Simultaneous ant colony optimisation algorithms for learning linguistic fuzzy rules

Galea, Michelle; Shen, Qiang

Published in:
Swarm Intelligence in Data Mining

Publication date:
2006

Citation for published version (APA):
Galea, M., & Shen, Q. (2006). Simultaneous ant colony optimisation algorithms for learning linguistic fuzzy rules. In C. Grosan, & V. Ramos (Eds.), *Swarm Intelligence in Data Mining* (pp. 75-99). Springer Nature.
http://hdl.handle.net/2160/489

Simultaneous Ant Colony Optimization Algorithms for Learning Linguistic Fuzzy Rules

Michelle Galea[1] and Qiang Shen[2]

[1] Centre for Intelligent Systems and their Application
School of Informatics
University of Edinburgh
Edinburgh EH8 9LE, UK
m.galea@sms.ed.ac.uk
[2] Department of Computer Science
University of Wales
Aberystwyth SY23 3DB, UK
qqs@aber.ac.uk

Summary. An approach based on Ant Colony Optimization for the induction of fuzzy rules is presented. Several Ant Colony Optimization algorithms are run simultaneously, with each focusing on finding descriptive rules for a specific class. The final outcome is a fuzzy rulebase that has been evolved so that individual rules complement each other during the classification process. This novel approach to fuzzy rule induction is compared against several other fuzzy rule induction algorithms, including a fuzzy genetic algorithm and a fuzzy decision tree. The initial findings indicate comparable or better classification accuracy, and superior comprehensibility. This is attributed to both the strategy of evolving fuzzy rules simultaneously, and to the individual rule discovery mechanism, the Ant Colony Optimization heuristic. The strengths and potential of the approach, and its current limitations, are discussed in detail.

1 Introduction

Many fuzzy rule induction algorithms are adaptations of crisp rule induction algorithms that fail to take into account a fundamental difference between crisp and fuzzy rules, which is how they interact during the inference or classification process. This chapter presents a strategy based on the simultaneous running of several Ant Colony Optimization (ACO) algorithms, designed specifically with the induction of a complete fuzzy rulebase in mind.

Due to their very nature, fuzzy rules will match or cover all cases within a training set, but to varying degrees. Having a final rulebase of complementary

fuzzy rules is therefore essential to the inference process – i.e. it is necessary to avoid a situation where a case requiring classification is closely matched by two or more rules that have different conclusions. To encourage complementary rules an approach is adopted that allows fuzzy rules describing different classes to be evolved and evaluated simultaneously.

The mechanism adopted for discovering the individual fuzzy rules is based on the ACO heuristic, so that several ACOs are run in parallel, with each constructing rules that describe a specific class. The constructionist nature of the adapted ACO algorithm itself affords several additional advantages, such as providing inbuilt mechanisms for preventing over-fitting to the training data, and dealing with imbalanced datasets, a common occurrence in real-world datasets.

The next section introduces fuzzy rules and rule-based systems, in so far as it is necessary to understand the work presented here. For a more comprehensive exposition the reader is directed to [1] for fuzzy set theory and logic in general, and to [2] for classification and modeling with linguistic fuzzy rules in particular. This section also describes ACO in the context of rule induction, and reviews the limited existing literature on the topic. Section 3 highlights the potential advantage provided by simultaneous rule learning, and describes the implemented system. Section 4 then presents experiments and an analysis of results. The final section highlights the advantages and limitations of the current research, which in turn suggest several avenues for future work.

2 Background

2.1 Fuzzy Rules and Rule-Based Systems

There are several different approaches for reasoning with imperfect or imprecise knowledge [3], including fuzzy rule-based systems (FRBSs) that are based on fuzzy set theory and fuzzy logic [4]. FRBSs capture and reason with imprecise or inexact knowledge (in fuzzy logic everything is a measure of degree [5]), and since many real-world problems contain a measure of imprecision and noise, the application of such approximate reasoning systems in these situations is often not only a viable but a necessary approach. This is supported by many successful applications in industry and commerce that deal with automated classification, diagnosis, monitoring and control (e.g. [6, 7]).

A simplified view of an FRBS is depicted in Fig. 1 on the facing page. At the core of such a system are:

1. A knowledge base that consists of fuzzy production IF-THEN rules (the rulebase – RB) that conceptualise domain knowledge, and the membership functions (the database – DB) defining the fuzzy sets associated with conditions and conclusions in the rules.
2. An inference procedure that uses this stored knowledge to formulate a mapping from a given input (e.g. in classification, conditions denoted by

attribute values) to an output (e.g. in classification, a conclusion denoted by a class label).

The knowledge base has traditionally been determined via discussions with domain experts but this approach has many problems and shortcomings [8] – the interviews are generally long, inefficient and frustrating for both the domain experts and knowledge engineers, especially so in domains where experts make decisions based on incomplete or imprecise information. Data mining for both the fuzzy rules and associated membership functions has therefore been an active research area in the last decade. In this work the membership functions are already determined, and the data mining is applied to the induction of linguistic fuzzy rules.

The following subsections present basic concepts such as fuzzy sets, membership functions and linguistic variables, and describe fuzzy rules and how they are used in the inference process.

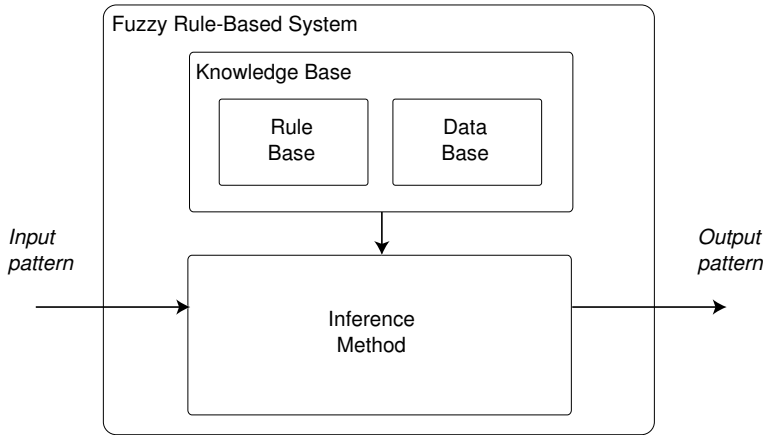

Fig. 1. Fuzzy rule-based system

Fuzzy Sets and Operators

A fuzzy set is a generalisation of a classical crisp set. A crisp set has a clearly defined boundary that either fully includes or fully excludes elements. A fuzzy set has a fuzzy boundary and each element u in the universe of discourse U belongs to the fuzzy set, but with a degree of membership in the real interval [0,1]. The closer this value is to 0, the less u may be considered as belonging to the fuzzy set in question, whilst the closer the membership value is to 1, the more u may be considered as belonging.

