Iterative vs Simultaneous Fuzzy Rule Induction

Galea, Michelle; Shen, Qiang

DOI:
10.1109/FUZZY.2005.1452491

Publication date:
2005

Citation for published version (APA):
Galea, M., & Shen, Q. (2005). *Iterative vs Simultaneous Fuzzy Rule Induction*. 767-772. https://doi.org/10.1109/FUZZY.2005.1452491

Iterative vs Simultaneous Fuzzy Rule Induction

Michelle Galea
School of Informatics
University of Edinburgh, U.K.
Email: m.galea@sms.ed.ac.uk

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

Abstract— **Iterative rule learning is a common strategy for fuzzy rule induction using stochastic population-based algorithms (SPBAs) such as Ant Colony Optimisation (ACO) and genetic algorithms. Several SPBAs are run in succession with the result of each being a rule added to an emerging final ruleset. Each successive rule is generally produced without taking into account the rules already in the final ruleset, and how well they may interact during fuzzy inference. This popular approach is compared with the simultaneous rule learning strategy introduced here, whereby the fuzzy rules that form the final ruleset are evolved and evaluated together. This latter strategy is found to maintain or improve classification accuracy of the evolved ruleset, and simplify the ACO algorithm used here as the rule discovery mechanism by removing the need for one parameter, and adding robustness to value changes in another. This initial work also suggests that the rulesets may be obtained at less computational expense than when using an iterative rule learning strategy.**

I. Introduction

The use of stochastic population-based algorithms (SPBAs) for fuzzy rule induction has proved both popular and successful, with one of the most common strategies being that of Iterative Rule Learning (IRL) (see [1] for a review on the use of evolutionary algorithms such as genetic algorithms (GAs) [2] and genetic programming (GP) [3] for this purpose). In IRL, an SPBA is run several times in succession, with the result of each—the best fuzzy rule generated by the current algorithm—being considered a partial solution. Between runs of SPBAs, the training set is generally reduced by removing from it the cases that are covered by the newly evolved best rule. This is done so as to encourage the next algorithm to find good rules that describe the remaining cases in the training set.

However, work on fuzzy rule induction using this strategy suggests a potential shortcoming [4]. This arises out of the fact that fuzzy rules, due to their very nature, will match or cover all cases within a training set, but to varying degrees. Having a final set 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.

An alternative approach is introduced in this work called Simultaneous Rule Learning (SRL). Several Ant Colony Optimisation (ACO) algorithms are run simultaneously on the training set, and rules from the different populations (ACOs) are combined and evaluated together on the complete dataset. A similar strategy has been reported using multi-population GAs or GP but the separate populations are not all evolving rules—there are generally two populations with one evolving rulesets and the other associated membership functions (e.g. [5], [6]). The focus of this work is on evolving the fuzzy rule base, with each population finding rules to describe a specific class. The evolved rulesets are tested on benchmark classification problems and compared with rulesets evolved using IRL.

Since the application of ACO to fuzzy rule induction is still a relatively unexplored area (compared to GAs and GP, for instance), the next section introduces this topic. Section III then describes the implemented system and its two main variants, the first inducing rules using IRL and the second using SRL. Section IV presents an analysis of the results obtained using the two approaches, while Section V highlights the strengths and limitations of the current research, and the need for future work.

II. Ant Colony Optimisation and Rule Induction

Though often blind and travelling over considerable distances, real ants have the ability 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.

ACO is a population-based heuristic motivated by these foraging strategies of real ants, and its appeal lies in several factors: it provides 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-sepcific elements—such as the fitness function and heuristic—which may be readily borrowed from existing literature on rule induction; and, an explicit heuristic embedded in the solution construction mechanism that makes for easy insertion of domain knowledge.

