Fuzzy Rules from Ant-Inspired Computation

Michelle Galea

Doctor of Philosophy

Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh

2007

Abstract

This research identifies and investigates major issues in inducing accurate and comprehensible fuzzy rules from datasets. A review of the current literature on fuzzy rulebase induction uncovers two significant issues:

A. There is a tradeoff between inducing accurate fuzzy rules and inducing comprehensible fuzzy rules; and,

B. A common strategy for the induction of fuzzy rulebases, that of iterative rule learning where the rules are generated one by one and independently of each other, may not be an optimal one.

FRANTIC, a system that provides a framework for exploring the claims above is developed. At the core lies a mechanism for creating individual fuzzy rules. This is based on a significantly modified social insect-inspired heuristic for combinatorial optimisation – Ant Colony Optimisation. The rule discovery mechanism is utilised in two very different strategies for the induction of a complete fuzzy rulebase:

1. The first follows the common iterative rule learning approach for the induction of crisp and fuzzy rules;

2. The second has been designed during this research explicitly for the induction of a fuzzy rulebase, and generates all rules in parallel.

Both strategies have been tested on a number of classification problems, including medical diagnosis and industrial plant fault detection, and compared against other crisp or fuzzy induction algorithms that use more well-established approaches. The results challenge statement A above, by presenting evidence to show that one criterion need not be met at the expense of the other. This research also uncovers the cost that is paid – that of computational expenditure – and makes concrete suggestions on how this may be resolved.

With regards to statement B, until now little or no evidence has been put forward to support or disprove the claim. The results of this research indicate that definite advantages are offered by the second simultaneous strategy that are not offered by the iterative one. These benefits include improved accuracy over a wide range of values for several key system parameters. However, both approaches also fare well when compared to other learning algorithms. This latter fact is due to the rule discovery mechanism itself – the adapted Ant Colony Optimisation algorithm – which affords several additional advantages. These include a simple mechanism within the rule construction process that enables it to cope with datasets that have an imbalanced distribution between the classes, and another for controlling the amount of fit to the training data.

In addition, several system parameters have been designed to be semi-autonomous so as to

avoid unnecessary user intervention, and in future work the social insect metaphor may be exploited and extended further to enable it to deal with industrial-strength data mining issues involving large volumes of data, and distributed and/or heterogeneous databases.

Acknowledgements

Acknowledgement and thanks go principally to the following people for their continuous support, guidance, and patience:

- Professor Qiang Shen, University of Aberystwyth, my principal supervisor, for encouraging me to start this research and providing expert advice all along the way;
- Dr John Levine, University of Strathclyde, and Dr Stuart Aitken, University of Edinburgh, who have at different times taken on the role of second supervisor;
- Mark Westwood, Max and Jamila of Pilot Solutions Ltd, Edinburgh, for invaluable advice on object-oriented analysis, design and programming;
- Rónán Daly and Dave Murray-Rust, University of Edinburgh, always happy to discuss findings and to help in general; and,
- Dr Gillian Hayes, University of Edinburgh, and Dr Alex Freitas, University of Kent, my thesis examiners, for their time and constructive comments.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Michelle Galea*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This research investigates the induction from empirical data of rulebases that are comprised of accurate and comprehensible fuzzy IF-THEN classification rules.

In this work an explicit distinction is made between the process of creating individual rules, and the process of selecting which ones should form the final rulebase. The system developed during this research, *FRANTIC*, has been designed with flexibility in mind, in order to investigate the potential of a constructionist mechanism inspired by real ants for inducing individual rules, and the impact that the strategy for selecting the rules that make up the final rulebase may have on its quality.

This chapter sets the scene for the research undertaken. It describes the motivation, the aims and objectives, the investigative approach adopted, and the achievements and conclusions drawn. It also describes the organisation of this thesis.

1.1 Motivation

This research investigates the extraction of fuzzy linguistic rules from datasets using computational methods modelled on social insect behaviour. This section answers the following questions:

1. Why address automated knowledge discovery?
2. Why use fuzzy linguistic rules as the knowledge representation language?
3. Why approach the problem using ant-inspired techniques?

Automated knowledge extraction

Knowledge discovery in databases [55] is a term commonly used to refer to the process of discovering useful knowledge from data. Automated techniques for extracting knowledge from data is a research area that benefits different application contexts.

One context is the development of intelligent reasoning systems that are based on a core knowledge base, such as fuzzy rule-based systems (see Appendix A). This knowledge base has traditionally been determined via discussions with domain experts but this approach suffers from many problems and shortcomings [22] – 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 is a specific and core activity within the broader knowledge discovery process, and is defined by Hand *et al*. [86] as:

> ...the analysis of (often large) observational data sets to find unsuspected relationships and to summarise the data in novel ways that are both understandable and useful to the data owner.

Here, the purpose has generally been to capitalise on the large volumes of transaction data that are generated by many organisations – industrial, commercial, and governmental – and process it in search of new knowledge that may be incorporated into their operational, managerial and strategic planning procedures.

These techniques are also increasingly being used in scientific fields such as astronomy, chemistry, biology and genetics. Traditionally, scientific endeavour has started with a hypothesis and gathered experimental data that is analysed in order to provide evidence that supports or disproves the hypothesis. However, new and/or improved technology – for example, high-throughput genetic micro-array processing, a new generation of earth observation satellites, and more powerful computer storage and processing devices – allows scientific research to generate and collect more data than ever before. This is often data about which scientists have not yet hypothesised, and this may therefore lead them to treat and analyse it from a data mining perspective, i.e. new scientific knowledge may be produced as a result of a data-driven rather than a hypothesis-driven approach.

The requirement for effective techniques for inducing knowledge from empirical data is greater than ever, and research in this area is eminently justifiable.

Fuzzy rules and rule-based systems

Many real-world problems contain a measure of noise and imprecision, and automating processes in these situations necessitates a capability to capture and reason with imprecise or inexact knowledge.

Fuzzy IF-THEN production rules, and fuzzy rule-based systems, are based on fuzzy set theory and fuzzy logic [195], where everything is a measure of degree [199]. This allows such systems to reach conclusions and make decisions even in circumstances where the available knowledge and data is not always precise. The success of this approach is evidenced by many applications in industry and commerce (e.g. [10; 90; 151]). Appendix A provides an introduction to the fundamental concepts of fuzzy set theory and rule-based systems, in so far as it is necessary to understand the work conducted during this research.

A distinction is commonly made between *approximate* fuzzy rules, and *linguistic* or *descriptive* fuzzy rules [130]. The focus of the former type is on the accuracy of the rules in describing the underlying patterns of the data from which they are induced, and the focus of the latter is on their ability to describe these patterns in a form comprehensible to humans. Since the contexts for automated knowledge discovery described in the previous subsection generally necessitate the extraction of knowledge in a form which humans can validate, the focus of this research is on using descriptive rules to represent extracted knowledge.

Though fuzzy rules are often used as an integral part of an automated decision-making system, their inherent comprehensibility also makes them an effective form of knowledge representation to use for data mining purposes. Furthermore, comprehensible knowledge in general may help to validate a domain expert's knowledge and provide confidence in the automated system affecting decisions [150], highlight previously undiscovered knowledge [176], and indicate both significant and insignificant features (or attributes) of the domain [97].

Ants and combinatorial problems

The induction of an individual rule as implemented in this thesis is a combinatorial optimisation problem, one of finding an optimal subset of terms to accurately describe a class in the dataset being processed. A term is defined as a condition found in the antecedent or IF- part of a rule, e.g. OUTLOOK=Sunny.

The approach adopted for creating individual rules is that of Ant Colony Optimization (ACO) [50]. This is a heuristic for solving combinatorial optimization problems inspired by the foraging behaviour of real ants, which have the ability to find the shortest path between their nest and a food source. ACO is increasingly and successfully being applied to different combinatorial

optimization problems [50], and its success lies mainly in trying to emulate fundamental characteristics of social insect colonies such as robustness (arising from the ability of the society as a whole to survive when individuals may fail), flexibility (from the ability to adapt to changing environments) and decentralisation (the colony as a whole works even though there is no one leader to direct other members).

In ACO, each artificial ant is considered a simple agent, communicating with others indirectly by laying a chemical called a pheromone on the path it travels, and thereby affecting changes to a common environment. An artificial ant constructs a solution incrementally, by traversing a construction graph whose nodes are components that form part of a solution. In the case of constructing rules, each node represents a term that may form part of a rule antecedent. The selection of the next component is guided by both background information provided by the training set, and information learned as the ACO algorithm runs through successive iterations. The output of one ACO in this work is one fuzzy IF-THEN rule, the best of all rules created by the artificial ants, and as determined by some fitness criterion. A number of ACO algorithms need therefore be run in order to create a complete rulebase that describes all classes in a dataset.

ACO, similar to other stochastic population-based algorithms such as genetic algorithms [93] and genetic programming [93], provides an effective mechanism for simultaneously exploring different regions of large solution spaces, and a simplicity of implementation that requires minimal understanding of the problem domain. Its solution construction mechanism, where each component of a solution is incrementally added to a partial solution, uses randomness to deliberately avoid always adding the next 'best' (as defined by some criteria) component to the solution being built; this is in direct contrast to greedy approaches, and is done in the expectation of avoiding local minima. However, this constructionist approach affords specific advantages for rule induction. These include a simple mechanism within the rule construction process that enables it to cope with datasets that have an imbalanced distribution between the classes, and another for controlling the amount of fit to the training data.

ACO is described in more detail in Chapter 3, where the available literature on its application to crisp and fuzzy rule induction is also reviewed.

1.2 Aims and Objectives

The aim of this research is to identify and investigate the major issues in inducing accurate and comprehensible fuzzy rules from data, using ACO as the rule discovery mechanism.

Supporting objectives are to:

1. identify the main issues specific to current methods of fuzzy rule induction
2. determine possible solutions to alleviate or resolve these issues
3. implement a framework for investigating the potential solutions identified
4. establish an experimental methodology and evaluate the solutions identified
5. analyse results and formulate conclusions.

The current literature on the induction of fuzzy rulebases suggests two significant and related points:

- that there is a tradeoff between inducing accurate fuzzy rules or inducing comprehensible fuzzy rules, and
- that a common strategy for the induction of fuzzy rulebases, that of iterative rule learning where the rules are generated one by one and independently of each other, is not an optimal one.

This work attempts to challenge the first statement, and to provide evidence for the unsupported claim made by the second statement.

1.3 Approach

A starting point is to induce descriptive fuzzy rules, and explore ways of increasing their accuracy. To this end, *FRANTIC* is developed, a tool for inducing fuzzy linguistic rulebases with the flexibility to allow one to investigate the issues identified above. At the core of the system is a mechanism for creating individual fuzzy rules, which is based on an ACO algorithm. The rule discovery mechanism is then utilised in two very different strategies for the induction of a complete fuzzy rulebase:

1. The first follows the common iterative rule learning approach for the induction of crisp and fuzzy rules;
2. The second has been designed during this research explicitly for the induction of a fuzzy rulebase, and generates all rules in parallel.

Current fuzzy rule induction algorithms are often simple extensions of classical crisp rule induction algorithms, and therefore many fail to sufficiently take into consideration a fundamental difference between crisp and fuzzy rules – a crisp rule either matches or does not match an instance requiring classification, while most generally speaking, all fuzzy rules match all instances but to varying degrees (including a degree of zero that indicates no match). In the

latter case, it is important that fuzzy rules describing different classes do not end up competing with each other to classify the same instances. The interaction between fuzzy rules during the inference process, can therefore be very different from the interaction between crisp rules.

The alternative strategy to iterative rule learning which is introduced in this work is in recognition and acceptance of this difference – fuzzy rules describing different classes are required to complement, rather than compete with each other during classification. The learning process should therefore take this into account. A basic premise in this work is that a strategy whereby rules describing different classes are constructed and evaluated together, will result in rulebases with fuzzy rules that interact optimally during inferencing. The other avenue explored for increasing the accuracy of fuzzy linguistic rules is that of increasing the richness of the knowledge representation language – the premise here is that a more expressive language may be able to more accurately capture the underlying patterns in a dataset.

Experiments for this research are designed to gain a 'holistic' understanding of the system implemented - how and why it works, and what its current limitations are. At all times, a principled and systematic manner is used to set system parameters. These parameters control different elements of the rule construction mechanism and the different overall strategies; in appreciating how the system works, one therefore gains a deeper understanding of the issues involved in fuzzy rulebase induction, and how they may be alleviated or resolved.

1.4 Preview of Contributions

As described in more detail in later chapters, rulebase quality is determined by its accuracy on test datasets, and its complexity or size, while *FRANTIC* system operation is also considered in light of robustness to changes in parameter values, and computational complexity.

Analyses of experiment results provide evidence to support both basic premises mentioned in the previous section – accuracy of the resulting rulebases may be improved by increasing the richness of the hypothesis language, and/or by changing the overall strategy for inducing a complete rulebase. Additional benefits such as an increased robustness to key *FRANTIC* parameter changes are also observed.

The improvement in accuracy is *not* obtained at a cost to the comprehensibility of the induced knowledge, and the performance of *FRANTIC*-induced rulebases is compared with that of models induced by other learning algorithms, including some well-recognised for the accuracy of the models they produce.

However, *FRANTIC*'s strength and potential lie not only in its ability to deal with fuzzy-specific rulebase induction issues, but also in managing other issues relevant to both crisp and fuzzy rule

induction. These include dealing with datasets that have an unequal distribution between the classes, preventing over-fitting to the training data, and providing mechanisms for balancing the generalisation and specialisation of the constructed rules (i.e. avoiding covering too many or too few instances in the training data).

The results chapters provide several key insights into *FRANTIC*'s performance and its ability to induce accurate and comprehensible rulebases. These include an understanding of:

- the role that the background knowledge used by ants during rule construction plays in speeding up the discovery of optimal rules, though not necessarily in improving their predictive accuracy;
- the impact of the artificial pheromone used by ants during rule construction, and its crucial interaction with how the best rule of an ACO algorithm is selected, and the size of the solution space being searched;
- the effect on rulebase quality of the different forms of knowledge representation that an ant may utilise during rule construction (more expressive vs. less expressive forms of IF-THEN rules);
- the computational expense involved in creating an individual rule, and its dependency not only on the size of the construction graph an artificial ant uses to construct its rule, but also on the construction validation method that ensures only valid rules are created;
- the interaction between the two main parameters involved in rule construction, how together they provide a balance between generalisation and specialisation of the constructed rules, and how they may be used to control the size of the solution space being searched;
- the use of the `minInstPerRule` parameter[1] for constructing rules, and how it enables the system to construct accurate rulebases from datasets that have a highly imbalanced distribution between the classes; and,
- the impact that simultaneously constructing and evaluating rules to describe all classes in a dataset has on both rulebase quality and system robustness.

Such a fundamental appreciation of how and why the system works may then be used to provide practical guidelines for its appropriate use, and to direct further future work. These are covered in more detail in the latter half of this thesis, Chapters 6–9.

[1] This parameter stipulates the minimum number of training instances that a rule must cover.

1.5 Document Structure

The structure of this thesis generally follows and meets the research objectives listed in Section 1.2 on page 4.

The next chapter reviews the literature for work on inducing fuzzy rulebases. It describes the more established approaches to this problem, especially that of evolutionary computation, and highlights common issues such as obtaining a balance between the accuracy and comprehensibility of the induced rulebases.

Chapter 3 then introduces ACO, outlines its potential benefits and how it may be applied to general combinatorial optimisation problems. It also describes how ACO has been applied to rule induction problems.

Chapter 4 describes *FRANTIC*, the implemented system. The ACO-based rule construction mechanism is presented in detail, as is its use in two very different strategies for creating a complete rulebase. The first approach follows a common iterative approach where ACOs are run in succession, while the second approach, introduced in this work, instead runs ACOs simultaneously.

Chapter 5 details the methodology adopted for designing and conducting experiments, and the evaluation criteria used in their analyses. It also specifies how an equitable comparison with other learning algorithms is ensured, by replicating the testing environment between algorithms.

Chapter 6 reports and analyses the results of several experiments designed to explore the different elements of the ACO rule construction mechanism, while the experiments discussed in Chapter 7 are designed to highlight the advantages and limitations of the two different rulebase induction strategies.

Chapter 8 then pits *FRANTIC* against several well-established crisp and fuzzy learning algorithms, including decision trees and support vector machines. The results are extremely promising and *FRANTIC* exhibits a robustness to inherent dataset characteristics, such as class distribution and separability, that is not observed in the other algorithms.

Chapter 9 summarises the research presented and the contributions made. It also points to future work, aimed at further enhancing *FRANTIC*'s capabilities, whilst continuing to build on the achievements of this thesis by enhancing current strategies, and exploring new ones, specifically designed for the induction of fuzzy rulebases.

Several appendices follow providing more detailed information on fuzzy rules and rule-based systems, the experimental setup, and supplementary results to Chapters 6–8.

Chapter 2

Fuzzy Modelling for Pattern Classification

This chapter presents an overview of approaches and issues in fuzzy knowledge acquisition for classification purposes. The meaning of the term 'knowledge acquisition' has been broadened in this work to include both automated knowledge acquisition for the development of intelligent reasoning systems, and knowledge discovery from large databases.

Learning techniques have been used to generate a fuzzy rulebase, or fine tune the membership functions of rules in a rulebase, or both. Since this thesis investigates rulebase induction, the focus of this review is on research exploring rulebase generation, where membership functions have been pre-defined (whether by a user or other automated process). For the reasons discussed in Section 1.1, an emphasis is placed on inducing knowledge that is comprehensible to a human user. This chapter therefore describes the induction of descriptive fuzzy rules using linguistic variables in both antecedent and consequent, as opposed to approximate fuzzy rules whose main focus is accuracy.