The degree of membership of the element u for the fuzzy set A is denoted by $\mu_A(u)$, where μ_A is called the membership function of A. This function

maps each input $u \in U$ to its appropriate membership value. The fuzzy set A may therefore be denoted by the set of pairs:

$$A = \{(u, \mu_A(u)) \mid u \in U,\ \mu_A(u) \in [0, 1]\} \tag{1}$$

The graph of a membership function may take different shapes, and whether a particular shape is appropriate is generally determined by the application context. Common functions include the triangular, trapezoidal and the Gaussian [7].

Fuzzy sets are associated with each condition in a fuzzy rule and so it is necessary to be able to perform specific operations on single or multiple fuzzy sets. Fuzzy generalisations of the standard set union, intersection and complement are, respectively, *min*, *max* and the additive complement:

$$\mu_{A \cap B}(u) = min(\mu_A(u), \mu_B(u)) \tag{2}$$

$$\mu_{A \cup B}(u) = max(\mu_A(u), \mu_B(u)) \tag{3}$$

$$\mu_{\neg A}(u) = 1 - \mu_A(u) \tag{4}$$

The above three operators are the ones most commonly used for interpreting and combining fuzzy values over the corresponding logical connectives in a fuzzy IF-THEN rule (conjunction, disjunction and negation), but there are several other definitions that may be used instead. In general, an intersection of two fuzzy sets A and B is defined by a binary operator called a *triangular norm* (or *t-norm*), that can aggregate two membership values. Similarly, a union of two fuzzy sets A and B is defined by a binary operator called a *triangular co-norm* (or *s-norm*). Other *t-norms*, *s-norms* and alternative fuzzy complement operators are discussed in [9] in more detail.

Linguistic Variables and Fuzzy Rules

A linguistic variable is a variable that has words or sentences in a natural or synthetic language as its domain values [10]. The domain values are called linguistic terms or labels, and each has associated with it a defining membership function.

Figure 2 on the next page illustrates an example of a linguistic variable called *Income*, that has three linguistic terms in its domain {*low_income, medium_income, high_income*}. It is the overlap between the membership functions defining the linguistic terms that allows fuzzy rules to represent and reason with imprecise or vague information.

When an observation of a linguistic variable is made, or a measurement is taken, the value needs to be 'fuzzified' before it can be used by the FRBS, i.e. its degrees of membership for the different linguistic terms of the variable need to be determined. For instance, consider Fig. 2 again. If the income for a person is given as \$30k, this translates to $\mu_{low\_income}(\$30k) = 0.3$, $\mu_{medium\_income}(\$30k) = 0.6$, and $\mu_{high\_income}(\$30k) = 0.0$.

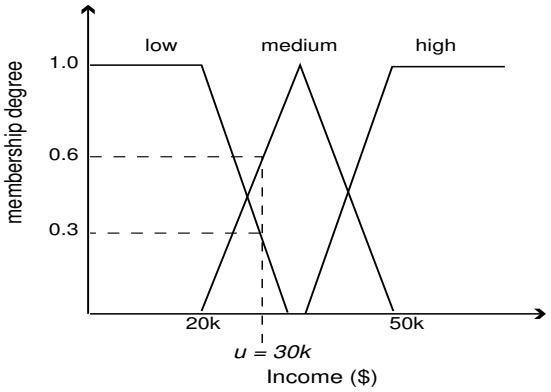

Fig. 2. A linguistic fuzzy variable

There are different types of fuzzy IF-THEN rules, but the rules induced here are linguistic Mamdani-type rules [11], e.g.:

$$R_1 : \; IF \text{ TEMPERATURE } is \; Mild \text{ OR } Cool \; AND \text{ WIND } is \; Windy \; THEN \; Weightlifting$$

The underlying dataset which this rule partly describes, with its attributes and domain values, is described in more detail in Sect. 4.1.

Linguistic fuzzy rules are a particularly explicit and human-comprehensible form for representing domain knowledge. Comprehensible knowledge, in turn, may help to validate a domain expert's knowledge, refine an incomplete or inaccurate domain theory, provide confidence in the automated system now affecting decisions, highlight previously undiscovered knowledge, and, optimize system performance by highlighting both significant and insignificant features (attributes) of the domain.

Classification using Fuzzy Rules

In an FRBS used for classification purposes, all rules are applied to the input vector, and each will match the input pattern but to varying degrees. How a decision is reached as to what output (classification) should be assigned is handled by the inference method.

There are several different inference methods, with variations on each depending on which *t-norm*, *s-norm* and other aggregation operators are used. References [12, 13] provide several different examples of inference methods. The one used here is a popular one, mainly due to the high transparency of the classification process. It was chosen because it is the one utilised by other works against which the system implemented here is compared. This makes

the comparison between the different algorithms more equitable, by ensuring that any difference in performance is due to the rule induction algorithm, and not due to the inference method. To this end, the same fuzzy sets and membership functions that were used by the other algorithms, are also used by the implemented system.

The inference process used is a single winner based-method [13] and the rule that achieves the highest degree of match with the input pattern or vector gets to classify that vector. This is depicted in Fig. 3 where $mCond(R_i, u)$ denotes the degree of match between the antecedent part of rule R_i and the input pattern u, and c^{R_i} is the class of R_i.