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

```
(1)  while termination condition false
(2)     each ant constructs a new solution
(3)     evaluate new solutions
```

```
(4)     update pheromone levels
(5)   output best solution
```

Following is a brief introduction of the main elements necessary for an implementation of an ACO [7], set in the context of rule induction. The first four elements relate to line (2) above, the fifth relates to line (3), and the sixth to line (4):

1) An appropriate *problem representation* is required that allows an artificial ant to incrementally build a solution using a *probabilistic transition rule*; the 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) A local *heuristic* provides guidance to an ant in choosing the next node for the path (solution) it is building. Possible examples may be based on fuzzy subsethood values, or a measure of the vagueness in a fuzzy set.
3) The *probabilistic transition rule* determines which node an ant should visit next. The transition rule is dependent on the *heuristic* value and the *pheromone* level associated with a node.
4) A *constraint satisfaction method* forces the construction of feasible rules. For instance, if simple propositional IF-THEN rule antecedents are being constructed, then only one fuzzy linguistic term from each fuzzy variable may be selected.
5) A *fitness function* determines the quality of the solution built by an ant.
6) The *pheromone update rule* specifies how to modify the pheromone levels of each node in the graph. For instance, between iterations of an ACO algorithm, the nodes (conditions) contained in the best rule antecedent created get their pheromone levels increased.

A first attempt to apply ACO to fuzzy modelling is found in [8], though in this work the ACO algoirthm is not used for constructing fuzzy rule antecedents, but for assigning rule conclusions. In a graphical representation of the problem, the fixed number of nodes are fuzzy rule antecedents found by a deterministic method from the training set and an ant traverses the graph, visiting each and every node and probabilistically assiging a rule conclusion to each.

In [9] an IRL approach is adopted with each iteration running an ACO algorithm to produce fuzzy rules. The problem graph for the ACO consists of nodes that represent conditions that may be selected by an ant when building its fuzzy rule. It is against this work that the SRL strategy is compared, and more detail on both approaches is provided in the next section.

III. The *FRANTIC* System

FRANTIC (*F*uzzy *R*ules from *ANT*-*I*nspired *C*omputation) as introduced in [9] implements a class-dependent IRL strategy. For each class in the dataset one or more ACOs are run and from each the best rule constructed is determined and added to the final ruleset. However, before the next ACO is run to find another rule describing the same class, the cases belonging to that class that are covered by the previous best rule are removed from the training set. Before ACOs are run to find rules describing the next class, the full training set is reinstated.

A simplified version of IRL is to run just one ACO algorithm for each class, the assumption being that one rule is sufficient to describe a class. In this initial work to compare the two strategies, the simplifed form was used:

```
(1)   for each class
(2)      for noIterations
(3)         each ant constructs rule
(4)         evaluate all rules
(5)         update pheromone levels
(6)      add best rule to finalRuleSet
(7)   output finalRuleSet
```

FRANTIC has now been developed to induce rules via SRL. In the simplified form one ACO algorithm is also run for each class. However, instead of running the ACOs in succession, they are run in parallel (in principle, i.e. this is not as yet a true parallel implementation running on multiple processors), with each maintaining its own problem graph, pheromone levels and heuristic values. A brief description follows:

```
(1)   for noIterations
(2)      for each class
(3)         each ant constructs rule
(4)      for each combined ruleset
(5)         evaluate ruleset
(6)      update pheromone levels
(7)   output bestRuleSet of final iteration
```

After each class has had its rules created for a particular iteration (lines (2)–(3), all possible combinations of rules (one from each class) are formed into a ruleset and this is tested on the training set (lines (4)–(5)). The rules in the best performing ruleset 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.

In the next subsection the similarites between the two strategies as implemented in *FRANTIC* are described, while the following subsection describes the main differences.

A. Rule Construction

FRANTIC has the flexibility to create simple propositional rules (e.g. *IF* TEMPERATURE *is Cool AND* WIND *is Windy THEN Swimming*), propositional rules with internal disjunction (e.g. *IF* TEMPERATURE *is Cool* OR *Mild AND* WIND *is Windy THEN Swimming*), or propositional rules that include negated terms (e.g. *IF* TEMPERATURE *is* NOT_*Rain AND* WIND *is Windy THEN Swimming*). 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.

However, after selection and before a term is added to a rule antecedent, a check is made—this ensures that the resultant rule antecedent covers a minimum number of the appropriate classinstances from the training set (set by a parameter called `minInstPerRule`), and is a way of avoiding over-fitting to the training data.

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.