There is a bias in this review towards work utilising Evolutionary Computation (EC) for rulebase generation, that is justified in the following paragraphs. EC is the application of methods inspired by Darwinian principles of evolution to computationally difficult problems. Evolutionary algorithms (EAs) re-iteratively apply genetic-inspired operators to a population of solutions, modifying or replacing members of the population so that on average each new generation tends to be better than the previous one, according to some predefined fitness criteria. EAs have been extensively and successfully applied to combinatorial and search problems. Reasons for their popularity include their broad range of application (e.g. robotics, control, and natural language processing), and their relative simplicity of implementation that requires little domain knowledge. More detailed discussion of EC strengths, issues, and theoretical foundations may

be found in [9; 58; 112; 190].

One reason for the bias is that there is a rich literature that utilises such techniques for fuzzy modelling in general, and for fuzzy rulebase generation in particular. The branches of EC that have been mainly applied to this task are genetic algorithms (GAs) [93] and genetic programming (GP) [120]. Other major branches are evolutionary programming (EP) [59] and evolution strategies (ES) [162]. These branches are similar to each other and differ mainly in the representation used for the individuals in a population (e.g. binary strings vs. real-valued vectors), and in the application of the genetic operators such as recombination and mutation of individuals.

Another reason is that very little work has been carried out on rule induction using ACO, but there are high-level similarities between EC and Swarm Intelligence (SI), the field of research that includes ACO – both are inspired by nature, both are population-based approaches that search diverse areas of the solution space simultaneously, and both use an element of randomness to avoid converging to local solutions, and to deal with the intractability of large solution spaces. Relevant EC literature is therefore reviewed in order to identify issues pertaining to rule induction that may also be applicable when using an SI approach. Since ACO is highly pertinent to this thesis, the small body of literature utilising ACO for rule induction is reviewed in the next chapter, after the main features of ACO have been described.

It should be noted for copyright reasons, that where sections of this chapter review an EC-based approach, these are generally excerpts from [68]. This publication is co-written by the author of this thesis, and provides a more comprehensive review of evolutionary approaches to fuzzy modelling for classification. It includes reviews of work for knowledge base induction (i.e. induction or refinement of both the fuzzy rules and membership functions), and also addresses the common evaluation criteria used for induced knowledge in the context of developing intelligent reasoning systems and data mining.

The main content of this chapter is divided into two parts: Section 2.1 reviews the common strategies used in the induction of highly comprehensible fuzzy decision trees and rulebases, and Section 2.2 describes different approaches for improving the accuracy of comprehensible models.

2.1 Inducing Comprehensible Fuzzy Models

This section provides an overview of the existing literature from the perspective of the representation of induced knowledge. Due to their inherent comprehensibility the focus is placed on work utilising decision trees for their solution, or various forms of IF-THEN rules.

Artificial neural networks [14] are another popular form for representing induced knowledge. As is commonly recognised, these models have excellent generalisation capabilities when it comes to classifying previously unseen instances, but generally offer little explanatory advantages to the knowledge user. This is why much research nowadays is focused on extracting comprehensible rules from neural networks [6; 51] and [105, chp.13], while other work utilises neural networks to simultaneously induce and/or refine a knowledge base, i.e. both the fuzzy rules and associated membership functions [33; 140]. Though such research is outside the scope of this review, it should be noted that in order to test whether both highly comprehensible and competitively accurate rules may be induced, *FRANTIC*, the system developed during this research, is compared to one such algorithm called *NEFCLASS* [140] (Chapter 8).

It is a similar matter for classifiers induced using support vector machines (SVM) [183]. In recent years they have emerged as strong competitors to neural networks with regards to generalisation power, though again, the models produced are not particularly human comprehensible. *FRANTIC*'s output is also compared with that of an SVM, along with other crisp and fuzzy induction algorithms. More details about these algorithms are provided in Sections 5.1 and B.2.

2.1.1 Decision Trees

Decision trees are a popular hypothesis language as they are easy to comprehend and give an explicit model of the decision-making process. An internal node of an induced decision tree specifies a test on an attribute of the data set (though more complex trees may be built by specifying tests at nodes based on more than one attribute). Each outgoing branch of the node corresponds to a possible result of the test, and leaf nodes represent the class label to be assigned to an instance. Each individual path from root to leaf node of the tree may also be equated with an IF-THEN rule.

To classify an instance, a path from the root node of a *crisp* decision tree is traced to a leaf node – each internal node reached specifies which outgoing branch should be taken, depending on the value of the relevant attribute in the instance. The instance is assigned the class label of the leaf node reached at the end of the path. In *fuzzy* decision trees, each test at an internal node is a fuzzy one, and results in membership values for *all* edges branching out of the node; this is because an observed measurement for one linguistic attribute results in matches to all values of that attribute, but to different degrees. In line with inference in fuzzy rule-based systems (Section A.3), decisions that need to be made for using fuzzy decision trees for classification include how to aggregate the membership values along a path, how to aggregate the resulting values of paths labelled with the same class, and how to finally assign a class to an instance.

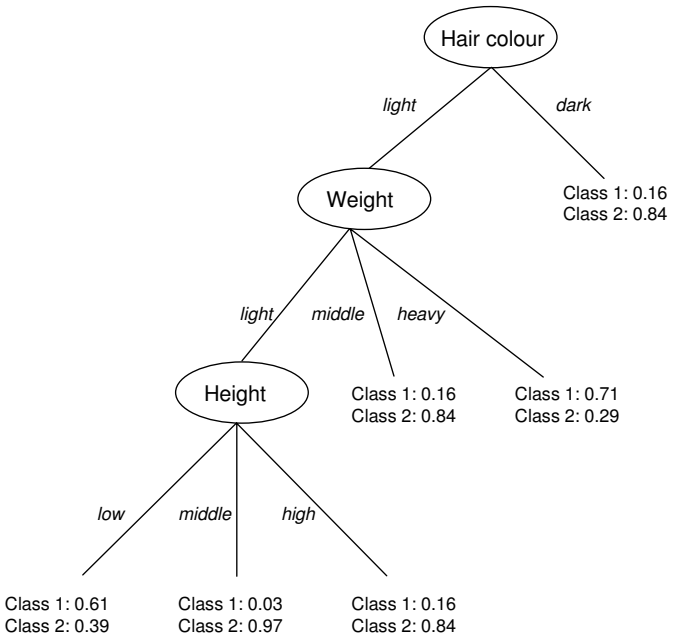


Figure 2.1: An example fuzzy decision tree

Divide-and-conquer strategy

In traditional machine learning, decision trees are induced by following a divide-and-conquer approach or strategy – the training data is partitioned into disjoint subsets (one for each value of the attribute selected), and the algorithm is applied recursively to each subset. The criterion for splitting the dataset selects the attribute that appears to best classify the training examples.

Such learning algorithms generally use an attribute-value representation as the example language, i.e. each instance in the training data is described by specific values of a given set of attributes. The domain of each attribute may be nominal or numerical. The numerical attributes are either handled directly by the algorithm or discretised before the induction of crisp decision trees, or they are fuzzified prior to the induction of fuzzy decision trees. Ultimately, such attributes are translated to finite domains.

In fuzzy tree induction algorithms, thresholds are used to determine when to stop expanding a branch, including ones based on the proportions of the classes of instances that are in the current node, and/or the proportion of all instances in the current node when compared to all instances in the training set, e.g. [182]. Figure 2.1 depicts a simple fuzzy decision tree from [182] – each leaf at the bottom of the tree attaches a certainty or confidence measure to the different classes, based on the exampless/instances at the leaf.

Many fuzzy decision tree induction algorithms are inspired by the *ID3* algorithm [157] for the induction of crisp decision trees. In *ID3* the splitting criterion is information gain, which

measures the expected reduction in Shannon's entropy [167] that is achieved by partitioning the training data according to the values of an attribute. References [107; 182] use selection measures based on fuzzy entropy [128], [193] uses a measure of ambiguity or nonspecificity [89] of the classes over the training set, and [188] uses yet a different measure to determine the degree to which an attribute contributes towards classification. Reference [188] compares its own proposed selection criterion with the one described in [193], and a fuzzy entropy-based one. The difference in accuracies obtained by the resulting decision trees is not considerable. This is in line with the conclusions drawn in equivalent comparative studies on the use of different selection criteria for the induction of crisp decision trees [135] (though [24] suggests that random selection of attributes does lead to increased errors).

Other research on fuzzy trees suggests that smaller decision trees may lead to better generalisation, as this may help prevent over-fitting to the training data. There are different ways for reducing the size of a fuzzy tree, including the more common ones of using thresholds similar to the ones described above, to determine when to stop growing individual branches [182], and pruning branches after tree construction has been completed [107].

Reference [96] instead uses a pre-construction step to select the best features with which to build a fuzzy decision tree, and tests the outcome on the fuzzy induction algorithm described in [193]. As perhaps might be expected, trees that are built from extracted feature subsets are smaller than ones that are built from the full set, but they also show an improved accuracy on several datasets, supporting the thesis that smaller trees are less likely to over-fit to training data. No comments are forthcoming, however, as to whether the time taken for the extra initial step might be compensated for by a reduction in the time taken to build a tree from fewer features.

Reference [187] uses yet a different approach. Everytime a new node (attribute) is added to the decision tree, fuzzy clustering is used on the training examples in the node to see whether some values of the attribute may be merged. This leads to fewer terminal leaves in the tree. The authors study the impact of this merging step on a fuzzy entropy-based tree induction algorithm – higher accuracy and fewer leaves in the resulting decision trees are observed, than if the merging step had not been used. The authors concede that the training time increases, but conclude that the extra time incurred merging attribute values (upwards of half again the time taken if value merging were not used), might be a reasonable price to pay for an average increase in accuracy of 11% over three datasets tested. It is not clear, though, whether the trees with a reduced number of leaves (a significant reduction of an average of 28% fewer nodes over three datasets), have similar path lengths to the trees induced without attribute value merging.

Evolutionary computation approaches

Decision trees have been induced using EC in one of two different ways:

- an EA (evolutionary algorithm) evolves a population of decision trees, or
- a more traditional decision tree induction algorithm integrates an EA, generally as part of its splitting mechanism for the training data.

In the former approach, GP has generally been used to evolve solutions, as it is the tree representation utilised by GP that makes it a natural candidate for the representation of decision trees [119].

Reference [54] provides an example of work implementing this approach for fuzzy decision tree induction. Each continuous attribute of a dataset is fuzzified by first applying a clustering algorithm on the values, and next a membership function is defined for each cluster. GP is then used to evolve a population of fuzzy decision trees. The function set contains conditions of the form ATTRIBUTE=VALUE where value is either a nominal value or a linguistic term with underlying fuzzy semantics, and the terminal set contains the different possible classes. When classifying an instance a fuzzy membership value for each of the classes is computed and the class with the greatest value is the one assigned to the instance.

This algorithm is tested on several benchmark datasets and the classification results are compared with those obtained by several other algorithms. They include the crisp decision tree induction algorithm *C4.5* [158] and its bagged [21] and boosted [61] variants, an evolutionary induction algorithm for crisp rules called *ESIA* [127], and another system for fuzzy rule induction called *CEFR-MINER* [131]. The authors conclude that the bagged and boosted *C4.5* variants perform better than the original algorithm and that their own algorithm's performance is comparable with that of *CEFR-MINER*. It appears that *ESIA* performs worst overall, but it is difficult otherwise to judge between the *C4.5* variants and the evolutionary fuzzy algorithms. This is hampered by the fact that several results for some of the datasets for some algorithms are missing from the report.

The second way of utilising EAs in the induction of fuzzy decision trees is by creating a hybrid algorithm such as in [106; 159].

In [106], for instance, a fuzzy decision tree-building algorithm is augmented with a dynamic optimising procedure for the node partitioning. The basic algorithm is based on *ID3* while the optimising procedure is again a GA. After all attributes' values have been fuzzified, this algorithm recursively partitions the training data set until a termination criterion is satisfied. Note that the training examples within a node are not partitioned further if they all belong to the same class, or all attributes have been used in the path leading to the node, or the information

content of the node reaches a certain threshold level. Nodes are then processed according to an ordering measure—the node containing the greatest number of examples is processed first, and the partitioning of the training examples is then dependent on selecting the attribute that maximises the information gain measure based on entropy.

Once an attribute has been selected its information content, based on the specific examples in the node, is minimised by modifying its original fuzzy sets. This dynamic redefinition of the fuzzy sets of an attribute softens the constraint made by *ID3* and its fuzzy adaptations, that real-valued attributes must be partitioned (or fuzzified) just once prior to tree-building – it is aimed at increasing the consistency of the fuzzy tree. It should be noted that in order to preserve the comprehensibility of the induced tree, at the possible expense of generalisation capabilities, the number of fuzzy sets of an attribute remains fixed during tree-building and the optimisation procedure is carried out only once.

2l-FDT (*2-l*evel *F*uzzy *D*ecision *T*ree)[159] creates a fuzzy decision tree by combining two different fuzzy clustering techniques and a GA. In the first phase partitions of the training data are recursively determined by a fuzzy C-means based clustering algorithm [12; 72] integrated with a GA that decides on the best feature subspace for each partitioning. The GA fitness function evaluating each feature subspace is based on a fuzzy adaptation of the information gain measure. The termination criteria for the recursive partitioning include the number of training examples in the final partitions, the size of the tree, and the information gain criterion itself. At this point, the second phase, a second fuzzy clustering algorithm as reported in [121] is applied on the training examples, once for each class in each of the final partitions or nodes resulting from the previous level; a final leaf is added for each subclass determined.

The resulting fuzzy decision tree outperforms the decision tree produced by *C4.5* on both classification accuracy on a test set, and in comprehensibility, i.e. *2l-FDT* produces trees considerably smaller in size and therefore more easily interpretable by human experts.

2.1.2 Production Rules

The majority of work in fuzzy knowledge acquisition has induced knowledge in the form of simple propositional IF-THEN rules, i.e. a conjunction of crisp or fuzzy conditions leading to a crisp or fuzzy conclusion. These are generally equivalent to the rules that may be extracted from decision trees. The example language, i.e. the training data language, is generally an attribute-value representation.

More expressive rules are propositional rules with internal disjunction between attribute values, negations of attribute values, and disjunctions between attributes. The rules induced may or may not include a confidence measure that is used in fuzzy reasoning when classifying a new

instance. Examples of work inducing such fuzzy rules include [102] – simple propositional, [194] – internal disjunction, and [131] – disjunction between attributes and negation. These rules may take the following form:

$$Rule\ R_j : IF\ A_1\ is\ (v_{11}\ OR\ v_{12})\ AND\ A_2\ is\ v_{21}\ AND ...\ AND\ A_n\ is\ NOT(v_{n1})$$
$$THEN\ Class\ is\ C_j\ with\ CF = CF_j$$

where A_1 to A_n are the attributes in a data set, v_{ik} is a specific linguistic term of attribute A_i, C_j is the rule consequent of rule R_j, and CF_j is the rule confidence factor.

Depending on the learning algorithm, incomplete fuzzy rules may be induced, i.e. not all attributes need be present in the rule antecedent, leading to shorter rules that may also therefore be more comprehensible. For instance, in [194] where a dataset has four attributes (OUTLOOK, TEMPERATURE, HUMIDITY and WIND), rules may be generated that do not include all four attributes, such as:

$$IF\ (\text{OUTLOOK}\ is\ Sunny\ OR\ Cloudy)\ AND\ \text{TEMPERATURE}\ is\ Hot\ THEN\ Swimming$$

In the context of data mining (as opposed to the development of fuzzy rule-based reasoning systems), where the data being mined often contains a large number of attributes, such incomplete rules are the norm rather than an exception, as this can help prevent over-fitting to the training data.

By far the most commonly used approaches for inducing fuzzy knowledge are based on EC; not only are there numerous references to be found in appropriate journals and conference proceedings, but the body of research has grown to the extent that several volumes on various aspects of fuzzy knowledge base induction and refinement have now been published. These include [30; 31; 44; 105], which are either entirely devoted to the use of genetics-inspired techniques for such work, or are heavily biased in favour of this approach.

A few non-EC based examples inducing descriptive rules for pattern classification are [36; 40; 185]. Reference [185] follows a separate-and-conquer approach very common in the induction of crisp rules [62]. This strategy is also known as set covering, or sequential covering. For each class – or concept, as it is generally referred to in crisp rule induction – this strategy finds one or more rules to describe that concept. Instances in the training set that describe this concept are referred to as positive instances, while all other instances are negative. After finding a rule that is to be added to the final rulebase, the instances in the training set that are covered by this rule are removed, before finding other rules to describe the same concept. When all rules for describing a concept have been found, the training set is reinstated to its full original size, and instances are relabelled as positive and negative depending on the next concept to be described.

In [62] each rule to describe a concept is incrementally constructed by adding attribute-values (or terms) from the dataset that best meet a selection criterion based on maximising fuzzy information gain. The addition of terms to the rule antecedent stops as soon as the rule attains a sufficiently high strength, as determined by a user-defined parameter and based on a measure of how well the rule covers training instances. Reference [40] follows the same separate-and-conquer strategy, but its discovery mechanism is different – it starts out with a very general rule covering all instances in the training set, and incrementally specialises the rule (maintaining a fixed number of best rules at any one time) until a rule quality criterion is satisfied; the best rule of the ones the search process has been maintaining is then selected and added to the final rulebase. Reference [36], on the other hand, partitions the training data into subsets depending on the classes in the dataset, and for each subset (one per class), constructs a rule. Each rule is constructed by selecting the terms that best describe the specific class under consideration; the selection criterion is based on subsethood values [118], which provides a measure of the degree to which one fuzzy set is a subset of another fuzzy set.

Other approaches to fuzzy IF-THEN rule induction include grid partitioning methods (e.g. [186]) and gradient descent learning (e.g. [144]), as well as the neural network and SVM approaches previously mentioned. These, however, are generally applied to the induction of non-descriptive fuzzy rules, and/or to function approximation, prediction or control problems.