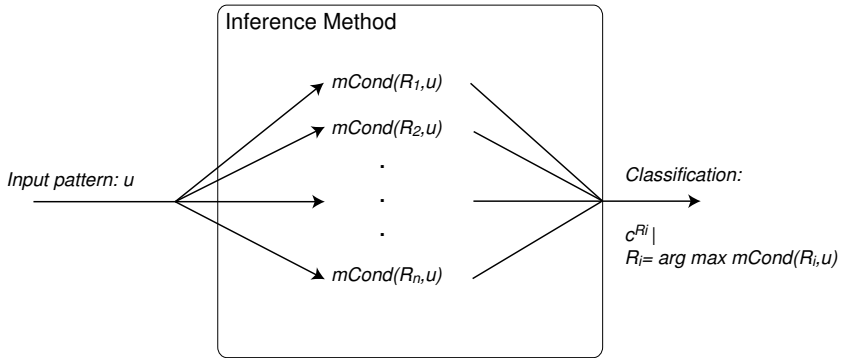

Fig. 3. Classification by an FRBS – single winner method

A Rule-Matching Example

Since the process of finding a degree of match between a fuzzy rule antecedent and an input pattern is used not only in classifying the test set for estimating the accuracy of the induced rulebase, but also in constructing the rules and in evaluating them, an example follows.

For illustration purposes a more convenient representation of the rule presented earlier is used: R_1=(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 two or three 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), i.e. each observation or measurement of each variable has already been fuzzified. The representation is similar as for rule R_1, 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, but 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 degree of condition match between rule R_1 and instance u is given by

$$mCond(R_1, u) = min_k(mAtt(R_1^k, u^k)) \tag{5}$$

In (5) above $mAtt(R_1^k, u^k)$ measures the degree of match between an attribute k in R_1 and the corresponding attribute in u:

$$mAtt(R_1^k, u^k) = \begin{cases} 1 & : \ R_1^k \ empty \\ max_j(min(\mu_j(R_1^k), \mu_j(u^k))) & : \ otherwise \end{cases} \tag{6}$$

where R_1^k *empty* indicates that no term from the domain of attribute k is present in rule R_1, 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: $mAtt(R_1^1, u^1) = 1.0$, $mAtt(R_1^2, u^2) = 0.7$, $mAtt(R_1^3, u^3) = 1.0$ and $mAtt(R_1^4, u^4) = 0.9$, with the degree of match between the rule antecedent of R_1 and the input pattern u therefore being $mCond(R_1, u) = min(1.0, 0.7, 1.0, 0.9) = 0.7$.

If the purpose were classification of the input pattern u, then the degree of condition match between u and all other rule antecedents in the rulebase is determined. For instance, if two other rules were present, say R_2 describing the conditions leading to a decision to go *Swimming*, and R_3 leading to a decision to play *Volleyball*, and their degree of condition matches were $mCond(R_2, u) = 0.2$ and $mCond(R_3, u) = 0.4$, then u would be assigned the same class as that of R_1 – *Weightlifting*. Since the actual class of u is *Weightlifting*, then during training or testing this would be counted as a correct classification. If more than one rule describing different classes obtained the highest degree of condition match with u, this would be considered a misclassification when determining the accuracy of the induced rulebase.

2.2 Ant Colony Optimization and Rule Induction

Ant algorithms are heuristics inspired by various behaviours of ants that rely on stigmergy, a form of indirect communication between individual ants that is enabled by effecting changes to a common environment [14]. Examples of such behaviours include cemetery organisation, brood sorting and foraging by real ants. Ant algorithms form a major branch of research in Swarm Intelligence [15], which may be broadly considered as the application of social insect-inspired algorithms to hard problems. They are increasingly being applied to core data mining tasks such as clustering (e.g. [16, 17]), feature selection (e.g. [18, 19]) and rule induction (e.g. [20, 21]).

Ant Colony Optimization (ACO) [22] is a particular instantiation of ant algorithms. It is a population-based algorithm motivated by the foraging strategies of real ants, which have been observed capable of finding the shortest path

between their nest and a food source [23]. 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.

In ACO each artificial ant is considered a simple agent, communicating with other ants only indirectly. A high-level description of an ACO-based algorithm is given in Fig. 4 on the facing page. Following is a brief introduction of the main elements necessary for an implementation of an ACO algorithm [15], set in the context of rule induction. More detail is provided in Sect. 3, which describes the implemented system. The first four elements relate to line (2) of Fig. 4, 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 problem is modelled as a 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. 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. It is biased towards nodes that have higher probabilities, but there is no guarantee that the node with highest probability will get selected. This allows for greater exploration of the solution space.
3. A local *heuristic* provides guidance to an ant in choosing the next node for the path (solution) it is building. This may be similar to criteria used in greedy algorithms, such as information gain for the induction of crisp rules, or fuzzy subsethood values and measurements of vagueness in a fuzzy set, for the induction of fuzzy rules.
4. A *constraint satisfaction method* forces the construction of feasible rules. For instance, if simple propositional IF-THEN rule antecedents are being constructed, then at most 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. This could be a measure based on how well the rule classifies the instances in the training set.
6. The *pheromone update rule* specifies how to modify the pheromone levels of each node in the graph between iterations of an ACO algorithm. For instance, the nodes (conditions) contained in the best rule antecedent created get their pheromone levels increased.

