value set by `minInstPerRule`. On the other hand, `constructionThreshold` may be set so high that no ant is able to construct a rule that covers the required number of class instances to the specified degree of match. In this case, the values of `minInstPerRule` and/or `constructionThreshold` may be automatically and gradually reduced until rules describing the class may be generated. The adaptive nature of these parameters provides the system with a useful degree of autonomy that reduces the need for unnecessary user intervention.

V. Main Results

The next subsection details the results obtained when running *FRANTIC* with and without the capability to introduce linguistic hedges into the rules, while the following subsection discusses various ways of resolving the increase in computation time due to their use. The final subsection compares the *FRANTIC* results obtained through the use of linguistic hedges with the other fuzzy induction algorithms mentioned in Section IV-A.

A. Impact of Linguistic Hedges on Accuracy

Table III presents the results obtained using different construction threshold values for the previous version of *FRANTIC* (hedges), and the enhanced version that includes linguistic hedges (+hedges).

For the SM dataset, each result is the average of the accuracies obtained from 30 *FRANTIC* runs on the training set. The figure in brackets is the standard deviation based on the predictive accuracies. It is clear that there is a general improvement in the accuracy obtained by rulebases that have rules with linguistic hedges—this is seen over the range of values used for the `constructionThreshold` parameter.

The increase or decrease effect of the linguistic hedges on the precision of the conditions in the rule antecedent have enabled the system to more accurately describe the underlying dataset, and the same general improvement may be seen in the WT results, where each each result is the average of ten 10-fold cross-validations. The figure in brackets is the standard deviation of the ten predictive accuracies of a 10-fold cross-validation, averaged over all ten cross-validations.

There appears to be no clear and obvious change in the standard deviation, from running *FRANTIC* with or without linguistic hedges, and this requires a more thorough investigation to confirm the impact of enriching the hypothesis language on the robustness and consistency of the overall system.

B. Impact of Linguistic Hedges on Computation

FRANTIC rules that include negated terms take twice as long to generate as simple propositional rules or rules with internal disjunction between attribute values, since the number of nodes in the problem graph is double. With the addition of two possible linguistic hedges, therefore, including the use of negated terms, rules may take up to four times as long to generate. For datasets with a small number of attribute-values (such as the SM and WT datasets), the increase in computation time may be insignificant. Yet, in general, ways of alleviating the increase in computational expense should be considered. There are several possibilities.

FRANTIC was re-run on the SM and WT datasets with a change in the value for the `numAnts` parameter. For the SM dataset this parameter was reset equal to 1, 2 and 3. For the WT dataset it was set to 2 and 6. Experiments were rerun once generating rules using only negated terms, and another time using negated terms *and* linguistic hedges. As might be expected, the greater the number of ants within an iteration the greater the opportunity for the system to find good rules, and the resultant accuracy tended to improve as the number of ants increased (both when linguistic hedges were used, and when they were not). However, the results also suggest that if a richer hypothesis language is used, then fewer ants are required in order to create rulebases with comparable accuracy, than ants using a less rich knowledge representation. This is illustrated in Figs. 5 and 6—the same accuracy as that achieved by rules created by ants with a less-rich knowledge representation, can be achieved or surpassed by fewer ants with a more expressive knowledge representation at their disposal. This is observed over a range of values for the `constructionThreshold` parameter.

Fewer ants within an iteration saves time not only dur-

TABLE III
Impact of hedges on *FRANTIC* rulebase accuracy

Construction Threshold	Saturday Morning		Water Treatment	
	hedges	+hedges	hedges	+hedges
0.45	92.08 (3.7)	95.21 (5.6)	76.53 (9.5)	80.52 (9.1)
0.50	93.13 (1.9)	92.08 (5.7)	77.24 (9.6)	82.34 (7.3)
0.55	92.71 (3.7)	98.96 (2.4)	75.44 (10.0)	84.58 (6.8)
0.60	93.13 (3.4)	97.50 (4.5)	76.71 (10.6)	81.80 (9.2)
0.65	93.33 (1.6)	93.33 (1.6)	76.66 (10.8)	81.72 (8.9)
0.70	91.67 (5.8)	93.33 (1.6)	75.58 (8.4)	85.00 (9.0)
0.75	68.75 (0.0)	93.75 (0.0)	75.53 (8.4)	85.08 (8.6)
0.80	68.75 (0.0)	92.50 (3.8)	76.93 (5.0)	85.25 (8.8)
0.85	31.25 (0.0)	68.75 (0.0)	79.02 (5.9)	79.31 (5.6)
0.90	31.25 (0.0)	31.25 (0.0)	82.78 (6.1)	79.18 (5.9)
0.95	25.00 (0.0)	25.00 (0.0)	82.78 (6.1)	82.78 (6.1)