Overview of evolutionary computation for rule induction

From some of the early work on evolutionary-inspired systems two terms have emerged that are still in common usage today: Michigan-style [94] and Pittsburgh-style [173], the names given in recognition of the institutions of the originators of the two approaches. In the first approach, a rule is encoded as one individual of the population, whilst in the second later approach, a rulebase is encoded as one individual of the population. These terms are still used in discussing GAs, but within this review their meaning has been extended to describe the encoding of solutions within the other branches of EC where appropriate.

Figure 2.2 illustrates the low-level dependency that exists between what an individual represents, and the number of individuals selected as a solution from the final population of an EA. This interdependency gives rise to the different EC-based induction strategies for fuzzy (or crisp) rulebase generation. If a basic EA is run (Fig. 2.3), then a population of individuals is evolved for several generations (iterations) and the solution to the problem is selected from the final generation. If an individual represents only one rule, then the whole of the final population, or a subset of it, is selected, leading to one of the basic EA strategies (top right square in Fig. 2.2).

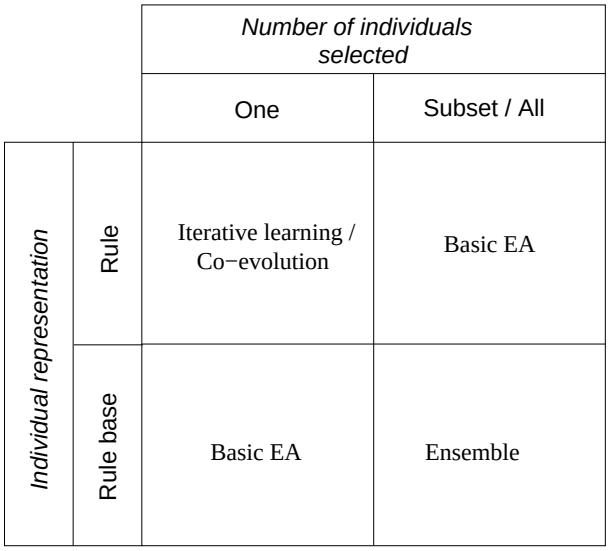


Figure 2.2: Inter-dependency between induction strategy, representation of individuals, and number of individuals selected

If, on the other hand, only one individual (rule) is selected, then this is a partial solution. Several more EAs must therefore be run to complete a rulebase. If the EAs are run in succession, this leads to an iterative rule learning (IRL) strategy (top left corner in Fig. 2.2). This strategy mirrors the separate-and-conquer strategy discussed in the preceding subsection and applied in deterministic rule learning algorithms. In a separate-and-conquer strategy, the class (or concept) of the rule being built is pre-determined. In EC-based IRL, however, the class may or may not be predetermined. It is therefore useful to distinguish between the two strategies, and in this review the term IRL is used instead of separate-and-conquer to denote denote EC-based approaches.

If several EAs are run, then they may also be run simultaneously instead of in succession, and this leads to a co-evolution strategy (top left corner in Fig. 2.2). Most work employing co-evolution for fuzzy modelling is used for knowledge base induction (rather than rulebase induction)[1], and often one population evolves fuzzy rulebases or decision trees, and the other associated fuzzy functions [131; 153]. Reference [111] does use a multi-population form of co-evolution to generate just the rulebase, with each population eventually contributing one rule. However, the rules are of the Takagi-Sugeno type [180], where the conclusion is a linear combination of the variables in the rule antecedent, and therefore not particularly comprehensible. The authors test the rulebases evolved by their co-evolutionary system on a fuzzy control problem under various initial conditions, and the results are compared with those obtained by

[1]In fuzzy rule-based systems the knowledge base is composed of the rulebase and the database. The latter denotes the membership functions that define the fuzzy sets found in the rules. A brief review of fuzzy rule-based systems and fundamental terminology is found in Appendix A.

rulebases generated by an evolutionary (but not co-evolutionary) system. The co-evolved fuzzy logic controllers outperform the traditionally evolved controllers on generalisation capability and computational efficiency. This lends some support to the idea that evolving fuzzy rules simultaneously leads to their optimal interaction during inferencing.

If an individual represents a rulebase and one individual is selected as a solution, then this leads to the other basic EA strategy (bottom-left square). However, if several rulebases are selected from the final generation, then these form an ensemble (bottom-right square), where the votes of the different rulebases are combined when classifying instances. Research indicates that very often an ensemble performs better than any of the individual rulebases making it up [143]. However, the combination of individual rulebase decisions makes their interpretation and validation by human users difficult, which makes their induction and use outside the scope of this review.

The remainder of this section provides an overview of how a solution (rulebase) or partial solution (rule) is encoded, and reviews work on linguistic rulebase generation using the basic strategies and the IRL approach.

Michigan vs. Pittsburgh style individuals

When only the rulebase is being induced, the membership functions are predefined either by a human expert or some other process (e.g clustering), and remain fixed throughout the inductive process. If Michigan-style encoding is adopted, then the EA generally evolves individuals that represent either a rule antecedent or an entire rule.

In the first case, at each iteration a separate deterministic procedure is used to determine the rule consequent before evaluating the rule for fitness, e.g. [138]. Alternatively, the rule consequent, i.e. class, may already be specified for each rule antecedent during the entire run of EA. This is where a variant of IRL is followed, and an EA is run several times in succession with each run concentrating on evolving rule antecedents pertaining to a specific class, e.g. [77; 163].

If a complete rule is encoded in an individual, then the rule consequent may also be subject to evolution and change by the genetic operators. A restriction may be placed on the crossover operator to ensure that only parents belonging to the same rule consequent are combined to produce offspring [184; 194].

With a Pittsburgh-style approach, generally both rule antecedents and associated consequents are encoded. The genetic operators may then act on the individual rules within a rulebase, or on the composition of the rulebase itself. An alternative is proposed in [192], where each individual represents a fixed number of rule antecedents. The consequents of the rules are

```
(1)   Create initial population
(2)   WHILE termination condition false
(3)       Evaluate new population
(4)       Perform selection for reproduction
(5)       Perform recombination
(6)       Perform mutation
(7)       Generate new population
(8)   ENDWHILE
(9)   Output best solution/s
```

Figure 2.3: Basic induction strategy – a generic evolutionary algorithm

dependent on the number of positive and negative examples they match in the training set and are determined prior to evaluation of the rulebase.

Basic EA strategies

As the name suggests these are the simplest ways in which an EA may be used to induce rulebases (or decision trees). A high-level description of an EA is provided in Fig. 2.3. Note that the actual genetic operators used and the implementation-level coding of the individuals (e.g. whether binary or real-valued) are dependent on the specific EA being implemented, and the particular problem being addressed.

Induction systems using Michigan-style encoding to generate fuzzy rules include *FGA* [194], *Fuzzy-ROSA* [171], and the learning systems described in [104]. In these works an individual represents one rule, has an associated fitness level, and the whole population represents the rulebase. A direct consequence is the necessity to maintain a population of different rules that can represent the complete problem domain, since it is unlikely that one rule would be able to explain the entire training set or classify all new instances.

In *Fuzzy-ROSA* this issue is resolved by dynamically changing parameter values of the EA. Various indicators of how well the search for rules is progressing are calculated for each generation and, based on these indicators, a separate fuzzy system adapts the genetic operators of the EA. The search indicators include a diversity measure for the individuals of the population based on a heuristic distance measure between two individuals, the best and mean fitness of individuals, and the simplicity or complexity of the rules being generated. For instance, if the average diversity of the population is low then the mutation rate is increased.

Another decision arising from the Michigan-style encoding scheme is whether to utilise the entire final population of the adaptive learning algorithm as the rulebase, or a subset of the population. The system *FGA*, for instance, uses only a subset of the rules, the aim being to

end up with a compact set of high-quality rules. The user defines an accuracy level and all rules that meet that level are selected. The extraction process is then carried out on this smaller population based on three criteria in descending order of importance: accuracy, coverage and fitness (the fitness measure is based on accuracy and coverage of the rule but also on its relative importance within the whole rule set as defined by a uniqueness measure). The rule or rules with the highest accuracy are identified; if there are two or more rules that have an accuracy within a predefined tolerance level then the one/s with the greatest coverage are selected; of these the one with the highest fitness is selected and placed in the final rule set. The training examples that are covered by this rule are removed from the training data set and the second rule for the final rule set is selected in the same way. This goes on until the training data set is empty.

In the second variant of the basic evolutionary strategy, an individual in the population generally represents an entire rulebase and hence only one individual need be selected from the final population as a solution. Reference [88] presents a fuzzy extension to *GABIL* [46], a crisp rule induction system for classification. Since each individual is a rulebase, the search space has consequently increased and the calculation of the fitness function is generally more computationally expensive. However, this encoding does give rise to distinct advantages: the fitness associated with each individual takes into account rule interaction and no additional procedures are required for maintaining diversity in a population.

In [103] an empirical study is conducted, comparing the two variants of this basic implementation for supervised classification problems. (It should be noted that, in general, such informative comparative studies are rare, whether making comparisons between the variants within one induction strategy, or making comparisons between different induction strategies.) A Pittsburgh-style individual represents a set of complete fuzzy rule antecedents with each individual having a predetermined number of rules, i.e. all individuals have the same number of antecedents. The rule consequent and confidence factor for each rule antecedent is determined from the training data set by a deterministic procedure. The fitness of this Pittsburgh-style individual is determined by the number of correctly classified training examples.

In the Michigan-style algorithm, each rule is represented by a separate individual so that the whole population corresponds to a single rule base. The fitness of an individual here is dependent on the numbers of correctly and incorrectly classified training examples. The authors conclude that, for the data sets tested, the Michigan-style encoding and resulting algorithm provides a greater classification accuracy at a lower computational cost, when compared with the Pittsburgh-style encoded algorithm.

Continued experiments with these two variants in later work by the same researchers enable them to make more refined observations – the algorithm using Michigan-style encoding for

individuals obtains higher classification rates for high-dimensional problems, while the algorithm using Pittsburgh-style encoding obtains higher classification rates on low-dimensional problems [100], though it should be noted that these conclusions are based on results obtained from evaluating the resulting models on the training data. This however, suggests that the Michigan-style encoded algorithm may have a greater ability to find good fuzzy rules in large search spaces. However, this algorithm can not directly optimise a rulebase as it only measures the performance of individual rules. The authors suggest that it is this indirect optimisation that leads to inferior results by the Michigan-style algorithm on low-dimensional problems.

In [99] the same authors conclude that in order to maximise performance new rules should be generated from existing good rules (as defined by the fitness function), good rules from previous generations should not die out, and optimisation of a complete rulebase should be done directly (though conclusions are based on evaluations of models on training data). This results in the design of a hybrid algorithm that attempts to combine the best features from both variants.

The individuals of this new hybrid algorithm are Pittsburgh-style and may have different lengths, i.e. different number of rules. The fitness function used in this hybrid promotes rulebases that are good at classifying the training set but also have a small number of rules. These two elements of the fitness function may be given different emphasis by attaching a weighting factor to each part. The mutation operator is, however, the main change from the original Pittsburgh-style implementation – mutation is now a single iteration of the Michigan-style algorithm and is applied to all generated rulebases after selection and crossover, i.e. the worst rules in each rulebase are now replaced with new rules that have been generated from good rules in that same rulebase.

In [100] the authors test this hybrid algorithm against their original pure Michigan-style and Pittsburgh-style algorithms. The hybrid achieves the same or better accuracy on all six data sets used, with, in most cases, a smaller number of rules thereby aiding comprehensibility of the induced knowledge. They also provide a comparison of the CPU time taken by all three variants—the Pittsburgh-style algorithm requires more time than the Michigan-style algorithm, and the hybrid algorithm is even more computationally expensive as it is essentially combining the both.

Iterative rule learning

There are two main variants to the IRL strategy – iteration by class where in each iteration rules describing a specific class are learnt, or independent of class where in each iteration good rule antecedents are first found and the class is determined afterwards. In either case, an EA is

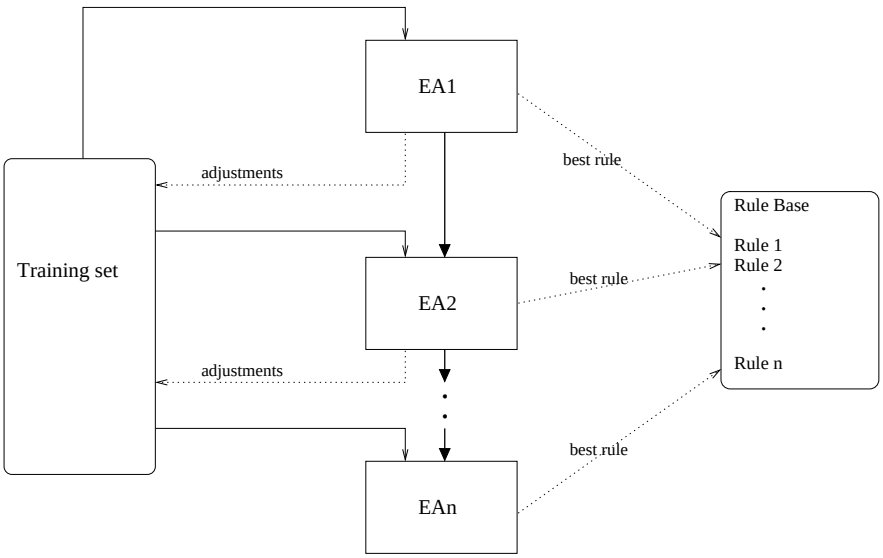


Figure 2.4: Iterative rule learning strategy

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 EAs, the training set is generally reduced by removing from it the instances that are covered by the newly evolved best rule (Fig. 2.4). This is done so as to encourage the next EA run to find good rules that describe the remaining cases in the training set.

An example fuzzy rulebase induction system following the class-dependent IRL strategy is *SLAVE* [76; 77; 80]. Considering only one class in the dataset, a GA utilising Michigan-style individuals is run and the best rule describing that class is selected and added to a final rulebase. Instances in the training set covered by this best rule are removed and another GA is run to find another rule describing the same class. GAs are run until all class instances are covered by rules in the final rulebase (or some other termination criterion is reached). This procedure is repeated for all classes in the dataset.

A common modification to the class-dependent iterative approach is to simplify the overall algorithm so that instead of running the basic EA several times for each class, it is run only once. In [163] for instance, a GA using Michigan-style encoding is run once for each class with the best individual from each run being added to the final rule set. A main aim here is to obtain a minimum number of fuzzy rules and the resulting rule base compares well in predictive accuracy with other algorithms.

Reference [91] follows the class-independent IRL strategy. In this work each individual represents both a rule antecedent and consequent (and associated membership functions), and individuals in the same population describe different classes. Reducing the training set between EA runs by removing instances covered by generated rules appears to be the most common way

of adjusting the training set. However, this work experiments with an interesting alternative.

An ES (evolution strategy) is repeatedly invoked and each time identifies the fuzzy rule that best classifies the current distribution of training examples. Each training example has an attached weight and a mechanism is employed to change the distribution of the examples from one iteration to the next – examples that have been correctly classified by fuzzy rules generated in earlier iterations have a lower weight, while those that have been misclassified have a higher weight. In each iteration the ES is guided to concentrate on generating fuzzy rules that are best adapted at dealing with the previously misclassified examples.

The resultant fuzzy rules are tested against several other techniques including neural networks and Bayesian classifiers, and on the whole achieve comparable classification accuracy. Each rule induced in this work has its own definition of associated fuzzy sets, that are evolved simultaneously with the rules, and this therefore detracts from the innate comprehensibility of the rules. The author indicates that the algorithm has recently been extended to the induction of descriptive fuzzy knowledge bases for classification, and that similar classification accuracy is achieved.

The author cites as the motivation for this new training set adjustment mechanism the requirement for cooperative, as opposed to competitive, fuzzy rules in a rulebase. However, no direct comparison is provided between the new mechanism and the more common method of removing covered training examples. It is therefore difficult to judge the exact contribution of the training set weighting method employed. Other potential mechanisms for encouraging cooperative fuzzy rules are discussed in Section 2.2.2.

2.2 Improving the Accuracy of Comprehensible Models

Recent (2003) publications of collected works on methods for improving accuracy in linguistic fuzzy models[30], and on improving interpretability in approximate fuzzy modelling [31], suggest an increasing requirement for human-comprehensible models.

Reference [31] explores different ways of improving a model's accuracy, including tuning membership functions and fuzzy rules [38], and using hierarchical knowledge bases [5]. These enhancements may improve the accuracy of a rulebase, but at some cost to its comprehensibility. This review starts from highly interpretable models and looks at ways of improving their accuracy, without sacrificing their interpretability.

The next subsection reviews literature on the use of linguistic hedges, which enrichens the knowledge representation in the expectation that it may be able to more accurately describe the underlying behaviour generating the dataset. The following subsection explores direct ways

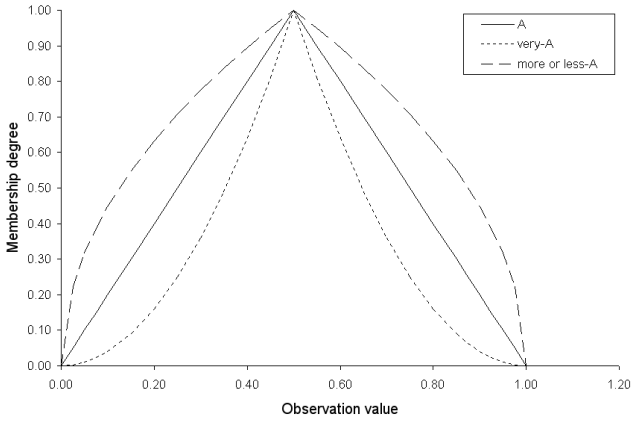


Figure 2.5: Impact of linguistic hedges on a fuzzy set A

of encouraging cooperation between fuzzy rules, such as post-processing a rulebase to remove redundant or conflicting rules, or integrating additional features within the IRL process so that rules existing in the final rulebase are taken into consideration when adding new rules.