The application of ant algorithms to classification rule induction is a relatively new research area, but one that is gaining increasing interest. A first attempt is found in [24], where an ACO algorithm is used, however, not for

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

Fig. 4. Basic ACO Algorithm

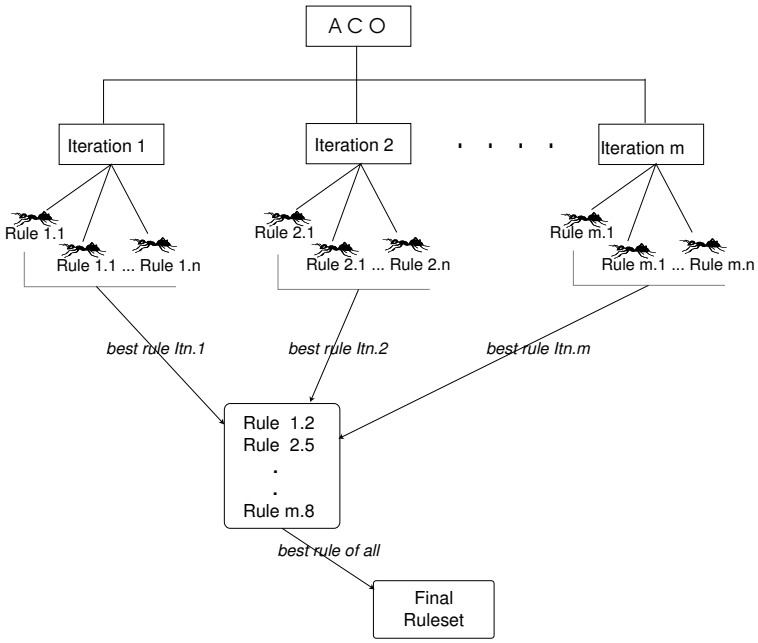

Fig. 5. Overview of a basic rule-inducing ACO algorithm

constructing fuzzy rule antecedents, but for assigning rule conclusions. In a graphical representation of the problem, the fixed number of graph nodes are fuzzy rule antecedents found previously by a deterministic method from the training set. An ant traverses the graph, visiting each and every node and probabilistically assigns a rule conclusion to each.

In [20] Parpinelli et al. introduce *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 goes round the graph selecting nodes and building its simple propositional rule antecedent. The rule conclusion is assigned afterwards by a deterministic method. Recent interest in *Ant-Miner* has resulted in various modifications to it, and applications to different problem domains. These include changes to the heuristic, transition

and pheromone update rules (e.g. [25, 26]), and application to web page classification [27] and handwritten number recognition [28].

The overall strategy *Ant-Miner* uses 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 ruleset. Instances in the training set that are covered by this best rule are removed before a new ACO algorithm is run, to find another best rule that is added to the final ruleset. This process is reiterated until only a few instances (as pre-determined by the user) 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. [29] uses the same iterative strategy as *Ant-Miner* and also produces a decision list. However, Particle Swarm Optimization [30], another major branch of Swarm Intelligence [15], is used instead of ACO as the rule discovery mechanism.

```
(1)   for each class
(2)       reinstate full training set
(3)       while classInstRemaining>maxClassInstUncovered
(4)           for numIterations
(5)               each ant constructs rule
(6)               evaluate all rules
(7)               update pheromone levels
(8)           add best rule to final rulebase
(9)           remove covered class instances
(10)   output best rulebase
```

Fig. 6. Iterative rule learning for fuzzy rules – *FRANTIC-IRL*

In [21] a different iterative strategy, outlined in Fig. 6, is used for the induction of fuzzy rules. This system is called *FRANTIC-IRL* and several ACO algorithms are run for each class, with each producing a fuzzy rule that covers a subset of the instances in the training set belonging to that class. Fuzzy rules describing one class are produced until only a few class instances remain in the training set, line (3). The system then reinstates the full training set and proceeds to run ACO algorithms to find fuzzy rules describing another class. The process continues until all classes have been adequately described.

This iterative approach to fuzzy rule learning using ACO algorithms compared very favourably against other fuzzy rule induction algorithms, in terms of both accuracy and comprehensibility. The results and analyses also suggested, however, that a simultaneous rule learning strategy would be more appropriate for the induction of fuzzy rules. This chapter presents such a simultaneous strategy for the induction of a cooperative fuzzy rulebase.

3 Simultaneous Fuzzy Rule Learning

This section first highlights the potential advantage of simultaneous rule learning over iterative rule learning, and then describes the implemented system – *FRANTIC-SRL*.

3.1 Why Simultaneous Rule Learning

As described in the previous section, *FRANTIC-IRL* follows an iterative rule learning approach where the fuzzy rules making up a rulebase are created independently of each other, i.e. without taking into account how they will interact in the final rulebase.

[21] highlights a disadvantage with this approach. *FRANTIC-IRL* was run on the Saturday Morning Problem dataset (also described in detail in Sect. 4.1 of this chapter). Rulebase A which consists of rules R1–R3 in Table 1 is one of the rulebases commonly produced, achieving an accuracy of 93.75% on the dataset, while Rulebase B which consists of rules R1–R4 is another rulebase and achieves an accuracy of 87.50%. Table 2 on the next page helps to illustrate a potential problem with fuzzy rule interaction during classification. The first column is an instance identifier of some of the instances in the dataset, the second provides the actual class of an instance, while columns 3–6 give the degree of match between a fuzzy rule from Table 1 and an instance. The abbreviation in brackets following a rule identifier denotes the class the rule describes: VB–*Volleyball*, SW–*Swimming*, and WL–*Weightlifting*. Column 7 of Table 2 gives the classification made by Rulebase A (Rb A), while the last column gives the classification made by Rulebase B (Rb B). It should be remembered that an instance is classified by the fuzzy rule with the highest degree of match.

Table 1. *FRANTIC-IRL* rulebases for Saturday Morning Problem – Rulebase A: R1-R3, 93.75% accuracy; Rulebase B: R1-R4, 87.50% accuracy

R1	IF OUTLOOK is NOT_Rain AND TEMPERATURE is NOT_Cool 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 TEMPERATURE is NOT_Hot AND WIND is Windy THEN *Weightlifting*
R4	IF OUTLOOK is NOT_Sunny AND TEMPERATURE is NOT_Mild THEN *Weightlifting*

Consider now only the instances that actually describe *Weightlifting* (instances in Table 2 with 'WL' in column 2) – Rulebase B is a closer match to the data than Rulebase A, since the additional rule R4 describing *Weightlifting*