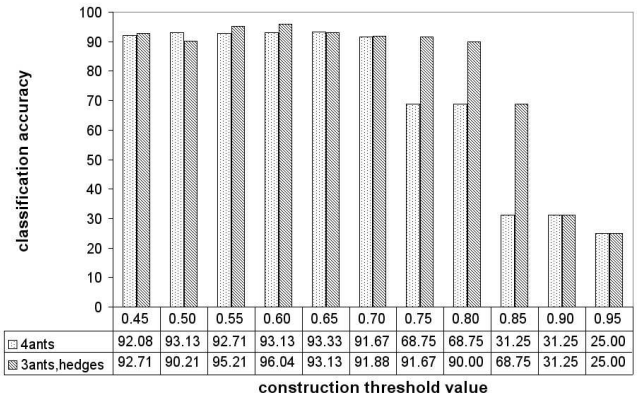
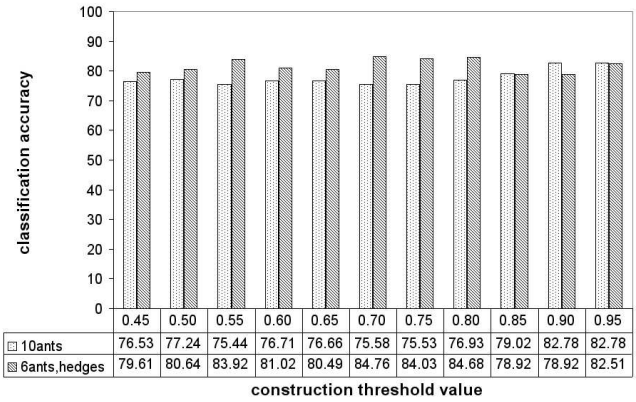


Fig. 5. SM dataset—impact of hedges on number of ants and accuracy



	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
10ants	76.53	77.24	75.44	76.71	76.66	75.58	75.53	76.93	79.02	82.78	82.78
6ants,hedges	79.61	80.64	83.92	81.02	80.49	84.76	84.03	84.68	78.92	78.92	82.51

Fig. 6. WT dataset—impact of hedges on number of ants and accuracy

TABLE IV
SATURDAY MORNING DATASET—COMPARISON OF ALGORITHMS

Algorithm	%Accuracy	#Rules	#Terms
FDT	81.25	6	1.7
FGA	87.50	5	3.2
FSBA	93.75	3	2.3
FRANTIC	98.96	3	2.7

TABLE V
FSBA RULEBASE FOR SATURDAY MORNING DATASET

R1	IF OUTLOOK is Not-Rain AND HUMIDITY is Normal AND WIND is Not-windy THEN *Volleyball*
R2	IF OUTLOOK is Not-Rain AND TEMPERATURE is Hot THEN *Swimming*
R3	IF $MF(R1) < \beta$ AND $MF(R2) < \beta$ THEN *Weightlifting*

ing the rule construction phase, but also during the rule evaluation phase—fewer ants per class means that fewer rulebases are formed and therefore need evaluation. It may be possible to reduce the number of evaluations even further—currently, *all* possible rulesets are created and evaluated after an iteration, by combining a rule from one class (ACO), with one rule from each of the other classes. In work using multi-population co-evolution to induce both a rulebase and associated membership functions, however, not all possible combinations are formed. Generally, only a few representatives from each population are used to form different knowledge bases (i.e. a rulebase and membership functions). The representatives may be chosen according to fitness, randomly, or a combination of both, and this suggests a useful avenue of further investigation.

Another possibility arises from *FRANTIC*'s ability to generate very short rules. Analysis of *FRANTIC* results indicates that the average number of terms per rule is fairly constant throughout all iterations. A procedure could therefore be implemented that determines the average number of terms within a rule for the first several iterations, and in subsequent iterations ants could stop building a rule antecedent when they have reached the average value. The actual number of iterations may also be decreased dynamically—*Ant-Miner* has a feature by which an ACO algorithm stops if the best n ants from the previous n successive iterations are identical. Again, analysis of *FRANTIC* results indicate that the highest rule fitness is generally achieved well before the final iteration and this fitness is generally replicated in subsequent iterations. Reference [22] also reports a significant speed up in computation when a different transition rule is used in *Ant-Miner*—a pseudo-random transition rule [23] allows ants to occasionally select terms in a deterministic manner, i.e. to exploit acquired information rather than explore continuously and select each term probabilistically. The resulting rulebases have comparable performance to those produced using the more common transition rule described in Section III-A.3.

A final advantage offered by *FRANTIC-SRL* that should not be ignored is the numerous opportunities for a multiple-processor implementation—at a coarse level of granularity several ACOs may be run truly in parallel, as may the rulebase evaluations; at a finer level of granularity the numerous ants of each ACO may create their rules simultaneously.