2.2.1 Use of Linguistic Hedges

A hedge or linguistic modifier [196; 197; 198] is defined by a mathematical function and is applied to a fuzzy set in order to decrease or increase its precision. Two of the most common hedges are 'Very' and 'More or less'. As an example, the result of applying 'Very' to an original membership value is the square of the value, while that of applying 'More or less' is its square root. The impact on a simple triangular membership function A of these hedges is depicted in Fig. 2.5.

Reference [79] incorporates the use of linguistic hedges in the previously discussed *SLAVE* system [76; 77; 80] (Section 2.1.2), which follows a class-dependent IRL approach. The EA used is a GA and each individual represents one rule, encoding the variables that are present in the rule, and their values.

In this work [79], the authors add an extra component to each individual – each variable value has an associated real-valued number that denotes a hedge. New genetic operators are introduced that act on these hedges and may change them from their original value of one – 1 denotes that no hedge is acting on the variable value, 2 denotes that the hedge 'Very' acts on the variable value, and 0.5 denotes that 'More or less' is present. The real number that denotes the hedge therefore acts as an exponent to the original fuzzy set. The range of values for a hedge is however $[0,\infty)$, so that values other than 1, 2 or 0.5 render a rule less comprehensible. The authors amend their fitness function to take this into account, by providing a greater reward to

individuals that achieve the same level of fitness as rules, but with fewer modifications to the original hedge values.

The task on which this amendment to *SLAVE* is tested is one of function approximation (as opposed to classification), but provides interesting information. An additional dataset is obtained from the original by adding irrelevant variables, and tests are conducted on both. When compared with the original *SLAVE* system, the amended system produces rulebases with an average reduction in the error rate of approximately 20% and 32%, on the original and extended datasets respectively. However, the higher accuracy rulebases are also larger. The authors do not provide an explanation for this, but the use of linguistic hedges may render rules more precise. This may therefore lead to each rule covering fewer instances from the training set, which in turn means that more GAs are run in the IRL process to cover all instances in the training set.

Work reported in [29] both fine tunes membership functions and adds linguistic hedges to fuzzy rules. The experiments conducted investigate the impact of these two changes separately and combined, and so the results are informative. A GA is used for the optimisation process, and presupposes that a knowledge base (fuzzy rulebase and associated membership functions) has already been determined (either by a human expert and/or automated learning technique).

An individual in the GA encodes the membership functions for all variables used, and the IF-THEN rules. Also encoded for each condition in the IF- part of the rule, is whether there is a linguistic hedge associated with it – if there is, it may be either 'Very' or 'More or less'. In the initial population one individual represents the previously defined knowledge base, half of the remaining population is initialised with random parameter values for representing the simple triangular membership functions, and the remaining members are initialised with hedges attached randomly to rule antecedents.

The authors use predefined membership functions and induce a fuzzy rulebase via a grid partitioning rule learning algorithm ([186], discussed later on page 29). Using one dataset, they compare the results of the original rulebase and membership functions with those obtained by fine tuning membership functions only, adding linguistic hedges only, and both fine tuning functions and adding hedges. What is interesting to note is that the sole use of linguistic hedges, provides a greater reduction in error rate from the original rulebase (a reduction of approximately 87%) than soley fine tuning membership functions (approximate reduction of 84%). The authors suggest that fine tuning membership functions may be leading to over-fitting to the training data. Furthermore, the additional computational cost of fine tuning membership functions of the original rulebase is over twice as high as the additional cost of adding linguistic hedges.

Combining both tuning of functions and use of linguistic hedges results in an approximate reduction in the error rate of 98%. The authors indicate that improving accuracy even at the expense of losing some degree of interpretability is their goal, and future work will therefore explore the use of not only linguistic hedges, but more general ones (i.e. using exponents on the original fuzzy sets of not only 2 and 0.5, but of values defined over a range).

2.2.2 Direct Rulebase Optimization

Examples of methods aimed at directly encouraging the production of a complementary fuzzy rulebase have already been provided. One example is discussed in Section 2.1.2, where the authors conclude, when comparing Michigan and Pittsburgh EAs, that direct optimisation of a rulebase provides an improvement in accuracy. Another example is discussed in Section 2.1.2 on page 23, where the author develops a new mechanism for adjusting the training set in between EA runs in an IRL strategy [91]. The conclusions drawn from that work, however, are indefinite as to whether this mechanism directly leads to a more cooperative fuzzy rulebase.

This section presents further examples on how cooperation between rules may be accomplished, the first two aimed at improving the IRL strategy, and the third adopting a post-processing step after the rulebase has been induced.

Monitoring coverage of training examples by emerging rulebase

Reference [78] highlights the potential shortcomings of the IRL strategy. In this work the necessity for co-operation between fuzzy rules is stressed – since fuzzy rules cover (match) all examples to varying degrees, having a set of cooperative rules is essential to the inference process. This means that it is important to avoid, as far as possible, a situation where an instance requiring classification is matched by two or more rules that have different conclusions. The IRL approach as it is generally implemented, however, is not particularly conducive to producing cooperative rules since the rule selection process at the end of each EA run does not take into account the rules already in the final rulebase, or the degree of coverage they provide over the entire training examples.

The authors therefore implement a significant enhancement on their earlier work involving the *SLAVE* system [76; 77; 80] (introduced in Section 2.1.2 on page 23). In order to encourage co-operation between the induced rules during inference, the authors retain the same IRL strategy but do not eliminate training examples between GA runs. Instead, they attach to each example various indicators that are used in the evaluation of a rule when classifying the training examples.

The first indicator gives a measure of the maximum degree of coverage provided by rules in the current rule set having the same class as the example – the maximum positive covering degree of an example. A second indicator gives a measure of the maximum degree of coverage provided by the other rules in current final rule set, i.e. those rules having a different class from the example – the maximum negative covering degree of an example. Depending on the relative values of the maximum positive and maximum negative covering degrees of an example, and the classes of the example and rule being evaluated, the example may be considered as either a positive or a negative one for that particular rule.

The numbers of negative examples and positive examples covered by a rule are used in its evaluation as an indication of its contribution to the completeness and consistency of the current rule set. It should be noted that the values of the indicators are based on the rules already in the final set and these values are therefore modified each time a new rule is added to the final rule set – the new rule may well increase the maximum positive or negative covering degrees of certain examples, and therefore the relative values of the two indicators will also be modified. This dynamic method of keeping track of previously selected rules, and their success at classifying the training set, provides an indication of how the current rule set as a whole acts on the entire training set.

This new version of *SLAVE* is tested against the original, resulting in an improvement on generalisation capability and a significant reduction in both the number of rules in the final rule set and the execution time. In this newer version, however, the authors also amend the fitness function for evaluating individuals, adding in a term to encourage rules with fewer and less complicated conditions in the rule antecedent. However, since it is possible that fewer and more general rules (as encouraged by the new fitness function) lead to a final rule set with increased generalisation capability, it is difficult to judge the exact contribution of their new way of adjusting the training set towards the improvements reported.

Post-processing for rulebase refinement

Another approach to encouraging cooperation between fuzzy rules induced using an IRL strategy is found in [43]. This work uses a post-processing step to achieve the optimisation, as opposed to the inbuilt optimisation mechanism of the *SLAVE* system.

Reference [43] presents *MOGUL* [42], a methodology for inducing and tuning fuzzy rulebases using an IRL approach. *MOGUL* proposes three stages: the first for generating a fuzzy rulebase using IRL (with predefined membership functions), the second for refining the induced rulebase by removing redundant rules or others that do not cooperate well with other rules, and the third stage for fine tuning the initial membership functions. EAs are generally used as the rule

discovery mechanism in stage one, for optimising the rulebase in stage two, and for fine tuning the membership functions in the final stage.

In [43] only the first two stages are applied, with a GA used for the second rulebase optimisation stage. Each individual in the GA represents a fuzzy rulebase, and is encoded using a fixed-length binary string. Each bit represents one of the rules in the fuzzy rulebase generated by the first stage, so the length of the string corresponds to the size of the rulebase. If a bit in a string (individual) is equal to one, then that denotes that the rule is present in that particular rulebase (individual); if a bit is equal to zero, then that rule is considered not to form part of the rulebase. The initial population of the rulebase-optimising GA contains one individual that corresponds directly to the IRL-induced rulebase, and all other individuals are initialised randomly, so that they contain a subset of the IRL-induced rulebase. Individuals/rulebases are evaluated on two criteria: their accuracy on the training set and their coverage of it (each example in the training set must be covered by at least one of the rules to a pre-specified degree).

The authors test their refined rulebases on one classification problem and compare their results with those obtained by the non-refined rulebases, the previously discussed enhanced *SLAVE* system [78], and two other fuzzy rule induction algorithms. The dataset is fuzzified twice, so that one version of it has variables described by two linguistic values, and the second has variables described by three values. The results indicate a considerable reduction in error rates on both versions of the dataset achieved by the refined rulebases, as opposed to the non-refined ones. A reduction in the number of rules is also observed.

On the 2-label dataset, *SLAVE* and *MOGUL*-refined rulebases achieve the lowest error rates and produce the smallest rulebases. On the 3-label dataset the lowest error rate is achieved by one of the other algorithms, though its rulebases contain an average of over 300 rules. *SLAVE* and *MOGUL*-refined rulebases on the other hand, have an average size of 3 and 11 rules respectively, with error rates, however, that are 30% and 100% greater than the most accurate rulebase. On these two versions of the dataset, *MOGUL*-refined rulebases achieve lower error rates than *SLAVE* rulebases, but their average size is approximately three times the size of a corresponding *SLAVE* rulebase. The authors conclude that *SLAVE*'s and *MOGUL*'s superior performance may be attributed to the way the two algorithms encourage cooperation among fuzzy rules.

Selecting global best rule consequents

Reference [27] proposes the COR (Cooperative Rules) methodology, which aims to produce rulebases whose rules cooperate optimally during the inference process.

The authors use the space partitioning based induction algorithm in [186] as a starting point,

for generating rule antecedents whose consequents may be optimised with regards to overall cooperation between the rules. In [186] fuzzy linguistic rules are induced from examples following a multi-step process. The algorithm first obtains a fuzzy partition of the input variables (either through discourse with human experts or by a normalisation process on the dataset); generates a candidate rulebase by transforming each example in the dataset to a candidate rule (by replacing each variable value in an example with the linguistic label that best covers it); attaches to each candidate rule a measure of importance based on the example it covers; groups rules by the variable spaces they cover (i.e. rules with the same antecedent form a subgroup); and finally, for each subgroup selects the rule with the highest degree of importance.

The authors of [27] point out that each rule in the above algorithm is selected by considering only the particular variable space it covers, and no account is taken as to its potential cooperation (or lack of) with rules covering other variable spaces during inferencing. They address this point by modifying a couple of the steps in the process just described. Rules are not given a measure of importance, and after they have been sorted into subgroups according to the variable space they cover, not just one consequent is selected. Instead, for each subgroup a set is formed of the consequents that occur in the rules belonging to the subgroup.

This is therefore now a combinatorial optimisation problem, with the task being to find the best combination of consequents for the rules (one rule from each of the subgroups). This optimisation task is carried out by simulated annealing (SA) [32; 115], an algorithm that iteratively replaces the current solution with a probabilistically chosen ‘nearby’ solution. The algorithm iterates until an acceptable solution is found, or until some constraint is reached, such as a user-defined maximum number of iterations or computation time.

The authors compare this new way of obtaining consequents for the rulebase with the original method, and another fuzzy rule induction algorithm. This third algorithm is similar in that it also determines the rule antecedents that cover the different variable spaces, but instead of selecting one consequent for each space, it may select more and assigns to each a different certainty factor.

The performance of these algorithms are tested on a function approximation and on a classification problem. On the function approximation task the rulebase induced by the COR approach obtains a significantly lower error rate when compared to the original algorithm, and also performs better than the third algorithm. On the classification problem the COR rulebase achieves a lower error rate than the original algorithm, but a slightly higher error rate than the third algorithm. However, the third algorithm produces over 300 rules as opposed to the 23 rules produced by the other two, and the authors conclude that the accuracy–interpretability tradeoff achieved by their COR approach is promising.

2.3 Summary

Much research effort has been directed at inducing linguistic fuzzy models, some of it aimed at inducing fuzzy decision trees, but most at IF-THEN rules.

Decision trees have been induced using a divide-and-conquer approach, or various methods based on EC. The former approach is ultimately a greedy technique for knowledge acquisition, which bases selections of attributes on local information. Improvements in classification accuracy have been aimed at producing smaller decision trees so that over-fitting to the training data may be avoided. The population-based EC approaches may have an advantage in that several trees or attribute selections may be explored simultaneously, though the price paid is additional computation.

Descriptive production rules for pattern classification have been induced mainly via EC approaches. The rules may be evolved simultaneously using a basic EA. Unless an IRL approach is used, if an individual in the population represents one rule then diversity in the population must be enforced to ensure a reasonable selection of different rules covering different parts of the problem domain. If an individual represents an entire rule base then the search space is increased significantly, though work exists that suggests this approach produces rulebases with better classification abilities for low-dimensional problems [100]. The iterative rule learning approach avoids the necessity for enforcing diversity in a population, since during any EA run a rule must describe only a subset of the training set. However, additional measures may be required to encourage cooperation between the fuzzy rules generated.

Work aimed at improving the accuracy of linguistic models whilst maintaining their comprehensibility has also been reviewed. This involves enriching the hypothesis language so as to enable learning algorithms to more accurately describe the processes resulting in the observational datasets, or utilising techniques to encourage optimal interaction between the rules describing different classes during the classification process. Such work is very interesting and appears promising. However, the potential of this promise can not always be measured or realised, since inadequate experimental methodologies are sometimes employed. This includes occasions where more than one change is implemented (i.e. apart from the mechanism designed to improve rule interaction), so that it is difficult to judge which change(s) may have resulted in a reported difference in performance, and to what extent.

There are very few studies in the literature beyond the scope of investigating changes made to one algorithm, or comparing two or more algorithms against each other. This results in an ever-growing body of literature yielding interesting and potentially useful information, but which perhaps requires consolidation on a higher level, before it may be deemed true progress in the field of linguistic fuzzy modelling. These new consolidated levels need to focus on comparing

different strategies for fuzzy rule induction, and the impact of different modifications to one strategy. This thesis aims to partly fulfil this requirement.

The next chapter introduces ACO, the rule discovery mechanism on which the system designed, developed and implemented during this research is based, and reviews the literature using ACO for rule induction.

Chapter 3

Ant Colony Optimization and Rulebase Induction

The fuzzy rule induction system presented in this thesis, *FRANTIC*, utilises a rule construction mechanism that is based on an Ant Colony Optimization (ACO) algorithm. This chapter therefore has two aims. The first is to introduce ACO, describe its inspiration and potential benefits, and how it may be applied to general combinatorial optimisation problems.

The second aim of this chapter is to review the literature on how ACO has been applied to rule induction problems. These are still early days in the application of this technique to such problems, but the literature presents promising results and suggests opportunities for further productive exploration. Much of this potential is realised by *FRANTIC*, whose operation and performance is described in the following chapters.

3.1 The Inspiration

ACO lies within the broader field of research called Swarm Intelligence (SI), which seeks to apply animal society-inspired techniques to the solution of difficult problems. The behaviour of social insects, fish, and birds have been used to devise solutions to many such problems (e.g. [17; 52]).

As the name implies, ACO is inspired by the behaviour of ants. A fundamental concept underlying the behaviour of social insects such as ants is that of self-organisation – "a spontaneously formed higher-level pattern of structure or function that is emergent through the interaction of lower-level objects" [57]. Emergent refers to a property of a *collection* of simple lower-level subunits, that comes about through the *interactions* of the subunits. For example, the organisa-

tion of an ant colony is said to “emerge” from the interactions of the lower-level behaviours of the ants, and not from any single ant.

The other fundamental concept for SI is that of stigmergy. Self-organisation in social insects requires their interaction, either direct or indirect. Direct interaction is seen in antennation, in their exchange of food and liquid, and other physical or direct contact. Stigmergy is a form of indirect communication between individuals that is enabled by effecting changes to a common environment. Swarm intelligent systems are therefore complex systems, collections of simple units that operate in parallel and interact locally with each other and their environment to produce emergent behaviour.

The potential benefits of imitating the structures and behaviours of some animal societies in designing solutions to man-made problems include: robustness (arising from the ability of the society as a whole to survive when individuals may fail); flexibility (from the ability to adapt to changing environments); and decentralisation (removing the need to program for overall control).

Ant algorithms [47] are heuristics inspired by various behaviours of real ants that rely on stigmergy, such as cemetery organisation and brood sorting. ACO [50] is a particular instantiation of ant algorithms motivated by the foraging strategies of ants, which have been observed capable of finding the shortest path between their nest and a food source [81]. This is attributed to the chemical substance, a pheromone, that ants lay on the paths they follow to and from their nest, and the amount of which is used to guide their decision making when confronted with more than one path.

When a new food source is first located there is no pheromone to guide ants and so each will have made a random decision when presented with different paths, depositing pheromone as they travel. Several paths by different ants may therefore have been taken to reach the same food source. Since pheromone evaporates, the shortest path found by ants will accumulate the most pheromone, as ants can travel over this path more quickly and deposit more pheromone. Ants tend to select paths with more rather than less pheromone on them when presented with a choice, and therefore they eventually converge on the shortest path.