Table 2. Fuzzy rule interaction and classification

Inst. ID	Actual Class	Degree of match R1(VB)	R2(SW)	R3(WL)	R4(WL)	Classification Rb *A*	Rb *B*
5	WL	0.1	0.1	0.3	0.7	WL	WL
6	WL	0.3	0.0	0.4	0.7	WL	WL
7	WL	0.0	0.0	0.1	1.0	WL	WL
10	WL	0.1	0.0	0.9	0.9	WL	WL
13	WL	0.0	0.2	0.8	0.8	WL	WL
14	WL	0.3	0.0	0.7	0.7	WL	WL
15	WL	0.0	0.0	0.8	1.0	WL	WL
8	*VB*	*0.2*	*0.0*	*0.0*	*0.8*	*VB*	*WL*

achieves a very high degree of match with all WL instances. However, R4 also achieves the highest degree of match of all rules with instance 8, and therefore *mis*classifies this instance. Note that Rulebase *A* (rules R1-R3), though occasionally achieving a lower degree of match with WL instances than Rulebase *B*, still manages to correctly classify all WL instances, *and* avoids misclassifying instance 8. This issue arises as a direct consequence of the strategy used to induce the complete fuzzy rulebase – in iterative rule learning the fuzzy rules are added to the final rulebase sequentially, and without taking into account how they may interact with rules describing different classes already in the rulebase, or with other rules that may be added later on.

A strategy is therefore required that encourages optimal fuzzy rule interaction during classification. *FRANTIC-SRL* is a system designed with this requirement in mind – during creation of the rulebase individual rules are not evaluated separately on the training set as in *FRANTIC-IRL*, but are grouped together and evaluated as a potential complete rulebase. *FRANTIC-SRL* is described in detail in the following section.

3.2 *FRANTIC-SRL*

FRANTIC-SRL (Fuzzy Rules from ANT-Inspired Computation – Simultaneous Rule Learning) 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 7.