C. Comparative Studies

A summary of the results produced on the SM dataset by various algorithms is provided in Table IV—it gives the percentage classification accuracy on the training set, the number of rules generated, and the average number of conditions in a rule antecedent.

The accuracy of only one rulebase is reported for *FGA* in [19], and reproduced in Table IV, and so the assumption here is that it is the best rulebase obtained. The results for *FSBA* reported in [21] are for the parameter settings $\alpha = 0.$ and $\beta = 0.6$, and the assumption again is that this is the best obtainable. α is a threshold used to determine which linguistic terms should be present in a rule antecedent describing a specific class, and terms with a subsethood value equal to or greater than α are selected. If the subsethood values for all the terms associated with a particular class are lower than α, then an explicit rule can not be created. Instead, an indirect rule is formed and will fire if the membership of the instance to be classified is less than β for the other classes. See Table V for the best rulebase produced by *FSBA*.

The *FRANTIC* results are the best obtained using linguistic hedges from Table III. With only 5 out of the 30 runs obtaining below 100% accuracy, the standard deviation is 2.4, making the overall accuracy easily comparable with that obtained by *FSBA*. With regards to rulebase comprehensibility in terms of the number of rules and number of conditions in a rule, *FSBA* and *FRANTIC* overall provide the smallest number of conditions in a rulebase, making their output easier to assimilate. However, as illustrated in Table V, the third rule produced by *FSBA* has no explanatory power of its own.

The middle column of Table VI indicates the average accuracy obtained by several algorithms after performing stratified 10-fold cross-validation on the WT dataset. The same folds of the dataset were used for each algorithm, and the stratification ensures that each fold contains approximately the same proportions of instances of the different classes as the original complete dataset does. The figure in brackets is the standard deviation of the accuracies of the ten rulebases produced. The right column gives the average number of terms per rule, with standard deviation in brackets. All these algorithms generate just one rule to describe each class.

TABLE VI

WATER TREATMENT DATABASE—COMPARISON OF ALGORITHMS

Algorithm	%Accuracy	#Terms
QSBA	77.96 (11.4)	13.0 (0.0)
FSBA	66.39 (16.8)	2.0 (0.3)
FRANTIC	85.25 (8.8)	2.0 (0.0)

TABLE VII

QSBA RULEBASE FOR WATER TREATMENT DATABASE

R1	IF COND-E is (0.34*Low OR 0.45*Normal OR 0.21*High) AND PH-D is (0.22*Low OR 0.16*Normal OR 0.54*High) AND DBO-D is (0.39*Low OR 0.27*Normal OR 0.34*High) AND SED-S is (0.98*Low OR 0.02*High) AND RD-SED-G is (0.35*Low OR 0.84*High)) THEN OUTCOME is NORMAL
R2	IF COND-E is (0.21*Low OR 0.14*Normal OR 0.45*High) AND PH-D is (0.40*Low OR 0.27*Normal OR 0.20*High) AND DBO-D is (0.60*Low OR 0.00*Normal OR 0.40*High) AND SED-S is (0.80*Low OR 0.20*High) AND RD-SED-G is (0.26*Low OR 0.78*High) THEN OUTCOME is FAULTY

The *FRANTIC* result is the best obtained using linguistic hedges from Table III and is the average of *ten* 10-fold cross-validations. *FSBA* was run using all combinations of values for α and β from the range [0.5,1] with a step value of 0.05. The results reported here are the best obtained with $\alpha = 0.$ and $\beta = 0.$. *QSBA* is the closest to *FRANTIC* in terms of classification accuracy but it does achieve its accuracy at a cost to rule comprehensibility—it uses subsethood values to determine fuzzy quantifiers in the range [0,1], one for each possible condition in a rule antecedent. An example rulebase is provided in Table VII. Table VIII gives a *FRANTIC* rulebase, induced with the use of linguistic hedges.

VI. CONCLUSIONS

This work demonstrates that *FRANTIC* provides rulebases that are comparable or superior to rulebases produced by several fuzzy induction algorithms, in terms of both accuracy and comprehensibility. A significant boost in accuracy is achieved by enriching the hypothesis language with the addition of linguistic hedges in the rule antecedents. This comes at the cost of an increase in the time taken to construct those rules, but it has been demonstrated that there are several ways this issue may be resolved.

Future work lies in ensuring *FRANTIC* can handle more complex datasets—a major assumption in the *FRANTIC-SRL* strategy is that one rule is sufficient to adequately describe a class, and so *m* ACOs are run in parallel where *m* is the number of classes. Though a useful starting point for investigating a strategy that generates and evaluates a complete fuzzy rulebase simultaneously, it may be a naive assumption when using larger and more complex real-world datasets.