3.2 The ACO Algorithm

The first ACO algorithm was developed by Marco Dorigo as his PhD thesis in 1992, and later published under the name of *Ant System* (AS) in [49]. The application was the travelling salesman problem (TSP), where the goal is to find a closed path of minimal length connecting a given number of cities, each of which is visited once and only once.

Since the original TSP application, ACO has been utilised to solve many different problems such as binpacking, route planning, timetable scheduling, and graph colouring [50]. The general appeal of ACO and other SI techniques lies in several factors:

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

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. 3.1 on the next page. Following is a brief introduction of the main elements necessary for an implementation of an ACO algorithm [17], set in the context of rule induction. More detail is provided in the next chapter when describing *FRANTIC*. The first four elements relate to line (2) of Fig. 3.1, 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 $G = (N, A)$, called a construction graph, where N is the set of nodes composing the graph and A the set of arcs or edges connecting the nodes. A graph may or may not be fully connected and this depends on the problem being tackled and the particular approach adopted. In the context of rule induction a solution may be a rule antecedent with nodes of the graph representing individual conditions or terms, e.g. OUTLOOK=*Sunny*.
2. The *probabilistic transition rule* determines which node an ant should visit next. The choice 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 [201], for the induction of fuzzy rules.

```
(1)   WHILE termination condition false
(2)       Each ant constructs a new solution
(3)       Evaluate new solutions
(4)       Update pheromone levels
(5)   ENDWHILE
(6)   Output best solution
```

Figure 3.1: Basic ACO algorithm

4. A *solution construction validation method* forces the construction of feasible rules. For instance, if simple propositional fuzzy 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.

Increasingly, attempts are being made to research the theoretical underpinnings of ACO *per se*. Convergence proofs are available, though these are generally for highly simplified versions or for particular instantiations of the ACO heuristic [83; 84; 177]. Other researchers have proposed frameworks that seek to place ACO in context with other related learning and optimization approaches. In [13] for instance, the authors describe *Ant Programming*, a framework suggested by both ACO and optimal control theory [16]. By illustrating the similarities (and differences) with other fields such as reinforcement learning [179] and dynamic programming [11], and by utilising concepts that are clearly defined in optimal control, the authors suggest that *Ant Programming* may aid the understanding and investigation into the theoretical properties of ACO.

Other investigative frameworks include [95], which views ACO as belonging to the family of *stochastic local search* algorithms applied to combinatorial optimisation problems. These algorithms are characterised by an iterative approach that repeatedly applies small changes to solutions – with an element of randomness thrown in – in the expectation of improving their quality. They encompass nature-inspired algorithms and other techniques that utilise randomness in solution construction or modification, including EC, simulated annealing (SA) [32; 115] and tabu search (TS) [74].

Reference [200], on the other hand, places ACO in the category of *model-based* search. From this perspective, the construction graph and the artificial ants are considered as a probabilistic model that generates solutions, with the artificial pheromones considered as parameters to this

model. The ACO repeatedly uses the solutions generated in one iteration to update the parameters of the probabilistic model, which in turn generates new solutions. This model-based search category also includes the cross-entropy method [164] and estimation of distribution algorithms [136; 137] (a development of EC), but not GA, for instance. In contrast, GA belongs to the category of *instance-based* search, which generate new solutions using only the current solution/s.

The research that may lead to a proper understanding of how, why and when the application of ACO to man-made problems is successful, is still young. However, there is no denying that ACO is proving to be a useful tool in tackling difficult combinatorial problems, including those in the area of knowledge discovery and data mining. The next section describes how ACO has been applied to one such problem, specifically, to inducing crisp and fuzzy rules from empirical data.

3.3 Rule Induction via ACO

Swarm Intelligence techniques in general, and ACO in particular, are increasingly being applied to core data mining tasks [2], such as clustering (e.g. [3; 85]) and feature selection (e.g. [4; 110]). This also includes the application of ACO to rule induction. Work inducing both crisp and fuzzy rules is reviewed here, partly because the available literature is limited, but mainly because early work on crisp rule induction using ACO has influenced later work on both crisp and fuzzy induction.

This section includes some descriptions of induction strategies that have been briefly described in the previous chapter. However, they are very relevant to the work reviewed here, and so are presented in more detail for maximum clarity.

3.3.1 Crisp Rule Induction

The system called *Ant-Miner* [148] provides the first example use of ACO for constructing rule antecedents. The nodes of the ACO construction graph in *Ant-Miner* represent crisp conditions such as AGE=(20–40) or HEIGHT=(140–160) – an ant walks round the graph probabilistically selecting nodes and building its rule antecedent, ensuring during the construction process that the new partial rule antecedent covers a user-specified minimum number of instances from the training set. The rule conclusion is assigned afterwards by a deterministic method – the majority class of training instances covered by the rule antecedent is assigned. After each rule has been built a rule pruning procedure is applied, where terms are removed from the rule antecedent as long as this does not impact negatively on the quality of the rule as measured

```
(1)    WHILE termination condition false
(2)        Run ACO to generate rules
(3)        Add best rule to final ruleset
(4)        Remove instances covered by best rule
(5)    ENDWHILE
(6)    Output final ruleset
```

Figure 3.2: Class-independent iterative rule learning

by a fitness function. It should be noted that when each term is removed, the pruning process re-determines the class of the rule, so that the final pruned rule may well describe a different class from the original unpruned rule.

The system follows an IRL strategy and runs several ACOs in succession to generate a decision list made up of IF-THEN rules. Starting with a full training set, an ACO algorithm is run and the best rule created by an artificial ant is added to a final rule set. Instances in the training set that are covered by this best rule are removed before a new ACO algorithm is run. This process is re-iterated until only a few (as pre-determined by the user) instances remain in the training set, when a default rule is created to cover them. The final result is an ordered rule list with the rules being applied in the order in which they were created, when classifying a new instance in the test set (i.e. a set containing instances not used for training). Since the class of a rule antecedent being constructed by an ant is not pre-determined, this variant of IRL is here termed class-independent IRL (Fig. 3.2). Note that line (2) in Fig. 3.2 may be replaced by Fig. 3.1 on page 36.

Ant-Miner is tested on several datasets from the UCI Machine Learning Repository [7], and its performance is compared with that of *CN2* [39]. *Ant-Miner* rules are found to be comparable or superior with regards to classification accuracy on five of the six datasets used (with a statistically significant improvement in accuracy on two of the datasets), and superior with regards to ruleset and rule comprehensibility on all datasets – it generates fewer rules (a statistically significant smaller number for all datasets) that are also generally shorter than those produced by *CN2*. The authors also run *Ant-Miner* on the same datasets without the use of the rule pruning procedure. When compared to *CN2*, *Ant-Miner* rule sets are now as accurate as or better (but not significantly so) than *CN2* rule sets on only three out of the six datasets. The size of the rule sets and of individual rules is also significantly greater and closer to the size of *CN2* rules and rule sets. The rule pruning procedure therefore not only considerably improves rule and rule set size, but also leads to rule sets with a greater generalisation power.

Interest in *Ant-Miner* has resulted in various modifications to it. Reference [34] replaces the original computationally expensive rule pruning procedure with one that is less so; the new procedure produces rules that are on the whole less accurate, but are also much shorter, and

the authors conclude that a reasonable tradeoff is achieved between accuracy and comprehensibility. *Ant-Miner2* [125] uses a less computationally-expensive heuristic while maintaining the accuracy of the induced rule sets, though no results are provided that indicate the actual computational saving, and it should be noted that *Ant-Miner*'s original heuristic has a linear time complexity and the values need be calculated only once at the beginning of the algorithm. *Ant-Miner3* [126] adapts the pheromone update method and the probabilistic transition rule in order to encourage the exploration of different areas of the solution space – the result is rule sets with an improved classification accuracy, but at the cost of an increase in the number of rules.

ACO-Miner is introduced in [189]. Changes are implemented to the heuristic, pheromone updating method and the probabilistic transition rule. A few of the modifications are aimed at decreasing the computational expense of the original *Ant-Miner*, while others are aimed at controlling and balancing the tradeoff between exploration of new rules and exploitation of information learned in previous iterations. The new system is tested on several benchmark classification datasets and compared against *Ant-Miner*. *ACO-Miner* produces smaller rule sets with shorter rules at a reduced computation cost, and which have an improved classification accuracy. This, however, does come at the cost of tuning several new parameters introduced by the modified transition rule and pheromone updating method.

Reference [8] also implements several changes to *Ant-Miner*, and calls the new system *AntMiner+*. The pheromone updating strategy is based on that of the *MAX-MIN Ant System* [178], a particular ACO algorithm applied to the TSP. This includes setting lower and upper bounds for the pheromone levels, and initialising the levels at their maximum value.

Other significant changes lie in the construction graph. As for *Ant-Miner* each node represents a condition that may be added to the rule antecedent being built by an ant. Unlike *Ant-Miner*, *AntMiner+* uses a *directed* construction graph where values belonging to the same variable are grouped together and an ant is constrained to move from one group of values to another in order, though it is not specified how this order is determined. This imposed ordering on the variables may well bias the construction of rules in favour of ones that contain more values from the domains of the earlier ordered variables, than the later ones. This is because the more terms there are already present in a partial rule antecedent the less likely it will be to retain newly selected values from the later variables, since a rule antecedent with more terms is less likely to cover the user-sprecified minimum number of training instances than an antecedent with fewer terms. The authors do not comment on this potential bias introduced by their graph and construction mechanism, however, nor on its actual impact as may have been observed by inspecting the rules induced.

The *AntMiner+* construction graph also contains several additional nodes. Some are called

```
(1)   FOR each class
(2)       Reinitialise training set
(3)       WHILE class examples uncovered
(4)           Run ACO to generate rules
(5)           Add best rule to final ruleset
(6)           Remove instances covered by best rule
(7)       ENDWHILE
(8)   Output final ruleset
```

Figure 3.3: Class-dependent full iterative rule learning

dummy nodes – there is one for each variable and if the dummy node associated with a specific variable is selected by an ant, then this is interpreted as that variable having no import to the rule being built. The remaining graph nodes result out of a distinction that the authors make between nominal categorical and ordinal categorical attributes (as in *Ant-Miner*, this system cannot handle continuous attributes directly, and so their values must be discretised). Ordinal attributes may be used to create conditions in the rule antecedents made up of *two* values from their domain, and so specify an interval or range. This is achieved by creating double the number of nodes to represent the domain values of an ordinal attribute. The first set of nodes for an attribute A_k is composed of the set $\{A_k \geq V_{k1}, A_k \geq V_{k2}, \ldots, A_k \geq V_{k(m-1)}, A_k \geq V_{km}\}$ where V_{ki} corresponds to an ordinal value within the domain of size m of attribute A_k; these nodes correspond to the possible lower bounds of a range. The second set of nodes for attribute A_k is the set $\{A_k \leq V_{k1}, A_k \leq V_{k2}, \ldots, A_k \leq V_{k(m-1)}, A_k \leq V_{km}\}$ and correspond to the upper bounds of a range. When a node from each of these sets is selected and combined an interval is created that may result in more accurately descriptive rules.

This work reports marked improvement to the predictive accuracy and/or the number of rules in a rule set, when compared with *Ant-Miner* and *Ant-Miner3* on three datasets. However, further changes implemented in *AntMiner+* include those to the fitness function and the heuristic. It is therefore difficult to determine which improvement may be attributed to which of the several changes implemented.

In [174; 175] the authors make several significant changes to the original *Ant-Miner*. In both works the overall strategy followed is that of class-dependent full iterative rule learning (Fig. 3.3), where the class of the rule antecedents constructed by ants is predetermined. Among the modifications implemented, [174] changes the heuristic utilised, and prunes only the rule that is added to the final ruleset (instead of pruning every rule created during an ACO run). Reference [175] meanwhile explores rule construction without the aid of a heuristic and pruning, and uses *local* pheromone updates during construction [48] – as an ant selects a component to include in its solution, it decreases the amount of pheromone for that component in that

iteration, so that other ants in the same iteration are encouraged to create different solutions.

The changes implemented in both papers are tested on a network intrusion detection problem, and compared with results obtained from *Ant-Miner*. A suggestion is made that the improvement in predicting classes that are described by only a small number of instances in the training set is due to the class-dependent nature of the strategy. However, it is unclear how the decision list (which the authors claim is the resulting final rule set) is applied during the classification process, and which changes contribute to the improved performance.

In [172], *Ant-Miner* is greatly modified to produce a set of unordered rules. A class-dependent strategy is followed and other changes include the heuristic utilised, and the pheromone updating strategy. In order to perform classification two different rule conflict resolution strategies are tried, and the authors conclude that rule sets are produced with an accuracy level that is comparable to the decision lists produced by *Ant-Miner*, and the significant advantage obtained is in terms of comprehensibility – the rules now have an explanatory power that is independent of other rules in the ruleset.

Finally, several fairly straight-forward crisp applications of *Ant-Miner* to new problem domains may be found in [92] for web page classification, [145] for classification of texts by authors, [154] for handwritten number recognition, [168] for application to chemical engineering process monitoring problems, and in [45] for inducing rules that are used by agents in a soccer simulator.

3.3.2 Fuzzy Modelling

There is less work to report on with regards to the application of ACO to fuzzy modelling. A first attempt is found in [28] (this work also predates the introduction of *Ant-Miner*). In this work a fixed number of nodes of the ACO construction graph represent fuzzy IF-THEN *rule antecedents* (and not individual terms or conditions that make up an antecedent), which have been found previously by a deterministic method used on the training set. An ant traverses the graph, visiting each and every node and probabilistically assigns a rule conclusion (linguistic label) to each.

The ACO-based implementation is tested on a function approximation problem and on a real-world engineering task. Its performance is compared against several other fuzzy rule induction algorithms, including one based on a GA and another on SA. The results indicate that the ACO approach can achieve superior generalisation power (i.e. when measured on an independent test set), and in particular, faster convergence to optimum solutions when compared with the GA and SA. The latter benefit is attributed to the use of the heuristic in ACO, which incorporates background knowledge from the training set.

Galea *et al*. in [69] describes a system for inducing fuzzy rules that follows both *Ant-Miner*'s rule construction process, and its class-independent IRL strategy. Changes and extensions necessary for the induction of *fuzzy* rule antecedents include those to the heuristic and the fitness function. In addition, a definition of what constitutes coverage of a fuzzy instance by a fuzzy rule is required, as is a new inference process for using the fuzzy rules during the classification of new instances in the test set.

A major aim of this work, however, is to explore whether the IRL strategy may be improved upon for the induction of fuzzy rules. The authors run the system using two different ways for adjusting the training set between IRL iterations (Fig. 3.2 on page 38 line (4)) – the removal versus the re-weighting of instances covered by the best rule. The latter method is inspired by Hoffmann's work [91], which uses Evolution Strategy (ES) [162] as the rule discovery procedure within IRL. Hoffmann's system changes the distribution of the instances in the training set between ES runs by adjusting attached weights – instances that have been correctly classified by the newly found best rule are not removed outright, but instead have their associated weight reduced in value. The ES in the next iteration is encouraged to find rules that correctly classify instances with greater weights. This is in the expectation of encouraging cooperation between fuzzy rules already in the rulebase, and those that are still to be added. This paper, however, does not provide a direct comparison between this approach to training set adjustment, and the more common one of removing instances.

In [69] the authors use ACO as the rule discovery procedure and also directly compare these two different methods for encouraging the generation of complementary fuzzy rules, i.e. the aim is to avoid a situation where two rules describing different classes may both match closely an instance to be classified. Their results indicate that the re-weighting method produces rulebases with greater generalisation capabilities, but at the cost of an increase in the number of rules. However, they also note that the re-weighting method appears to increase the robustness to value changes in one parameter of the ACO (a parameter that ensures rules cover a minimum number of training instances), and suggest further work in investigating whether other parameters might also experience increased robustness. This would mean that the generalisation power of the rulebases produced are less susceptible to different parameter settings.

Two other different applications of ACO to fuzzy modeling problems are found in [26] and [146]. In the first work simple propositional fuzzy rules (i.e. rule antecedent *and* consequent), are pre-determined and act as nodes of the construction graph. Each ant then attempts to build a compact rulebase by selecting some of the fuzzy rules (nodes) and making them more general by including additional attribute values in the antecedent, for example, $\text{TEMPERATURE} = Mild$ might become $\text{TEMPERATURE} \geq Mild$. In the second work the author uses an ACO to optimise rules that have been extracted from a neuro-fuzzy network. Each condition value in the pre-

determined rules is an interval or range, and the purpose of the optimization is to adapt each interval so that the rule quality as defined by a fitness function is maximized.

As far as can be ascertained *FRANTIC*, to be presented in the next chapter, is the first system to use ACO for the construction of fuzzy rule antecedents [64]. The rule construction procedure is based on that of *Ant-Miner*'s, however, this has been extended for the induction of fuzzy rules, and developed further so that the knowledge representation of the induced rules can be easily enriched [66]. Furthermore, two very different strategies for the induction of the complete fuzzy rulebases are explored [65; 67].

3.4 Summary

This chapter has introduced ACO, which is used as the rule discovery mechanism by *FRANTIC*, the system designed, implemented and investigated during this research.

It has placed ACO within the broader field of research that is Swarm Intelligence, and outlined attempts by researchers to describe and analyse this approach to problem solving. These attempts are disparate – some at a very low-level and applicable only to specific instantiations of a particular ACO, and others aimed at viewing ACO in context with longer-established approaches, by highlighting their similarities and differences. Though the body of theory to explain how and why ACO works is still at a developmental stage, the technique has a strong appeal, which lies in emulating characteristics of real colonies such as robustness, adaptability and decentralisation, and in its broad range of application.