After each class has had its rules created for a particular iteration (Fig.7, lines (2)–(3)), all possible combinations of rules (one from each class) are formed into a rulebase and this is tested on the training set (lines (4)–(5)). The rules in the best performing rulebase are used to update the pheromone levels (line (6)), 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.

```
(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. 7. Simultaneous rule learning for fuzzy rules – *FRANTIC-SRL*

Rule Construction

FRANTIC-SRL has 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* TEMPERATURE *is* NOT_*Rain AND* WIND *is Windy THEN Weightlifting*).

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 depends on both a heuristic value and the pheromone level associated with the node. It is made probabilistically but is biased towards terms that have relatively higher heuristic and pheromone values.

After selection but before a term is actually added to a rule antecedent, a check is made – this ensures that the resultant rule antecedent covers a minimum number of the appropriate class instances from the training set (set by a parameter called `minInstPerRule`), and is a way of avoiding overfitting to the training data. As previously mentioned, all fuzzy rules cover all training instances, but to varying degrees, and so what constitutes coverage of an instance by a fuzzy rule needs defining. During rule construction, a fuzzy rule describing a specific class is said to cover or match an 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 set by a parameter called `constructionThreshold`.

Consider as an example the fuzzy rule and instance given in Sect. 2.1, paragraph *Classification using Fuzzy Rules*. Rule R_1 and instance u belonged to the same class so condition (1.) above is satisfied. The degree of condition match between R and u was found to be $mCond(R, u) = 0.7$. If `constructionThreshold` is set to 0.7 or lower, then R_1 is considered to ade-

quately cover u, while if it is set to higher than 0.7, then R_1 is considered not to cover u.

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, e.g. OUTLOOK=(Sunny OR Mild).

Heuristic

The heuristic used to guide ants when selecting terms is based on fuzzy subsethood values [31], giving a degree to which one fuzzy set A is a subset of another fuzzy set B:

$$S(A,B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u \in U} min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)} \tag{7}$$

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 $j - \eta_j$ – 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. $\eta_{NOT\_j} = 1 - \eta_j$.

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 TEMPERATURE *is Mild AND* 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 of an iteration of a particular ACO, say R, get their pheromone levels increased:

$$\tau_j(t+1) = \tau_j(t) + (\tau_j(t) \times Q), \forall j \in R \tag{8}$$

i.e. at time $t+1$ each term j that is present in rule R gets its pheromone level increased in proportion to the quality Q of the rule (defined in Sect. 3.2).

The pheromone levels of all terms are then normalised (each pheromone level is divided by the sum of all pheromone levels), which 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.

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:

$$P_j^m(t) = \frac{[\eta_j] \times [\tau_j(t)]}{\sum_{i \in I_m} [\eta_i] \times [\tau_i(t)]} \tag{9}$$

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 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.

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.

The method of classification used during evaluation is the single winner-based method described briefly in Sect. 2.1. More specifically, for each instance u:

1. for each rule, calculate the condition match for instance u;
2. assign to instance u the class of the rule with the highest condition 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.

Currently, all possible rulebases are created and evaluated after an iteration, by combining a rule from one class (ACO), with one rule from each of the other classes. This brings the total number of rulebase evaluations to:

$$numIterations * numAnts^{numClasses} \tag{10}$$

where $numIterations$ and $numAnts$ are as defined in Table 5 on page 19, and $numClasses$ is the number of class labels in the training set. It is quite possible, however, that this number may be drastically reduced without its impacting on the quality of the final rulebase. In work using co-operative co-evolution [32] to induce a complete knowledge base (e.g. [33], where one genetic algorithm evolves rulebases, and another evolves membership functions), not all possible combinations are formed. Generally, only a few representatives from each population are used to form different combinations of knowledge bases, and these representatives may be chosen according to fitness, randomly, or a combination of both. This suggests a useful direction for an investigation into the reduction of computational expense relating to rulebase evaluations for *FRANTIC-SRL*.

4 Experiments and Analyses

4.1 Experiment Setup

This subsection details the datasets, the fuzzy rule induction algorithms against which *FRANTIC-SRL* is compared, and the parameter settings that are used in the empirical study reported below.

The Datasets

Saturday Morning Problem. The first dataset on which *FRANTIC-SRL* is tested is a fuzzified version of the small, artificial Saturday Morning Problem

dataset originally used in the induction of decision trees [34]. This dataset was chosen as it has been used by several fuzzy rule inductions algorithm, and so permits a direct comparison between these algorithms and *FRANTIC-SRL*.

The dataset consists of sixteen instances, and four linguistic condition attributes with each having two or three linguistic terms. The class attribute PLAN classifies each instance into *Volleyball*, *Swimming* or *Weightlifting*. The condition attributes are described in Table 3.

Table 3. Saturday Morning Problem dataset features

Attribute	Linguistic Terms
OUTLOOK	Sunny, Cloudy, Rain
TEMPERATURE	Hot, Cool, Mild
HUMIDITY	Humid, Normal
WIND	Wind, Not-windy

Water Treatment Plant Database. This real-world database [35] is more challenging and 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 has 13 possible classifications for each daily set of observations, however, most classifications are 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 normal operation of the plant, versus faulty operation.

The 13 classifications have therefore been collapsed to two: *OK* and *Faulty*, as in [36]. 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 include fuzzification of the features using trapezoidal functions, and a feature subset selection process [37] to reduce the number of features ([36] indicated better accuracy results when a reduced water treatment dataset was used). A description of the retained features is shown in Table 4 on the following page. Each feature is described by three linguistic terms (*low, high, normal*), except for the last which uses only two (*low, high*).

Other Induction Algorithms

The fuzzy rulebases produced by *FRANTIC-SRL* are compared with those produced by a fuzzy decision tree algorithm (*FDT*) [38], a fuzzy genetic algorithm (*FGA*) [39], and two methods based on subsethood values (*FSBA* [40], *WSBA* [41]). Apart from *WSBA*, the algorithm acronyms are not the names

Table 4. Water Treatment Plant database 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

given to the algorithms by the original authors, but are introduced here for ease of reference.

FDT is a deterministic algorithm. *FGA* uses randomness to generate rules and so may generate different rulebases achieving different predictive accuracy. The number of fuzzy rules in the rulebases produced by both algorithms is not pre-determined by the user.

FSBA uses subsethood values to select a small number of conditions to formulate one rule for each class in the training set. It is a deterministic algorithm but requires the setting of two 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 the class. Instead, an indirect rule is formed and will fire if the membership of the instance to be classified is greater than β (e.g. $IF\ Membership(OK) < \beta\ THEN\ OUTCOME\ is\ FAULTY$).