Work will therefore be carried out to extend *FRANTIC-SRL* to run as many ACOs as are necessary to adequately describe a class. One approach is to determine beforehand how many rules may be required to describe a class, and to then initiate the appropriate number of ACOs. This may perhaps be accomplished by analysing the training data to see whether any subclusters of instances may be found within individual classes. The number of subclusters within a class would then indicate the number of ACOs to be initiated for that class.

TABLE VIII

FRANTIC-SRL RULEBASE WITH LINGUISTIC HEDGES FOR WATER TREATMENT DATABASE

R1	IF SED-S is Not-High AND RD-SED-G is More_or_less-High THEN OUTCOME is NORMAL
R2	IF PH-D is More_or_less-Normal AND DBO-D is Not-Normal THEN OUTCOME is FAULTY

REFERENCES

[1] M. Galea and Q. Shen, "Fuzzy rules from ant-inspired computation," *Proc. IEEE Int. Conf. Fuzzy Systems*, 2004, pp. 1691–1696.

[2] M. Galea and Q. Shen, "*FRANTIC*–a system for inducing accurate and comprehensible fuzzy rules," *Proc. UK Workshop Computational Intelligence*, 2004, pp. 136–143.

[3] M. Dorigo and T. Stützle, *Ant Colony Optimization*, Cambridge MA USA, London UK: The MIT Press, 2004.

[4] M. Galea and Q. Shen, "Iterative vs simultaneous fuzzy rule induction," *Proc. IEEE Int. Conf. Fuzzy Systems*, 2005, pp. 767–772.

[5] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—Parts I, II," *Information Sciences*, vol. 8 pp. 199–249, vol. 8 pp. 301–357, 1975.

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

[7] J. Casillas, O. Cordón and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," *Proc. 2nd Int. Workshop Ant Algorithms*, 2000, pp. 13–21.

[8] R. Parpinelli, H. Lopes and A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evolutionary Computation*, vol. 6, pp. 321–332, 2002.

[9] Z. Wang and B. Feng, "Classification rule mining with an improved ant colony algorithm," *Lecture Notes in Artificial Intelligence*, vol. 3339, pp. 357–367, Springer-Verlag, 2004.

[10] N. Holden N and A. Freitas, "Web page classification with an ant colony algorithm," *Lecture Notes in Computer Science*, vol. 3242, pp. 1092–1102, Springer Verlag, 2005.

[11] P. Carmona and J. L. Castro, "Using ant colony optimization for learning maximal structure fuzzy rules," *Proc. IEEE Int. Conf. Fuzzy Systems*, 2005, pp. 999–999.

[12] B. Kosko, "Fuzzy entropy and conditioning," *Information Sciences*, vol. 40, pp. 165–174, 1986.

[13] H. Ishibuchi, T. Nakashima and T. Morisawa, "Voting in fuzzy rule-based systems for pattern classification problems," *Fuzzy Sets and Systems*, vol. 103, pp. 223–238, 1999.

[14] J. R. Quinlan JR, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[15] C.L. Blake and C.J. Merz, UCI Repository of Machine Learning Data, Dept. Computer Science, Univ. California, Irvine CA, 1998. http://www.ics.uci.edu/~mlearn/MLRepositary.html

[16] R. Jensen and Q. Shen, "Fuzzy-rough attribute reduction with application to web categorization," *Fuzzy Sets and Systems*, vol. 141, pp. 469–485, 2004.

[17] Q. Shen and A. Chouchoulas, "A rough-fuzzy approach for generating classification rules," *Pattern Recognition*, vol. 35, pp. 2425–2438, 2002.

[18] Y. Yuan Y and M. J. Shaw (1995) "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, pp. 125–139, 1995

[19] Y. Yuan and H. Zhuang, "A genetic algorithm for generating fuzzy classification rules," *Fuzzy Sets and Systems*, vol. 84, pp. 1–19, 1996.

[20] K. Rasmani and Q. Shen, "Weighted linguistic modelling based on fuzzy subsethood values," *Proc. IEEE Int. Conf. Fuzzy Systems*, 2003, pp. 714–719.

[21] S.-M. Chen, S.-H. Lee and C.-H. Lee, "A new method for generating fuzzy rules from numerical data for handling classification problems," *Applied Artificial Intelligence*, vol. 15, pp. 645–664, 2001.

[22] M. Galea, "Applying swarm intelligence to rule induction," MSc thesis, Div. Informatics, Edinburgh Univ., Edinburgh UK, 2002.

[23] L. M. Gambardella and M. Dorigo, "Ant-Q: a reinforcement learning approach to the travelling salesman problem," *Proc. 12th Int. Conf. Machine Learning*, 1995, pp. 252–260.