The application of ACO to crisp and fuzzy modelling has also been reviewed. ACO has been applied to constructing both crisp and fuzzy rule antecedents (a node in the construction graph is a term/condition), assigning conclusions to predetermined fuzzy rule antecedents (a node is a fuzzy rule antecedent), tuning predetermined fuzzy rules (a node is a fuzzy rule), and constructing a complete fuzzy rulebase from predetermined fuzzy rules (a node is a fuzzy rule). Though the volume of work in the area of fuzzy modelling is less than that for crisp modelling, it appears to be more varied in the exploration of the use of the construction graph to tackle different problems in rule and rulebase induction. One work [69] also attempts to explore the impact of the IRL strategy itself on the accuracy of the induced fuzzy rulebases.

Much of the work on the application of ACO to crisp knowledge discovery lies in modifying different elements of the ACO, such as the heuristic or pheromone update method. Some work has also explored the use of a different variant of the IRL approach: class-dependent IRL where the class of the rule antecedent being constructed is predetermined, versus the original class-independent IRL approach introduced by *Ant-Miner*, where the class label is determined after

the rule antecedent is constructed. Often, however, several changes are introduced at once in reported work, and it is therefore difficult to discern which system modification leads to which specific change in performance.

The following chapter describes *FRANTIC* in detail. It should be noted that though the ACO algorithm is a fundamental part of the system, *FRANTIC*'s strength and promise do not lie solely in the adapted ACO used to discover individual rules – this research explicitly distinguishes between the mechanism used to construct individual rules, and the strategy used to select the rules which will form the complete rulebase. Later chapters demonstrate the impact that this overall rulebase strategy may have on the quality of the induced rulebases.

Chapter 4

FRANTIC – The System

This chapter describes the rulebase induction system *FRANTIC*, which stands for Fuzzy Rules from ANT-Inspired Computation.

ACO is used as the rule construction mechanism, with the result of each ACO algorithm being a fuzzy linguistic IF-THEN rule that is added to the final rulebase. A number of ACO algorithms therefore need to be run in order to formulate a complete rulebase, and this work explores two different strategies for rulebase induction.

The first strategy follows the common iterative rule learning (IRL) strategy, where a number of ACO algorithms are run in succession, with the result of each being a rule added to an emerging final rule base. Between ACO runs, cases in the training set that are covered by the newly discovered rule are removed, so as to encourage the next ACO algorithms to find good rules describing the remaining cases. When later rules are added to the rulebase, no consideration is made as to how they may interact with rules already in the rulebase.

The second strategy, introduced in this work, follows a simultaneous rule learning (SRL) approach where the rules that form the final rulebase are constructed and evaluated together. The expectation is that the final rulebase consists of rules that complement, rather than compete with, each other. The rule construction mechanism is identical in both strategies and the major distinguishing factor between the two strategies is how rules are evaluated. In IRL each rule is evaluated individually on the training set, whilst in SRL rules describing different classes are combined and evaluated together, thereby providing some information on how well they interact during classification.

The application of ACO to rule learning is still very much in its infancy, and there are several opportunities for changing or for fully exploiting the different elements of the rule construction mechanism. Though *FRANTIC* has been designed and built with the flexibility for easy

```
(1)    FOR numIterations
(2)        FOR numAnts
(3)            Construct rule antecedent (class predefined)
(4)            Evaluate rule
(5)        ENDFOR
(6)        Determine iteration best rule
(7)        Update pheromone levels using iteration best rule
(8)    ENDFOR
(9)    Output best rule of all iterations
```

Figure 4.1: *FRANTIC* – ACO algorithm

implementation and investigation of these options, no attempt has been made to explore these features at this stage of the research; as pointed out earlier, the main focus is on gaining a high-level understanding of the current strengths and limitations of the system. However, for the interested reader various opportunities for lower-level investigations are pointed out throughout this chapter, while future work resulting from the results analysed in the following chapters, and of a more strategic nature, is outlined in Section 9.2.

The next section describes how individual rules are constructed using ACO, and the following section the two rulebase learning strategies. The final section describes the roles played by the system parameters in inducing rules from imbalanced datasets, preventing over-fitting to the training data, and in semi-automating some of the user decisions. Chapters 6 and 7 investigate in detail how and why *FRANTIC* works by analysing experiment results, and an in-depth analysis of the impact of the system parameters on the quality of induced *FRANTIC* rulebases is left for these chapters.

4.1 Rule Construction

Figure 4.1 outlines an ACO in the context of rule induction as implemented in *FRANTIC*, while Fig. 4.2 on page 49 provides more detail on the rule antecedent construction procedure. How the best rule is determined (Fig. 4.1 lines (6) and (9)) is different in the two learning strategies explored in this work (i.e. IRL and SRL), and these distinctions are made explicit when discussing the strategies. The elements of the ACO algorithm discussed in this section apply equally to both.

Each ant within each iteration of *FRANTIC*'s rule discovery mechanism constructs the antecedent of a fuzzy linguistic IF-THEN rule whose class is predefined. When creating a rule antecedent an ant traverses a construction graph where each node represents a condition or term that may be added e.g OUTLOOK=*Sunny*. Before the first term is selected by an ant and

retained as part of its antecedent, the construction graph is a fully connected one so that all terms in the construction graph are considered by an ant. However, the structure of the graph changes after each term selection – specifically, the connections between nodes – depending on the type of antecedents that are being constructed; this is discussed in more detail in Section 4.1.1 when describing the hypothesis language.

The order in which the selection of terms is made, and whether they are actually kept as part of the rule antecedent, are controlled by two factors – the probabilistic transition rule, and the two rule construction parameters called `minInstPerRule` and `constructionThreshold`.

The probabilistic transition rule determines which node an ant should visit next, i.e. which term to add to the current rule antecedent being built by an ant. This choice depends on both the heuristic value and the pheromone level associated with the node. There is an element of randomness in the choice, but it is biased towards terms that have relatively higher heuristic and pheromone values. The heuristic values remain the same throughout the run of an ACO, but the pheromone levels are adjusted from iteration to iteration to take into account the quality of the rules constructed in previous iterations, and therefore guide an ant in future iterations in making better choices.

Whether an ant keeps a selected term as part of its antecedent is determined by the rule construction parameters. Since the creation of more general rules may help prevent over-fitting to the training data, `minInstPerRule` ensures that the rule antecedent being built covers a minimum number of instances from the training set. Since all fuzzy rules or rule antecedents cover all instances, but to varying degrees, what constitutes coverage of an instance by a fuzzy rule needs defining. A fuzzy rule is said to cover a fuzzy instance if:

1. the instance is labelled with the same class as that of the rule, and
2. the degree of match between the condition parts of rule and instance (i.e. between the rule antecedent and the conditions in the instance) is equal to or greater than a pre-defined threshold value.

This threshold is set by the `constructionThreshold` parameter. This parameter consequently also indirectly controls how long or short (and therefore how specific or general) a rule is, and this is discussed further in Section 4.3.3. Details on how a degree of match between a rule and instance is determined are in Section A.4.

If the total number of instances covered by the current partial rule antecedent is equal to or greater than the value of `minInstPerRule`, then the ant retains the term (Fig. 4.2 on page 49, line (5)). Note that `minInstPerRule` in conjunction with `constructionThreshold` together provide a way of controlling the search for rule antecedents in what might otherwise be a large space. This is discussed further in Chapter 7. This process of probabilistically selecting terms and

checking the construction criteria goes on until there are no more terms that may be selected.

It should be noted that *FRANTIC*, unlike *Ant-Miner*, does not prune a rule after it has been constructed. Experiments with *Ant-Miner* clearly indicated that rule pruning not only leads to shorter rules, but more accurate ones [148]. Early experiments with *FRANTIC* indicated that though pruning might lead to somewhat shorter rules, the accuracy of the rules was adversely affected. These experiments were run on a few tiny datasets and so cannot be conclusive. However, since pruning appeared to provide some benefit (shorter rules) at considerable cost (decreased accuracy of the resulting rulebase and increased computational expenditure), all experiments discussed in this thesis were run without rule pruning.

It is important to conjecture as to the possible reason for these contradictary results on the two systems. A possible explanation is that *Ant-Miner* constructs rule antecedents prior to determining their rule consequents, whilst the opposite is true for *FRANTIC*. During *FRANTIC* rule construction all terms in an antecedent are added on the basis of how well they describe the *same* predetermined consequent – the partial rule antecedent must always cover a prespecified number of training instances of the same class. During *Ant-Miner* rule construction the partial rule antecedent must still cover a minimum number of instances, but no restriction is placed on their class label, i.e. the instances covered could belong to different classes.

It is possible that this latter form of constructing an antecedent causes some terms to be added due to a preponderence of instances belonging to a specific class, whilst others are added due to a significant number of covered instances belonging to a different class(es). In such cases pruning may be beneficial by resolving (removing) terms added due to instances with different class labels, thereby creating a more accurate rule. Investigating the potential benefits and limitations of the two different rule construction approaches (determining the rule consequent before antecedent construction, versus assigning the consequent after antecedent construction), is left for important future research, however, since the emphasis in this work is on the rulebase induction strategy.

The following subsections describe the different forms of the hypothesis language that may be used during *FRANTIC* rule construction, and how the heuristic values, pheromone levels, and the probabilities that are associated with each node, and used in term selection, are determined.

4.1.1 Hypothesis Language

FRANTIC's rule construction mechanism has the flexibility so that an increasingly expressive hypothesis language may be used to induce rules. It can create:

- simple propositional rules (e.g. *IF* TEMPERATURE *is Cool AND* WIND *is Windy THEN*

```
(1)   WHILE terms are available
(2)       Update term probabilities
(3)       Select term probabilistically
(4)       Add term to current antecedent
(5)       IF numInstCovered ≥ minInstPerRule THEN
(6)           IF ruleType = disjunction THEN
(7)               Flag term unavailable
(8)           ELSE
(9)               Flag all attribute terms unavailable
(10)          ENDIF
(11)      ELSE
(12)          Remove term from antecedent
(13)          Flag term unavailable
(14)      ENDIF
(15)  ENDWHILE
```

Figure 4.2: *FRANTIC* – rule antecedent construction by an ant

Weightlifting),

- propositional rules with internal disjunction (e.g. *IF* TEMPERATURE *is Cool* OR *Mild AND* WIND *is Windy THEN Weightlifting*),
- propositional rules that include negated terms (e.g. *IF* TEMPERATURE *is* Not-*Hot AND* WIND *is Windy THEN Weightlifting*), and
- propositional rules that include both negated terms and linguistic hedges (e.g. *IF* TEMPERATURE *is* Not-*Hot AND* WIND *is* Very-*Windy THEN Weightlifting*).

As described in an earlier chapter, linguistic hedges or modifiers are functions that modify fuzzy sets in order to decrease or increase their precision. This therefore allows a more expressive language with which to capture the underlying knowledge in data. As a first step in testing the use of hedges in improving the accuracy of *FRANTIC* rulebases (whilst maintaining the level of comprehensibility), two of the more common hedges have been utilised in this work: the hedges 'Very' and 'More or less'. These are based on the fuzzy set operators of concentration (*CON*) and dilation (*DIL*) respectively [196; 197, Part I:226, Part II:322]:

$$CON(A) = A^2 \tag{4.1}$$

$$DIL(A) = \sqrt{A} \tag{4.2}$$

where A is a fuzzy set and the *CON* and *DIL* operators cause the degrees of membership to decrease or increase respectively. The impact on a simple triangular membership function of these hedges is depicted in Fig. 2.5 on page 25.

The construction graph for constructing rules with negated terms has double the number of nodes as those used for creating simple propositional rules or rules with internal disjunction – one extra node for each original linguistic term, e.g. OUTLOOK=Not-*Sunny* as well as the original OUTLOOK=*Sunny*. If linguistic hedges are also used, then additional nodes are required, one per hedge per term, e.g. OUTLOOK=Very-*Sunny* and OUTLOOK=More_or_less-*Sunny*.

Which type of rule is created depends on the *solution construction validation method*, or solution validation method for short, that an ant is adhering to:

- Simple propositional rules, rules with negated terms, and rules with negated terms and linguistic hedges – if a term from a particular variable (attribute) is selected and meets the `minInstPerRule` and `constructionThreshold` criterion, then all other terms from the domain of the same variable are ignored while continuing to construct this rule;
- Propositional rules with internal disjunction between attribute values – irrespective of whether the construction criterion is met or not, only the selected term gets flagged so that it is not reselected later on while the ant continues building its rule.

Figure 4.2 lines (5)–(14) encapsulate how the two different solution validation methods work. Consider, for example, a rule being constructed with the capability of having internal disjunction between attribute values, and that the current selected term meets the construction criterion. If the linguistic variable OUTLOOK has values *Sunny*, *Cloudy*, *Rain*, and the term OUTLOOK=*Rain* has just been added to the rule antecedent, then it may be possible for the ant to select another value from this attribute's domain, e.g. OUTLOOK=*Cloudy*, with the final interpretation for the rule antecedent being OUTLOOK=*Rain* OR *Cloudy* – only the selected term is flagged as unavailable (line (7)). If, however, the type of rule being built is one of the other types, then all other values for that attribute are ignored by the ant during this iteration (line (9)). For example, if OUTLOOK=*Rain* has been selected and will be kept as part of the rule antecedent, then the two other terms for this attribute – OUTLOOK=*Sunny* and OUTLOOK=*Cloudy* – will not be considered for possible inclusion in the antecedent. Irrespective of the type of rule, if the selected term does not meet the construction criterion, then it is removed from the antecedent and flagged as unsuitable for reselection by this ant during this iteration (lines (11)–(13)) – all other still unselected terms belonging to the same domain are available for selection.

Note therefore, that the type of solution validation method and whether a selected term is retained or not in the antecedent impacts on how the structure of the construction graph changes during the discovery of a rule antecedent. Figure 4.3 on page 52 illustrates the possible scenarios and their impact on the graph during antecedent construction for the different forms of the hypothesis language. Each subfigure depicts a simple construction graph made up of three

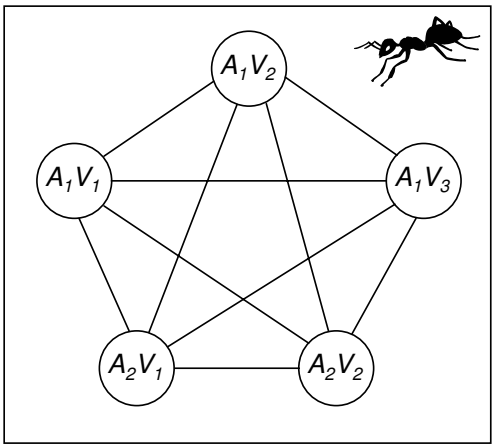
nodes representing values from the domain of one attribute (A_1V_1, A_1V_2, A_1V_3), and another two nodes representing values of another attribute (A_2V_1, A_2V_2).

Irrespective of the solution validation method, prior to the start of the antecedent construction the graph is fully connected and any term may be selected by an ant, Fig. 4.3(a). If a term is selected but not retained in the antecedent then connections to and between all other nodes in the graph are still valid; connections over which an ant may immediately travel are denoted by black arrows in Fig. 4.3(b), and the rest by gray lines. Note that as soon as an ant moves away from a selected term that was not retained, all connections to the node representing this term disappear, so that it is not reselected during this construction, Figs. 4.3(c) and 4.3(d).

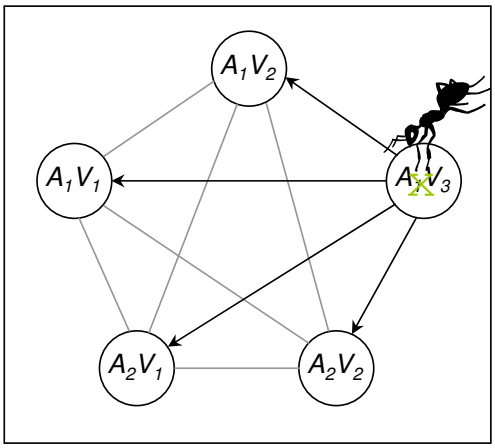
Figure 4.3(c) depicts the state of the graph after a term is selected and *retained* by an ant when creating antecedents for simple propositional rules, rules with negated terms, and rules with negated terms and linguistic hegdes – connections to nodes representing values in the same domain as the retained term disappear (connections to node A_2V_1). These connections are however maintained when creating antecedents for rules with attribute-value disjunction, Fig. 4.3(d).

This use of different solution validation methods, and/or different construction graphs, provides an extremely flexible way in which to change and/or enrich the knowledge representation language, and this has certainly not been fully exploited in this work. The types of rules utilised here have been kept fairly simple, mainly to control the degree of richness of the language and thereby be able to investigate its impact – both with regards to the quality of the induced rule-bases, and with respect to computational expense – more thoroughly. However, the language may certainly be explored further, for instance either by introducing disjunction between attributes (for e.g. OUTLOOK=*Rain* OR WIND=*Windy*), or increasing the number of linguistic hedges.

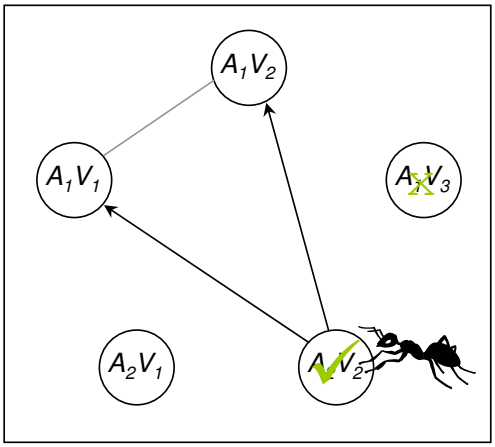
Furthermore, the current rule construction procedure predefines the operator in a condition – equality is used to define the relationship between an attribute and a value from its domain. However it is possible to consider other comparison operators such as $\geq$ or $<$, and constrain an ant to select an operator before selecting an attribute-value. The construction graph could therefore be composed of two subgraphs, one with nodes denoting different comparison operators and the other denoting attribute-values, and an ant would alternate between the two during the construction process. Investigating such other solution construction methods beyond the ones already mentioned is left for future work.