1) Fuzzy Rule Matching: 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 or match 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 with its attributes and respective domains is described in Section IV-A):

$$IF \text{ TEMPERATURE } is\ Cool \text{ OR } Mild\ AND \text{ WIND } is\ Windy\ THEN\ Swimming$$

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 as for 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

$$mCond(R,u) = Min_k(mAtt(R_k,u_k)) \quad (1)$$

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*:

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

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: $mAtt(R_1,u_1) = 1.0$, $mAtt(R_2,u_2) = 0.7$, $mAtt(R_3,u_3) = 1.0$ and $mAtt(R_4,u_4) = 0.9$, with the condition match therefore $mCond(R,u) = 0.7$. 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*.

2) Heuristic: The heuristic used to guide ants when selecting terms is based on fuzzy subsethood values [10], giving a degree to which one fuzzy set *A* is a subset of another fuzzy set *B*:

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

where in this case *u* is an instance from the training set *U*, *A* represents a class label and *B* a term that may be added to a rule antecedent.

The heuristic value of a term *j* (η_j) therefore gives a measurement of how important that term is in describing a specific class. The heuristic value for a negated term is the complement of the heuristic value for the non-negated term, i.e. $\eta_{NOT\_j} = 1 - \eta_j$. For a dataset with *n* classes, there are therefore *n* different sets of heuristic values, and an ACO with ants finding rules to describe a particular class will use the appropriate set of heuristic values.

3) Pheromone Updating: 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. Towards the end of each iteration rules created by all ants are evaluated (the difference in evaluation between *FRANTIC*-IRL and *FRANTIC*-SRL is described in Section III-B). In either case, the terms in the best rule of an iteration of a particular ACO, say *R*, get their pheromone levels increased:

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

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

4) 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:

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

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

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

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, in *FRANTIC*-IRL each rule is evaluated individually, without taking into account how it may interact with other rules describing other classes, while in *FRANTIC*-SRL a rule forms part of a ruleset that is evaluated as a whole.

1) IRL Rule Evaluation: The fitness function combines a measure of the sensitivity of a rule (its accuracy among instances of the same class as the rule) with a measure of the specificity of the rule (its accuracy among instances of different classes):

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

where

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

What constitutes coverage of a fuzzy instance by a rule is dependent on the class of the instance and rule, and on a user-defined threshold. This has been discussed in Section III-A.1. For maximum flexibility the threshold used during rule evaluation has been implemented separately from the one used during rule construction. However, further investigation is required to understand the dynamics between these (and other) parameters and it may well be possible to merge them.

The total number of rule evaluations conducted during IRL is the number of classes (ACOs run), multiplied by the number of iterations and ants: $\#classes * \#itns * \#ants$.

2) SRL Rule Evaluation: When each class has produced its set of rules, for each iteration, 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 (also used to evaluate the final ruleset produced by both IRL and SRL on a test set) is:

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 on the training set is used as a measure of the quality, *Q*, of each rule within a ruleset.

Each rule from each class is combined with each other possible rule from the other classes, so that the total number of *ruleset* evaluations conducted during SRL is $\#itns * \#ants^{\#classes}$. For the number of *rule* evaluations, multiply again by the number of rules in the ruleset (*#classes*).

TABLE I
WATER TREATMENT 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

IV. *FRANTIC* EXPERIMENTS AND RESULTS

A. The Datasets

The first set is the fuzzified Saturday Morning Problem dataset [11]. *FRANTIC*-IRL has already been tested using this dataset, and compared with several other fuzzy rule induction algorithms [9]. The results obtained indicate that *FRANTIC* has comparable or better classification accuracy, and superior rule comprehensibility (when considering the number of conditions in a rule).

The Saturday Morning 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 the real-world Water Treatment Plant Database [12]. It contains daily observations of 38 sensors

TABLE II
FRANTIC PARAMETERS

	Saturday Morning		Water Treatment	
	IRL	SRL	IRL	SRL
#itns	25	25	150	30
#ants	100	4	10	10
#cases	4	4	70%	70%
fi tness	0.5	n/a	0.5	n/a

TABLE III
AN EXAMPLE *FRANTIC* RULESET FOR THE WATER TREATMENT DATABASE (89.47% ACCURACY)

R1	IF SSV-D is NOT_Low THEN OUTCOME is OK
R2	IF PH-E is NOT_High AND SSV-P is Low AND RD-SED-G is High THEN OUTCOME is FAULTY

monitoring the operation of a waste water treatment plant, with the objective of predicting 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, but most classifications are assigned to only a few records in the database. The 13 classifications have therefore been collapsed to two: *OK* and *Faulty*, as in [13]. Records with missing values have been removed, leaving 377 records.