WSBA, the second subsethood-based algorithm, uses subsethood values not to determine which linguistic terms should be present in a rule, but to determine a fuzzy quantifier in the range [0,1] for each term, all of which are present. Like *FSBA*, and *FRANTIC-SRL* as it is currently implemented, it also generates only one rule per class. Examples of rules generated by all the algorithms are provided in the following subsections.

FRANTIC-SRL Parameters.

FRANTIC-SRL parameters that require setting are listed in Table 5 on the next page, 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). 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. It is therefore quite possible that different settings of these parameters may lead to even better results.

Table 5. *FRANTIC-SRL* Parameters

Parameter Name	Description	SM	WT
`numAnts`	Number of ants constructing a solution within an iteration	4	10
`numIterations`	Number of iterations per ACO run	25	30
`minInstPerRule`	Required during rule construction – minimum number of instances in training set that a rule must cover	4	70%
`constructionThreshold`	Required during rule construction – sets the value for the threshold below which a rule is considered not to cover an instance in the training set	0.65	0.65

The `minInstPerRule` parameter is flexible enough so that different values may be given to different classes. This is particularly useful in imbalanced datasets such as the Water Treatment 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 relevant 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 the 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, and reduces the need for unnecessary user intervention.

4.2 Saturday Morning Problem Results

A summary of the results produced on this dataset by various algorithms is provided in Table 6 on the following page – 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.

Table 6. Saturday Morning Problem – comparison of algorithms

	%Accuracy	#Rules	#Terms
FDT	81.25	6	1.7
FGA	87.50	5	3.2
FSBA	93.75	3	2.3
FRANTIC	93.33	3	2.7

The accuracy of only one rulebase is reported for *FGA* in [39], and reproduced in Table 6, and so the assumption here is that it is the best rulebase obtained. The results for *FSBA* reported in [40] are for $\alpha = 0.9$ and $\beta = 0.6$, and the assumption again is that this is the best obtainable. Since *FRANTIC-SRL* is a stochastic-based algorithm the result reported in Table 6 is the average of the accuracies obtained from 30 runs of the system with the parameters set as in Table 5. The standard deviation of the 30 accuracies is 1.6, i.e. only 2 out of the 30 runs produced accuracies below 93.75%, making the overall accuracy comparable with that obtained by *FSBA*.

Tables 7 to 10 on pages 20–21 provide examples of the rules generated by these algorithms. *FDT* generates simple propositional rules, *FGA* generates propositional rules with internal disjunction, and *FSBA* generates simple propositional rules that however may include negated terms. *FRANTIC-SRL* has the ability to generate all these variations on propositional rules, but a few early runs of the system determined that for this dataset rules that included negated terms were more accurately descriptive. *FRANTIC-SRL* and *FSBA* are also comparable with respect to the number of rules in a rulebase, and the average number of conditions per rule. Note, however, that the final rule produced by *FSBA* has no explanatory power of its own, as it is written in terms of the other rules.

Table 7. *FDT* rulebase for Saturday Morning Problem (81.25% accuracy)

R1	IF TEMPERATURE is Hot AND OUTLOOK is Sunny THEN *Swimming*
R2	IF TEMPERATURE is Hot AND OUTLOOK is Cloudy THEN *Swimming*
R3	IF OUTLOOK is Rain THEN *Weightlifting*
R4	IF TEMPERATURE is Mild AND WIND is Windy THEN *Weightlifting*
R5	IF TEMPERATURE is Cool THEN *Weightlifting*
R6	IF TEMPERATURE is Mild AND WIND is Not-windy THEN *Volleyball*

4.3 Water Treatment Plant Results

The middle column of Table 11 on the next page indicates the average accuracy obtained by several algorithms after performing stratified ten-fold cross-validation on the Water Treatment dataset. The same folds of the dataset were

Table 8. *FGA* rulebase for the Saturday Morning Problem (87.25% accuracy)

R1	IF OUTLOOK is Sunny OR Cloudy AND TEMPERATURE is Hot THEN *Swimming*
R2	IF OUTLOOK is Rain THEN *Weightlifting*
R3	IF TEMPERATURE is Mild OR Cool AND WIND is Windy THEN *Weightlifting*
R4	IF OUTLOOK is Cloudy OR Rain AND HUMIDITY is Humid THEN *Weightlifting*
R5	IF OUTLOOK is Sunny OR Cloudy AND TEMPERATURE is Mild OR Cool AND HUMIDITY is Normal AND WIND is Not-windy THEN *Volleyball*

Table 9. *FSBA* rulebase for Saturday Morning Problem (93.75% accuracy)

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*

Table 10. *FRANTIC-SRL* rulebase for Saturday Morning Problem (93.75% accuracy)

R1	IF OUTLOOK is NOT_Rain AND TEMPERATURE is NOT_Cool 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 TEMPERATURE is NOT_Hot AND WIND is Windy THEN *Weightlifting*

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 did. 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. Note that since *FRANTIC-SRL* is a stochastic algorithm, the *FRANTIC* results presented in Table 11 are averages of *ten* ten-fold cross-validations.

Table 11. Water Treatment Plant – comparison of algorithms

	%Accuracy	#Terms
WSBA	81.74 (7.6)	32.00 (0.0)
FSBA	69.51 (7.0)	4.45 (0.4)
FRANTIC	76.08 (6.6)	2.00 (0.0)

As for the Saturday Morning Problem dataset, a few initial runs indicated that *FRANTIC-SRL* rules with negated terms were better descriptors of the Water Treatment Plant database. *FSBA* 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 accuracy results reported here are the best obtained with $\alpha = 0.8$ and $\beta = 0.5$. *WSBA* obtained the best results in terms of predictive accuracy, for this particular partitioning of the data. However, it does come at a cost to rule comprehensibility.

There is a considerable difference in the length of the rules produced by each algorithm. *FRANTIC-SRL* produces the most comprehensible rulebases and an example is provided in Table 12. An *FSBA* rulebase is given in Table 13 – the rules are fairly comprehensible, though not as short as *FRANTIC* rules. It should also be remembered that this algorithm may produce rules that are described in terms of other rules, detracting from the intrinsic comprehensibility of individual rules. Table 14 on the facing page presents the best rulebase in terms of accuracy produced by *WSBA* – the fuzzy quantifiers attached to each condition allow the rules to be highly accurate, but also results in very long rules.

Table 12. *FRANTIC-SRL* rulebase for Water Treatment Plant (84.21% 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 SSV-D is NOT High THEN OUTCOME is *Faulty*

Table 13. *FSBA* rulebase for Water Treatment Plant (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*

5 Conclusions and Future Work

This initial work has demonstrated that *FRANTIC-SRL* is a viable approach to the induction of linguistic fuzzy rules – it appears to achieve a balance between rulebase accuracy and comprehensibility and compares favourably with several other fuzzy rule induction algorithms.

A hypothesis driving this work is that fuzzy rules that are evolved and evaluated simultaneously will interact better during the inference process,

Table 14. *WSBA* rulebase for Water Treatment Plant (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*

than fuzzy rules that have been evolved mainly independently of each other. Preliminary findings in [42], comparing *FRANTIC-SRL* with *FRANTIC-IRL* provides some evidence to support this. The results indicate that rule comprehensibility is maintained, that accuracy is maintained or improved, and that faster convergence to solutions, and robustness to value changes in some of the *FRANTIC* parameters may be achieved using the simultaneous approach.

However, *FRANTIC-SRL* as it is currently implemented is limited by the underlying assumption that one rule is sufficient to adequately describe a class, so that n ACO algorithms are run in parallel where n is the number of classes. Though a useful starting point to investigate this simultaneous strategy, this may be a naive assumption when applying *FRANTIC-SRL* to larger and more complex real-world problems. Work will therefore be carried out to extend the system to run as many ACO algorithms as are necessary to adequately describe a class.