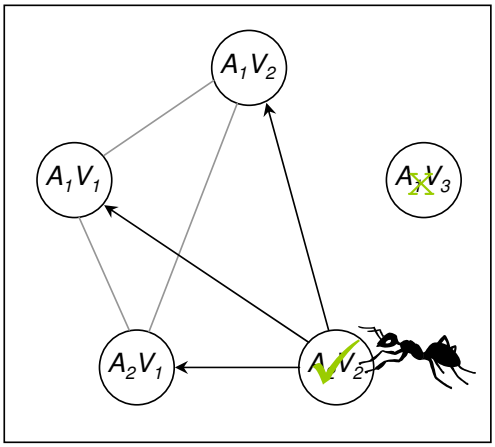
(a) All forms of hypothesis language – at start of antecedent construction



(b) All forms of hypothesis language – term selected but not retained by ant



(c) Simple propositional rules, rules with negated terms, and with negated terms and linguistic hedges – term selected and retained



(d) Rules with internal attribute-value disjunction – term selected and retained

Figure 4.3: Change in node connections of the construction graph during *FRANTIC* rule antecedent creation

4.1.2 Heuristic

The heuristic provides local information to guide ants during rule construction, as it applies to individual terms and this makes it insensitive to possible interactions between attributes. The pheromone levels (see next subsection) act as a global guide, as they are based on the fitness of a *complete* rule, and therefore take into account interaction among the attributes between the rule.

The current heuristic used by *FRANTIC* to guide ants when selecting terms uses fuzzy subsethood values [118], which give a degree to which one fuzzy set A is a subset of another fuzzy set B:

$$\begin{aligned} S(A,B) &= \frac{M(A \cap B)}{M(A)} \\ &= \frac{\sum_{u \in U} min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)} \end{aligned} \tag{4.3}$$

where u is an element in the universe of discourse U, and $\mu_A(u)$ and $\mu_B(u)$ are the degrees of membership of u in the fuzzy sets A and B respectively.

The subsethood value of a term therefore provides a measure of how important that term is in describing a specific class. The heuristic value of a term j, with respect to a class k_i, is given by:

$$\eta_{k_i}(j) = S(k_i, j) \tag{4.4}$$

where $S(k_i, j)$ is as defined by formula (4.3) above. If there are k class labels in a dataset, j will have k 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.

If fuzzy rules with negated terms are being constructed, the heuristic value for a negated term is the complement of the heuristic value for the non-negated term, i.e.:

$$\eta_{Not\_k_i(j)} = 1 - \eta_{k_i}(j) \tag{4.5}$$

whilst the heuristic values for terms with linguistic hedges are:

$$\eta_{Very\_k_i(j)} = (\eta_{k_i}(j))^2 \tag{4.6}$$

$$\eta_{More\ or\ less\_k_i(j)} = \sqrt{\eta_{k_i}(j)} \tag{4.7}$$

This heuristic was chosen primarily because it is often used in fuzzy rule-based learning, and so enables an interesting comparison with a few of these algorithms (in Chapter 8). Naturally, other heuristics are available (for instance, [201] provides a comparative analysis of 19 similarity measures that may be used), and which may be investigated after more strategic work has been accomplished (Section 9.2).

4.1.3 Pheromone Updating

The choice of where to deposit pheromones is dependent on the problem to be tackled. For instance, in the TSP the order in which the cities are selected in constructing a tour is crucial, and so in order to capture this information pheromones are deposited on the edges of the construction graph. However, rule construction using ACO may be interpreted as a *subset selection* problem, so that it is not the order of selection that is important, but the actual elements selected to form the subset [123]. For example, 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*

so that pheromones in *FRANTIC* rule construction are deposited on the nodes of the construction graph.

FRANTIC utilises the same pheromone initialisation and updating process that is used in *Ant-Miner* (to be described shortly). However, the population size in *Ant-Miner* is one, and in ant algorithms where the population size is greater (which is generally the case) a decision is required as to which ants are allowed to update the pheromone trails. All ants within an iteration may be used, or perhaps only the n best ants are used (where best is as determined by a fitness function). An elitist approach [49] also suggests that after pheromone updating has occurred at the end of an iteration with as many ants as are required, k elite ants that are the global best ants (i.e. the best k ants so far out of all iterations), are also used for updating pheromone levels. This is to ensure that exploration of the solution space is more directed. In *FRANTIC*, only the best ant of an iteration is used for updating pheromones. How this best ant is determined is the major distinguishing feature between the two rule learning strategies investigated in this work, and is discussed in Section 4.2.

At the beginning of an ACO run all nodes in the construction graph have an equal amount of pheromone, which is set to the inverse of the total number of nodes in the graph. Other ways of initialising pheromone levels are possible [50], but since this simple approach produces very reasonable results, the possibility of fine-tuning this aspect of the system is left for future work. At the end of each iteration, rules created by all ants are evaluated, and the terms that have been used in the construction of the best rule, say rule R, have their pheromone levels increased:

$$\tau_j(t+1) = \tau_j(t) + \tau_j(t) \cdot Q, \quad \forall j \in R \tag{4.8}$$

i.e. in the next iteration $t+1$, each term j in rule R will have had its pheromone level increased in proportion to the quality Q of the rule (Q is defined in Section 4.2). A normalisation of

the pheromone levels of *all* terms (with the pheromone level of each term divided by the sum of all pheromone levels), 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.

As indicated when reviewing the literature on the application of ACO to rule induction, there are various ways in which the pheromone updating strategy may be changed. These include a different pheromone initialisation, and the setting of minimum and/or maximum bounds on the levels reached. For instance, putting an upper bound on the maximum value of a pheromone level avoids some terms being reinforced to the point where it is impossible to create a solution without them, and so favours exploration [178].

4.1.4 Transition Rule

FRANTIC 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:

$$P_j^m(t) = \frac{[\eta_j]^\alpha \cdot [\tau_j(t)]^\beta}{\sum_{i \in I_m} [\eta_i]^\alpha \cdot [\tau_i(t)]^\beta} \tag{4.9}$$

α and β in the above formula are two adjustable parameters that control the relative influence of the heuristic and pheromone values. I_m is the set of terms that may still be considered for inclusion in the rule antecedent being built by ant m, and is therefore dependent on the solution validation method.

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, rules with negated terms, or rules with both negated terms and hedges are being created, then I_m will further exclude 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. The transition rule used in this work is based on that of the *Ant System* [49], designed to solve the TSP. It is called a random proportional

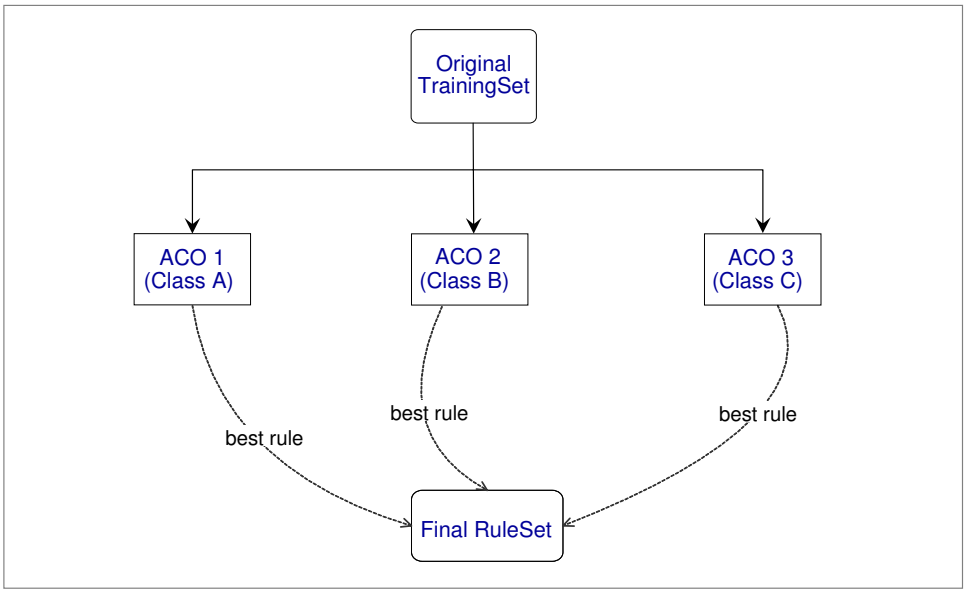
transition rule, and is the most commonly used one as it produces reasonable results in most cases.

A less often used transition rule is one called pseudo-random proportional rule, first used by a family of ant algorithms called *Ant-Q* [70]. These algorithms were used with different transition rules and also applied to the TSP. Algorithms applying the pseudo-random proportional rule were found to outperform others with different rules, including the random proportional one. How an ant chooses a term is now dependent on a random variable q uniformly distributed over $[0,1]$, and an adjustable parameter q_0 with range $[0,1]$. For each selection, if the value drawn from q is less than or equal to q_0, then the term with the highest proportion of heuristic and pheromone values is deterministically selected. Otherwise, the selection is carried out probabilisticlly according to formula (4.9) above. This transition rule is aimed at exploiting some of the information learnt in previous iterations, and may be useful since Galea [63] discovered that for crisp rule induction, rulebases were induced with comparable classification accuracies as for the random proportional transition rule, but at a lower run time cost. However, this rule does introduce a new parameter that must be tuned for each dataset, and so it is not investigated in this current research.

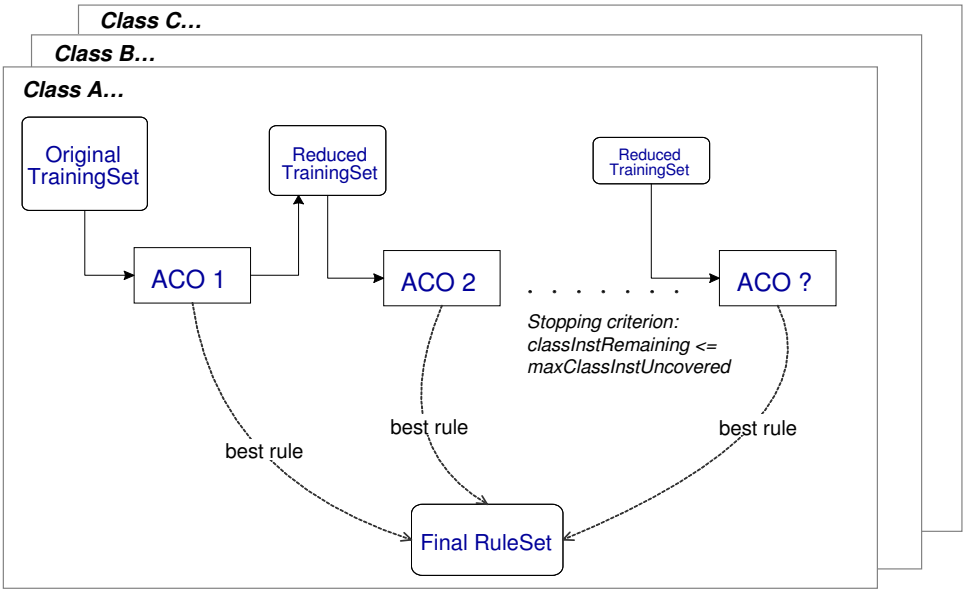
In the ACO literature, controlling the balance between prior information and newly learned information is often accomplished through the use of the exponents α and β in formula (4.9). This however, generally requires an empirical approach to determine their values for each different problem (or dataset, in this case). Since the purpose of the research at this stage is to gain a general appreciation of the system, α and β have been given equal importance and kept equal to one, over all datasets – note therefore, that this is also a more stringent test of the system's capabilities, since the results discussed in the following chapters have not been obtained through tuning of parameters that are considered to be problem-specific.

4.2 Rulebase Induction Strategies

This section describes the two major strategies for rulebase induction that are investigated in this work, and highlights their differences. Section 4.2.1 describes the iterative rule learning approach implemented, with its two variations – simplified iterative rule learning and full iterative rule learning, and Section 4.2.2 describes the alternative approach introduced, that of simultaneous rule learning and evaluation.



(a) Simplified iterative rule learning



(b) Full iterative rule learning

Figure 4.4: *FRANTIC* iterative rule learning

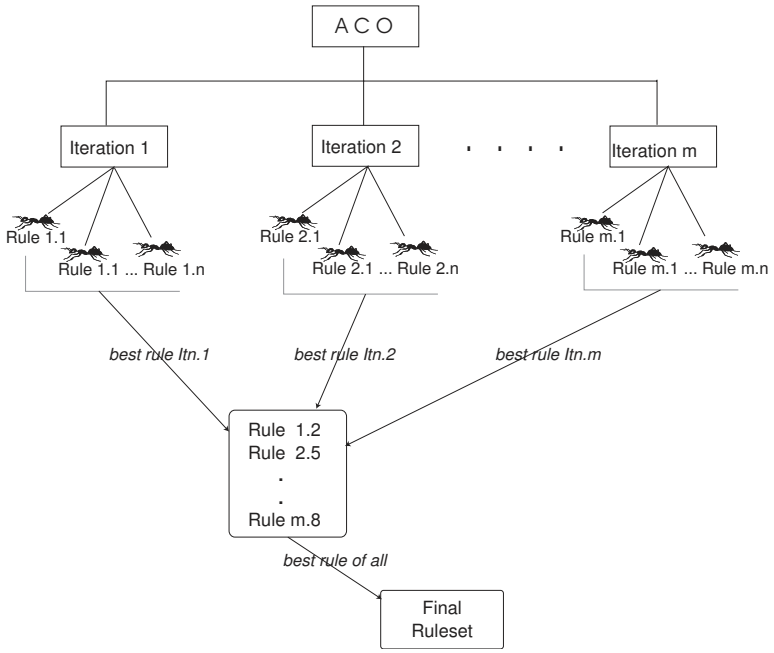


Figure 4.5: *FRANTIC* – ACO algorithm for iterative rule learning

4.2.1 Iterative Rule Learning

As already mentioned, IRL is an approach often employed when using stochastic population-based algorithms for rule discovery [68]. The specific IRL approach investigated in this work is the class-dependent one, where the class of the rules being constructed has been pre-defined. *FRANTIC* can be run in 'simplified' iterative rule learning mode, where only one rule is created to describe each class in the dataset, a common restriction in several rule induction algorithms (Fig. 4.4(a) on the preceding page), or in 'full' iterative rule learning mode (Fig. 4.4(b)), where user-specified or dynamic system parameters are used to determine how many rules are necessary to adequately describe each class in the dataset.

How each individual ACO in Fig. 4.4 on the previous page works is depicted in more detail in Fig. 4.5. The result of each ACO is the best fuzzy rule as determined by some quality function, and is added to the final rulebase. In full IRL, for each class several ACO algorithms are run in succession to find several rules for that class that are added to the final rulebase. In between ACO runs, the training set is reduced by removing from it the instances that are covered by the newly created best rule. This is done so as to encourage the next algorithm to find good rules that describe the remaining instances in the training set.

Note that in each ACO algorithm run (Fig. 4.5), the rule added to the final rulebase is not necessarily the best rule of the final iteration, but is the best rule produced from all iterations. This point is important in later discussions in Sections 6.2 and 7.8.

```
(1)   FOR each class
(2)       Reinstate full training set
(3)       WHILE classInstRemaining ≥ maxClassInstUncovered
(4)           Run ACO
(5)           Determine best rule R of ACO
(6)           FOR each instance u
(7)               IF class of R = class of u
(8)                   IF degree of match (R,u) ≥ removalThreshold
(9)                       Remove u from training set
(10)                  ENDIF
(11)              ENDIF
(12)          ENDFOR
(13)          Add best rule to finalRuleSet
(14)      ENDWHILE
(15)  ENDFOR
(16)  Output finalRuleSet
```

Figure 4.6: *FRANTIC* pseudocode for class-dependent full iterative rule learning

The next subsection describes the parameters that control the termination of *FRANTIC* running in full IRL mode, and the following subsection explains rule evaluation, which is identical in both simplified and full IRL.

FRANTIC-fullIRL stopping criteria

The two user-defined parameters `removalThreshold` and `maxClassInstUncovered` determine when *FRANTIC* stops adding rules that describe the same class to the final rulebase.

Figure 4.6 provides an outline of *FRANTIC* running in full IRL mode. Note that lines (4)–(5) may be replaced by Fig. 4.1, which outlines how an ACO works. In between runs of ACO algorithms to find rules describing the same class, instances that are covered by the best rule, and that have the same class label, are removed from the training set (Fig. 4.6 lines (6)–(12)). As for rule construction, a threshold is necessary to define coverage, and if the degree of match between the rule and an instance with the same class label meets or surpasses the set threshold, then the instance is deemed to be covered by that rule and is removed. This threshold is set by the parameter `removalThreshold`. The lower the value for this threshold, the more class instances that are removed from the training set in between ACO algorithms.

When the number of instances of the current class remaining in the training set is equal to or below the value stipulated by `maxClassInstUncovered` (Fig. 4.6 line (3)), then *FRANTIC* stops running ACO algorithms to find rules to describe that class, and starts running ACO algorithms to find rules for the next class (line (1)).

Note that these two parameters, together with `minInstPerRule`, also indirectly determine the number of rules in the final rulebase, and this is discussed in detail in Section 7.1.2 when analysing experiment results. Furthermore, Section 4.3 provides more details on all system parameters, and Section 5.5 describes the approach adopted for setting parameter values.