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 [14] to reduce the number of features (better accuracy was indicated in [13] with the reduced dataset). A description of the retained features is shown in Table I

B. FRANTIC Parameters

The different parameter values used for the two datasets and the two approaches are listed in Table II. It stipulates the values for the number of iterations for an ACO, the number of ants per iteration, the minimum number of cases a rule must cover during rule construction, and for IRL rule evaluation the fitness threshold. 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 Water Treatment one) where stipulating the same number of cases 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 cases from the training set, whilst the value '70%' means that a rule should cover at least 70% of the class cases.

For IRL runs, the parameter values for the Saturday Morning dataset are as in [9], while a few exploratory runs of *FRANTIC* were made to select parameter values giving reasonable results for the Water Treatment database. For SRL, the number of minimum cases per rule was kept the same as for the IRL runs. However, the number of iterations or the number of ants per iteration were reduced so as to make the number of evaluations between the two approaches more comparable (otherwise, with the same number of iterations and ants, the number of evaluations for SRL would far exceed the number of iterations for IRL). The impact of this is discussed further in Section V.

C. FRANTIC Results

Table IV presents a summary of the results obtained using different construction threshold values for the two approaches.

For the Saturday Morning dataset, each result is the average of the accuracies obtained from 30 *FRANTIC*-IRL or SRL runs on the training set. The figure in brackets is the standard deviation based on the predictive accuracies. IRL produced the highest average accuracy of 93.75% but for the lower and upper construction threshold values SRL significantly outperformed IRL. It should be noted that this considerable improvement in accuracy is obtained with only 4 ants per iteration for SRL runs, versus 100 ants for IRL runs, where it might be argued that the greater the number of ants, the greater the opportunity for exploring the search space and obtaining a more optimal solution.

A common observation was made in such cases—for SRL, the rules in the best ruleset of an iteration need not be the best individual rules describing a class (with 'best' as defined by the IRL fitness function). For instance, in several runs with construction threshold set at 0.45, the first iteration of rules describing the conditions necessary for a *Volleyball* decision produces rules with individual fitness of 0.80 or 0.72. For IRL, the rule with the higher fitness is chosen for pheromone updates.

These same rules are also produced during the SRL runs, but in this case the ruleset that contains the highest fitness level, and therefore is used for pheromone updates, actually contains the *Volleyball* rule of individual fitness 0.72, and not 0.80. This suggests that ruleset evaluation provides more useful information for pheromone updating than individual rule evaluation, as it takes into consideration how well the rules interact *together* on the dataset.

Apart from the increase in accuracy, SRL has introduced the added advantage of making the construction threshold more robust to value changes. Furthermore, the way the evaluation of the ruleset in SRL is done completely eliminates the need for another parameter—the fitness threshold.

For the Water Treatment dataset, each result is the average of ten 10-fold cross-validations. The figure in brackets is the standard deviation of the 10 predictive accuracies of a 10-fold cross-validation, averaged over all ten cross-validations. The SRL runs do not achieve the highest accuracy as often as IRL runs, but, overall the figures again suggest an increased robustness to variations in the construction threshold values (range of average predictive values for SRL is $76.08 - 73.29 = 2.79$, while that for IRL is $76.91 - 70.67 = 6.24$). This is further strengthened by observing that the average standard deviations over the ten 10-fold cross-validations are generally lower for SRL runs.

TABLE IV

PREDICTIVE ACCURACY -ITERATIVE VS SIMULTANEOUS RULE LEARNING

Construction Threshold	Saturday Morning		Water Treatment	
	IRL	SRL	IRL	SRL
0.45	75.21 (6.0)	92.08 (3.7)	74.38 (10.7)	74.11 (8.7)
0.50	74.38 (5.8)	93.13 (1.9)	70.67 (7.2)	75.08 (7.9)
0.55	93.75 (0.0)	92.71 (3.7)	72.15 (9.4)	73.87 (8.0)
0.60	93.75 (0.0)	93.13 (3.4)	71.45 (11.1)	73.29 (8.0)
0.65	93.75 (0.0)	93.33 (1.6)	76.91 (7.7)	76.08 (6.6)
0.70	93.75 (0.0)	91.67 (5.8)	76.42 (7.6)	74.07 (7.3)
0.75	68.75 (0.0)	68.75 (0.0)	75.58 (8.0)	73.66 (7.6)
0.80	68.75 (0.0)	68.75 (0.0)	75.58 (8.0)	74.37 (7.9)
0.85	25.00 (0.0)	31.25 (0.0)	76.37 (7.6)	75.32 (7.5)