One approach to achieving this is to determine beforehand how many rules may be required to describe a class, and then to initiate the appropriate number of ACO algorithms. This may 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 ACO algorithms to be initiated for that class. A more thorough investigation of the potential advantages of the simultaneous approach over the iterative one for the induction of fuzzy rules, may then be accomplished.

Using ACO for individual rule discovery has provided several advantages. The constructionist nature of the algorithm has allowed mechanisms to be

built into the rule construction process that allow the user flexibility in determining how general or specific rules should be, as determined by the parameters `minInstPerRule` and `constructionThreshold`. The interdependency and impact of these two parameters require further investigation, but it is clear that together they can work toward prevention of over-fitting to the training data, and hence towards rules with greater generalisation power.

The constraint satisfaction mechanism that ensures that valid fuzzy linguistic rules are built, also permits a useful flexibility – this is seen in *FRANTIC*'s ability to construct simple propositional rules that may also include internal disjunction between attribute values, or negated attribute values. This may be extended to further improve the expressiveness of the knowledge representation used by adding linguistic hedges such as 'very' and 'more or less' [10]. Linguistic hedges are functions that are applied to the linguistic terms of a fuzzy rule in order to increase or decrease the precision of such terms. An example rule might therefore be:

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

The use of such modifiers enriches the knowledge representation, giving it the flexibility to more accurately describe the underlying dataset, yet maintain the comprehensibility of the induced knowledge.

Another advantage offered by *FRANTIC-SRL* is the obvious and numerous opportunities for a multi-processor implementation. This will provide the necessary speed up in computation when inducing knowledge from very large databases. At a coarse level of granularity, several ACO algorithms may be run truly in parallel. At a finer level of granularity, the numerous ants of each ACO may create their rules simultaneously, for example, and the determination of the rulebase quality at the end of each iteration may also be conducted in parallel.

The potential for the application of ACO to fuzzy rule induction is high, and as yet relatively unexplored.

References

[1] Pedrycz W, Gomide F (1998) An introduction to fuzzy sets: analysis and design. A Bradford Book, The MIT Press, Cambridge MA, London
[2] Ishibuchi H, Nakashima T, Nii M (2005) Classification and modeling with linguistic information granules: advanced approaches to linguistic data mining. Springer-Verlag, Berlin Heidelberg
[3] Parsons S (2001) Qualitative methods for reasoning under uncertainty. The MIT Press, Cambridge MA, London
[4] Zadeh L (1965) Fuzzy sets. Information and Control 8:338–353
[5] Zadeh L (1988) Fuzzy logic. IEEE Computer 21:83–92

[6] Hirota K, Sugeno M (eds) (1995) Industrial applications of fuzzy technology. Advances in fuzzy systems – applications and theory 2. World Scientific
[7] Pedrycz W (ed) (1996) Fuzzy modelling: paradigms and practice. Kluwer Academic Publishers, Norwell, MA
[8] Buchanan BG, Wilkins DC (eds) (1993) Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems. Morgan Kaufmann Publishers, San Francisco, CA
[9] Klir GJ, Yuan B (1998) Operation of fuzzy sets. In: Ruspini EH, Bonisonne PP, Pedrycz W (eds) Handbook of Fuzzy Computation. Institute of Physics Publishing
[10] Zadeh L (1975) The concept of a linguistic variable and its application to approximate reasoning – Parts I, II, III. Information Sciences 8:199–249, 8:301–357, 9:43–80
[11] Mamdani EH (1976) Advances in the linguistic synthesis of fuzzy controllers. Journal of Man-Machine Studies 8:669–678
[12] Cordón O, del Jesus MJ, Herrera F (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. International Journal of Approximate Reasoning 20:21–45
[13] Ishibuchi H, Nakashima T, Morisawa T (1999) Voting in fuzzy rule-based systems for pattern classification problems. Fuzzy Sets and Systems 103:223–238
[14] Dorigo M, Bonabeau E, Theraulaz G (2000) Ant algorithms and stigmergy. Future Generation Computer Systems 16:851–871
[15] Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, New York Oxford
[16] Abraham A, Ramos V (2003) Web usage mining using artificial ant colony clustering and genetic programming. In: Proceedings of the IEEE Congress on Evolutionary Computation 2:1384–1391
[17] Hall L, Kanade P (2005) Swarm based fuzzy clustering with partition validity. In: Proceedings of the IEEE International Conference on Fuzzy Systems 991–995
[18] Jensen R, Shen Q (2005) Fuzzy-rough data reduction with ant colony optimization. Fuzzy Sets and Systems 149:5–20
[19] Al-Ani A (2005) Feature subset selection using ant colony optimization. International Journal of Computational Intelligence 2:53–58
[20] Parpinelli R, Lopes H, Freitas A (2002) Data mining with an ant colony optimization algorithm. IEEE Transactions in Evolutionary Computation 6:321–332
[21] Galea M, Shen Q (2004) Fuzzy rules from ant-inspired computation. In: Proceedings of the IEEE International Conference on Fuzzy Systems 3:1691–1696
[22] Dorigo M, Stützle T (2004) Ant colony optimization. A Bradford Book, The MIT Press, Cambridge MA, London

[23] Goss S, Aron S, Deneubourg J-L, Pasteels JM (1989) Self-organised shortcuts in the Argentine ant. Naturwissenschaften 76:579–581
[24] Casillas J, Cordón O, Herrera F (2000) Learning fuzzy rules using ant colony optimization algorithms. In: Proceedings of the 2nd International Workshop on Ant Algorithms 13–21
[25] Liu B, Abbas HA, McKay B (2003) Classification rule discovery with ant colony optimization. In: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology 83–88
[26] Wang Z, Feng B (2004) Classification rule mining with an improved ant colony algorithm. In: Lecture Notes in Artificial Intelligence 3339, Springer-Verlag, 357–367
[27] Holden N, Freitas A (2005) Web page classification with an ant colony algorithm. In: Lecture Notes in Computer Science 3242, Springer Verlag, 1092–1102
[28] Phokharatkul P, Phaiboon S (2005) Handwritten numerals recognition using an ant-miner algorithm. In: Proceedings of the International Conference on Control, Automation and Systems, Korea
[29] Sousa T, Silva A, Neves A (2004) Particle swarm based data mining aglorithms for classification rules. Parallel Computing 30:767–783
[30] Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks 4:1942–1948
[31] Kosko B (1986) Fuzzy entropy and conditioning. Information Sciences 40:165–174
[32] Potter MA, Jong KAD (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evolutionary Computation 8:1–29
[33] Pena-Reyes CA, Sipper M (2001) FuzzyCoCo: a cooperative coevolutionary approach to fuzzy modeling. IEEE Transactions on Fuzzy Systems 9:727–737
[34] Quinlan JR (1986) Induction of decision trees. Machine Learning 1:81–106
[35] Blake CL, Merz CJ (1998) UCI Repository of Machine Learning Data, Deparatment of Computer Science, University of California, Irvine CA. `http://www.ics.uci.edu/~mlearn/MLRepositary.html`
[36] Shen Q, Chouchoulas A (2002) A rough-fuzzy approach for generating classification rules. Pattern Recognition 35:2425–2438
[37] Jensen R, Shen Q (2004) Fuzzy-rough attribute reduction with application to web categorization. Fuzzy Sets and Systems 141:469–485
[38] Yuan Y, Shaw MJ (1995) Induction of fuzzy decision trees. Fuzzy Sets and Systems 69:125–139
[39] Yuan Y, Zhuang H (1996) A genetic algorithm for generating fuzzy classification rules. Fuzzy Sets and Systems 84:1–19

[40] Chen S-M, Lee S-H, Lee C-H (2001) A new method for generating fuzzy rules from numerical data for handling classification problems. Applied Artificial Intelligence 15:645–664
[41] Rasmani K, Shen Q (2004) Modifying weighted fuzzy subsethood-based rule models with fuzzy quantifiers. In: Proceedings of the IEEE International Conference on Fuzzy Systems 3:1679–1684
[42] Galea M, Shen Q (2005) Iteritive vs simultaneous fuzzy rule induction. In: Proceedings of the IEEE International Conference on Fuzzy Systems 767–772

Index