V. CONCLUSIONS AND FUTURE WORK

These preliminary findings indicate that SRL is a viable alternative to the more commonly used IRL. Predictive accuracy is maintained or improved, the need for one parameter is removed entirely, while robustness to value changes of another parameter is greatly improved.

However, there are still many aspects to explore of this alternative strategy, one of which is whether it also provides improved computational performance. With the parameter values as in Table II, the total number of ants and rule evaluations for an IRL or SRL run are shown in Table V. Note that rule and not ruleset evaluations are used for SRL, in order to make the comparison with IRL more equitable. Initial tests suggest that *FRANTIC*'s greatest computational expense is in rule creation so SRL runs may take considerably less time than IRL runs. For instance, for the Water Treatment dataset, though the number of evaluations for SRL is twice the number of evaluations for IRL, IRL runs generally take 3 to 5 times longer, due to the fact that more ants are used during each iteration.

It may even be possible to reduce the number of evaluations done during SRL. 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 for decreasing the computational expense of the SRL evaluation strategy. There are still other questions to be answered though, including exploring the possibility that IRL runs can maintain the current accuracy on these datasets, but use fewer ants and iterations per ACO.

A major assumption in this work is that one rule is sufficient to adequately describe a class, and so for SRL n ACOs are run in parallel where n is the number of classes. Though a useful starting point for investigating this alternative strategy, this may be a naive assumption when using larger and more complex real-world datasets. Work will therefore be carried out to extend 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 V

COMPLEXITY—ITERATIVE VS SIMULTANEOUS RULE LEARNING

	Saturday Morning		Water Treatment	
	IRL	SRL	IRL	SRL
evaluations	7500	4800	3000	6000
ants	7500	300	3000	600

ACKNOWLEDGMENT

The first author is supported by a UK EPSRC grant. Both authors are grateful to John Levine and Stuart Aitken for helpful discussions, whilst taking full responsibility for views expressed in this paper.

REFERENCES

[1] M. Galea, Q. Shen, and J. Levine, "Evolutionary approaches to fuzzy modelling for classification," to appear in *Knowledge Engineering Review*.

[2] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press, 1975.

[3] J. R. Koza, *Genetic programming: On the programming of computers by means of natural selection*. A Bradford Book, The MIT Press, 1992.

[4] O. Cordon, A. Gonzalez, F. Herrera, and R. Perez, "Encouraging cooperation in the genetic iterative rule learning approach for qualitative modeling," in *Computing with Words in Intelligence/Information Systems*, J. Kacprzyk and L. Zadeh, Eds. Pyhsica-Verlag, 1998.

[5] C. A. Pena-Reyes and M. Sipper, "FuzzyCoCo: A cooperative-coevolutionary approach to fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 9, pp. 727–737, 2001.

[6] R. Mendes, F. Voznika, A. A. Freitas, and J. C. Nievola, "Discovering fuzzy classification rules with genetic programming and co-evolution," in *Lecture Notes in Artificial Intelligence 2168*. Springer-Verlag, 2001.

[7] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: From natural to artificial systems*. New York: Oxfor University Press, 1999.

[8] J. Casillas, O. Cordon, and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," in *Proc. 2nd International Workshop on Ant Algorithms*, Brussels, Belgium, Sep. 2000, pp. 13–21.

[9] M. Galea and Q. Shen, "Fuzzy rules from ant-inspired computation," in *Proc. IEEE International Conference on Fuzzy Systems*, Budapest, Hungary, Jul. 2004.

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

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

[12] C. L. Blake and C. J. Merz. (1998) UCI repository of machine learning data. Department of Computer Science, University of California, Irvine CA. [Online]. Available: `http://www.ics.uci.edu/~mlearn/MLRepositary.html`

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

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