Rule evaluation

In *FRANTIC* IRL each rule is evaluated separately, without taking into account how it may interact with other rules describing the other classes. The fitness function used is a common one in rule induction and is the same as the one used in *Ant-Miner*; it evaluates an individual rule on the basis of how accurately it classifies all instances in the training set. It combines a measure of the sensitivity of a rule (its accuracy among instances of the same class as the rule) with a measure of the specificity of the rule (its accuracy among instances of different classes):

$$\begin{aligned} Q &= sensitivity \cdot specificity \\ &= \frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP} \end{aligned} \tag{4.10}$$

where

- TP (True Positives) is the number of instances that have the same class label as the rule and satisfy the `fitnessThreshold` parameter ;
- FP (False Positives) is the number of instances that have a different class label from the rule and satisfy the `fitnessThreshold` parameter;
- FN (False Negatives) is the number of instances that have the same class label as the rule and do not satisfy the `fitnessThreshold` parameter;
- TN (True Negatives) is the number of instances that do have a different class label from the rule and do not satisfy the `fitnessThreshold` parameter.

The determination of TP, FP, FN and TN is illustrated in Fig. 4.7. As for rule antecedent construction where `constructionThreshold` is required to define sufficient coverage, and for removal of training instances between ACO algorithm runs during full IRL where the parameter `removalThreshold` is used, another threshold is employed during rule evaluation to define coverage. The potential flexibility offered by keeping the thresholds separate and set at different values is not explored in this thesis, as the results discussed in the following chapters are obtained by keeping two of the parameters (`fitnessThreshold` and `removalThreshold`) at the

```
(1)    SET TP, FP, TN, FN to 0
(2)    FOR each instance u in training set
(3)        Determine degreeOfMatch between R and u
(4)        IF degreeOfMatch ≥ fitnessThreshold THEN
(5)            IF R and u have same class label THEN
(6)                TP := TP + 1
(7)            ELSE
(8)                FP := FP + 1
(9)            ENDIF
(10)       ELSE
(11)           IF R and u have same class label THEN
(12)               FN := FN + 1
(13)           ELSE
(14)               TN := TN + 1
(15)           ENDIF
(16)       ENDIF
(17)   ENDFOR
```

Figure 4.7: *FRANTIC* IRL rule evaluation – determining basic statistics for rule R

same value. However, these three thresholds are discussed in more detail in the next section, and indications for future work to explore their potential are provided.

4.2.2 Simultaneous Rule Learning

Simultaneous rule learning is inspired by ant super-colonies, i.e. colonies that have many nests spread over a large geographical area, and consisting of millions of workers and many queens. Two super-colonies of the Argentine ant *Linepitherma humile*, for instance, exist in southern Europe, with the largest of them following the Mediterranean and Atlantic coastline for over 6000 km [73].

Individual nests within super-colonies have local pheromone trails leading to local food sources, but there are also permanent trails and tunnels linking the different nests. Normally, ants from different nests would attack each other, even if of the same species, but what distinguishes the super-colonies is that workers from the different nests are allowed to move freely along the permanent network of trails. Furthermore, queens, larvae and workers are exchanged freely between the nest sites in response to environmental changes and consequent requirements.

Multi-nest colonies provide a template for simultaneous rule learning, with real ants of a particular nest finding the best path to their local food source, construed as artificial ants finding the best description for a particular class. *FRANTIC* SRL therefore runs several ACO algorithms in parallel, with each finding rules to describe a specific class, and maintaining its own con-

```
(1)    FOR numIterations
(2)        FOR each class
(3)            FOR each ant
(4)                Construct rule antecedent
(5)            ENDFOR
(6)        ENDFOR
(7)        FOR each combined rulebase
(8)            Evaluate rulebase
(9)        ENDFOR
(10)       Determine iteration best rulebase
(11)       Update pheromones using rules from best rulebase
(12)   ENDFOR
(13)   Output best rulebase of final iteration
```

Figure 4.8: *FRANTIC* simultaneous rule learning

struction graph, artificial 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. It is also currently a 'simplified' SRL implementation, with one rule included in the final rulebase to represent a class.

An overview of the system is provided in Fig. 4.8. After each class has had its rules created for a particular iteration (Fig.4.8, lines (2)–(4)), rulebases are created which contain one rule from each class, and these rulebases are tested on the training set (lines (5)–(7)). The rules in the best performing rulebase are used to update the pheromone levels (line (9)) – each rule is used to update the pheromone levels of the ACO that produced it (since each ACO generates rules that describe a specific class). The following subsection details the rulebase evaluation process.

Rulebase 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 independently on the training set, as in IRL, each rule is combined with rules describing the other classes in the dataset, and they are evaluated simultaneously.

During an iteration, when all rules have been created for each of the classes (Fig. 4.8 line (4)), complete rulebases are formed composed of one rule from each of the different classes. If *numClasses* is the number of classes in the dataset, and *numAnts* is the number of ants creating rules for each class, then the number of possible rulebases is:

$$numAnts^{numClasses} \tag{4.11}$$

Each rulebase is now evaluated by determining how well it classifies the training set (Fig. 4.8 lines (5)–(7)). Note that this way of evaluating a rulebase takes into account how rules describing different classes interact, and exactly mirrors how the rulebase is to be used in practice (unlike the evaluation method used for IRL). This evaluation method may therefore provide useful guidance to the pheromone updating process, and consequently to the process of creating future rules that must interact optimally during classification.

The method of classification used is the single winner-based method described briefly in Section A.3. 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 classification accuracy obtained by a rulebase on the training set is used as a measure of the quality, *Q*, of each rule within the rulebase:

$$Q = \textit{proportion of correctly classified instances} \tag{4.12}$$

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:

$$numIterations * numAnts^{numClasses} \tag{4.13}$$

where *numIterations* is the number of iterations run in an ACO. It is quite possible, however, that this number may be drastically reduced without its impacting on the quality of the final rulebase, and this is explored in detail in Section 7.3.1.

4.3 System Parameters and Control

Table 4.1 on the following page lists *FRANTIC* parameters that require user-setting. The first four parameters are necessary in whichever mode *FRANTIC* is operated, i.e. simplified and full IRL, and SRL. The descriptions for `numIterations` and `numAnts` are fairly straight-forward, indicating the number of iterations for an ACO, and the number of ants within each iteration. As already described, `minInstPerRule` and `maxClassInstUncovered` together determine whether a term selected by an ant is retained in its antecedent.

FRANTIC simplified IRL requires one extra parameter to *FRANTIC* SRL – `fitnessThreshold`. This parameter is necessary for the rule evaluation process in IRL (simplified and full mode), in

Table 4.1: *FRANTIC* parameters

Name	Description
`numIterations`	Number of iterations per ACO algorithm.
`numAnts`	Number of ants constructing a solution within an iteration of an ACO algorithm.
`minInstPerRule`	Used during rule construction – minimum number of instances in the training set that a rule must cover.
`constructionThreshold`	Used during rule construction – sets the value for the threshold above which a rule is considered to cover an instance in the training set.
`fitnessThreshold`	Applicable only for simplified and full IRL. Used during rule evaluation – sets the value for the fitness level threshold above which a rule is considered to cover an instance in the training set.
`removalThreshold`	Applicable only for full IRL mode. Used during removal of class instances from the training set between ACO runs – sets the value for the instance removal threshold above which a rule is considered to cover an instance in the training set.
`maxClassInstUncovered`	Applicable only for full IRL mode. Maximum number of class training instances that may be left uncovered by a rule.

order to define the degree to which a rule must cover a training instance when determining the number of TPs, FPs, TNs and FNs. Since rulebase evaluation in *FRANTIC* SRL uses a 'winner takes all' approach to classifying the training instances, where the rule with the highest degree of match classifies the training instance, there is no need for a threshold.

FRANTIC running in full IRL mode requires an additional two parameters over the simplified versions (simplified IRL and SRL) – `removalThreshold` and `maxclassInstUncovered`. Since in this mode more than one rule may be created to describe a class, these parameters help to determine when *FRANTIC* should stop adding rules to the final rulebase that describe the same class, and move to finding rules to describe a different class. Possible ways of extending the current SRL mode which constructs one rule per class are discussed in Section 9.2.3.

This subsection describes how *FRANTIC* parameters can help induction from datasets with an uneven class distribution, and how they prevent over-fitting to the training data. It also highlights how the semi-autonomous nature of several of the parameters can help prevent unnecessary user intervention, and how this may be exploited further with regards to the three fuzzy threshold parameters – `constructionThreshold`, `removalThreshold` and `fitnessThreshold` – discussed in previous sections. Note that Section 5.5 describes an approach for how parameter values may be determined when running *FRANTIC* on different datasets – it is shown that most system parameters have default settings that produce very reasonable results with regards to rulebase size and accuracy.

4.3.1 Initialisation of `minInstPerRule` and `maxClassInstUncovered`

These two parameters have been designed so that different values may be set for different classes, which is particularly useful when inducing rules from datasets that have an uneven class distribution (imbalanced datasets). For instance, in the Water Treatment dataset used in experiments in the following chapters, there are two classes, one containing 371 instances (called the *Normal* class) and the other with only 6 instances (the *Faulty* class). If `minInstPerRule` is constrained to be the same value for both classes, then the maximum value is upper bound by the size of the minority class, i.e. 6. This means that rules created need cover only 6 instances labelled with the appropriate class from the training set. Experimentation with an early prototype of *FRANTIC* suggested that this results in inducing rules that are too specialised for the *Normal* class, i.e. match too few instances in the training set and therefore over-fit it, and also make it difficult or impossible to generate a rule that matches all or nearly all instances of the *Faulty* class.

Similarly for `maxClassInstUncovered`. If this parameter is set to 0, then for each class as many ACOs as necessary are run so that all class instances from the training set are covered

by a rule. In theory, and confirmed often in practice by experiment results, this results in one or both of two possibilities: many rules in the final rulebase, and over-fitting to the training data. If `maxClassInstUncovered` is set to 2, for instance, then for the minority class in the Water Treatment dataset this might prevent over-fitting, but it would have little or no beneficial impact on the majority class.

The setting for these two parameters may therefore be stipulated in various ways, including an absolute value that is applicable to all classes, or a proportion that is also applicable to all classes but where the absolute per class is calculated based on the number of instances labelled with that class. For example, if `minInstPerRule=5` then a rule describing *Faulty* will cover at least 5 *Faulty* instances from the training set, and a rule describing *Normal* will cover at least 5 *Normal* instances. If, however, `minInstPerRule=80%` then a rule for *Faulty* will again cover at least 5 instances (rounded from 4.8), but a rule for *Normal* will cover at least 300 instances (rounded). These parameters have further potential, and this is discussed in Section 9.2.

4.3.2 Re-Setting of `minInstPerRule` in Full IRL

While operating in full IRL mode, ACOs are run for one class until the number of class instances remaining in the dataset, `classInstRemaining`, is no greater than that specified by `maxClassInstUncovered` (Fig. 4.6 on page 59, line (3)). This is done to both restrict the number of rules, and reduce over-fitting to the training data. However, in order to ensure that the purpose for which `maxClassInstUncovered` is implemented is actually achieved, it is sometimes necessary to change the value of `minInstPerRule` in between ACO runs. A specific example will serve to illustrate the rationale behind this.

Assume, for instance, that before the first ACO is run to find rules describing the *Normal* class for the Water Treatment dataset, the following parameters have the values: `classInstRemaining=371` (the original number of class instances in the training set), `minInstPerRule=186` (50% of original number of class instances), and `maxClassInstUncovered=37` (10% of original number). The best rule from this first ACO is determined and those class instances covered by this best rule are removed from the training set. If the number of instances removed is 190, say, then at the start of the second ACO `classInstRemaining=181`, which is less than the required `minInstPerRule=186`. *FRANTIC* therefore automatically resets the value of `minInstPerRule` taking into account both `classInstRemaining` *and* `maxClassInstUncovered`:

```
IF classInstRemaining≤(minInstPerRule+maxClassInstUncovered) THEN
                        minInstPerRule=(classInstRemaining−maxClassInstUncovered)
```

resulting, for this particular example, in `minInstPerRule=181-37=144` at the start of the second

ACO. This means that rules constructed are as close as possible to the original relative value of minInstPerRule, whilst ensuring that the purpose of maxClassInstUncovered is not compromised. Experiments run on an early *FRANTIC* prototype indicated that this produces better results, with regards to classification accuracy, than only re-setting minInstPerRule=classInstRemaining, which renders the use of maxClassInstUncovered pointless.

4.3.3 Parameter Adjustment During Rule Construction

Both minInstPerRule and constructionThreshold have been implemented so that their values may change automatically, if necessary, during a *FRANTIC* application run.

It is a simple matter to ensure that an initial setting for minInstPerRule is a valid one, i.e. that there really is at least that number of class instances in the training set. This can be achieved either by conducting a cursory analysis of the dataset in order to determine the class distribution (if an absolute value is used for this parameter), or by stipulating instead a proportion of class instances and letting *FRANTIC* determine the absolute values for all classes. The preceding subsection 4.3.2 also explained how the value of this parameter is re-set and remains valid if *full* IRL mode is used, i.e. after instances covered by a best rule have been removed from the training set, and before another ACO is run to discover another best rule that describes the same class.

The purpose of constructionThreshold is to set the level at which a rule is considered to cover a fuzzy instance – if the degree of match between the two meets or surpasses this threshold, then the instance is considered sufficiently matched by the rule. The lower the value of constructionThreshold, the easier it will be for an instance to be covered by the rule. The easier it is, the greater the number of instances that will be considered covered by the rule, and hence it is more likely that the criterion for keeping a newly selected term in the current rule antecedent being built is met. As a direct consequence of this, lower values of constructionThreshold also tend to generate rules with a greater number of terms in the rule antecedent than higher values.

It is quite possible, though not common, that the ‘best’ rule produced by an ACO is one with an empty antecedent. This happens if constructionThreshold is set so high that no ant is able to construct a rule that covers the required number of class instances to the specified degree of match. This rule is not added to the final rulebase. Instead, *FRANTIC* will reduce the value of minInstPerRule, and run another ACO so that a new ‘best’ rule is generated. Note that the value of constructionThreshold is kept the same, i.e. the system favours the construction of rules that may cover fewer instances but match them to a higher degree. Reducing minInstPerRule instead of constructionThreshold is a reasonable decision to make, since this problem

```
(1)   FOR each class
(2)     Reinstate full training set
(3)     WHILE classInstRemaining ≥ maxClassInstUncovered
(4)       IF classInstRemaining ≤ minInstPerRule + maxClassInstUncovered THEN
(5)           minInstPerRule = classInstRemaining − maxClassInstUncovered
(6)       ENDIF
(7)       validRule := false
(8)       WHILE validRule = false
(9)         Run ACO
(10)        Determine best rule
(11)        IF bestRule of ACO is valid THEN
(12)          Remove instances covered by bestRule
(13)          Add bestRule to finalRuleSet
(14)          validRule := true
(15)        ELSE
(16)          IF minInstPerRule > 1 THEN
(17)            minInstPerRule := minInstPerRule − 1
(18)          ELSE
(19)            constructionThreshold := constructionThreshold − 0.05
(20)          ENDIF
(21)        ENDIF
(22)      ENDWHILE
(23)    ENDWHILE
(24)  ENDFOR
(25)  Output finalRuleSet
```

Figure 4.9: *FRANTIC* – adjustment of `minInstPerRule` and `constructionThreshold`

is more likely to happen when *FRANTIC* is running in full IRL mode, rather than in simplified mode, i.e. when the number of class instances have been reduced one or more times in between ACO runs.

If the extra ACO also fails to produce a valid rule, i.e. a rule that has at least one term in the rule antecedent and satisfies the current `minInstPerRule` and `constructionThreshold` requirements, then *FRANTIC* will continue to reduce `minInstPerRule` and run additional ACOs until either a valid rule is created or `minInstPerRule` can not be reduced further (i.e. is already equal to 1). In this case, `constructionThreshold` is reduced gradually until a valid rule is produced – with class instances still in the training set, then at some lower value of `constructionThreshold` a valid rule *will* be produced that covers at least one of the remaining class instances.

Figure 4.9 provides a more detailed *FRANTIC* overview than Fig. 4.6, and highlights the details relating to the automatic adjustment of `minInstPerRule` and `constructionThreshold` as discussed in this and the preceding subsection. Lines (4)–(6) are relevant only if *FRANTIC* is running in full IRL mode, whilst lines (11)–(21) are valid for both IRL modes, and for SRL

operation.

In the experiments presented in this work the values for both parameter adjustments are as in Fig. 4.9. However, there may be circumstances where it might be necessary and/or desirable to reduce both parameters by values other than those used here. For instance, with a large training set and `minInstPerRule` originally set to 50%, say, then reducing the value by '1' may still require several ACO runs before a valid rule is created; a more reasonable value by which to reduce may be '10', or '10%' of the original `minInstPerRule` value, for example.

There are other ways the self-adjustment of `minInstPerRule` and `constructionThreshold` may be implemented. This includes gradually decreasing the value of the two parameters in tandem (instead of first reducing `minInstPerRule` to '1', before adjusting `constructionThreshold`), and setting a minimum threshold for `constructionThreshold` (in which case if no valid rule has been found for this class when this threshold is reached, then *FRANTIC* would move on to finding rules to describe the next class). These alternative methods have potential advantages of their own. The first alternative would ensure that rules are produced that match *both* original parameter settings as far as possible, whilst the second alternative would prevent the addition of a very specialised rule to the final rulebase. Of course, a third alternative would be combining the first two.

However, the simple adjustments implemented at this stage appear to be effective. They produce very reasonable rulebases with regards to accuracy, the number of rules in a rulebase, and the number of terms per rule (as evidenced by the experiment results presented in Chapters 6–8). Investigating more sophisticated ways for adjusting these parameters is therefore left for future work.

4.3.4 The Fuzzy Thresholds

As has been described, the concept of fuzzy rule matching is utilised in *FRANTIC* several times, and in each case a different parameter to define the fuzzy threshold has been implemented – `constructionThreshold` is used during rule construction, `fitnessThreshold` during rule evaluation, and `removalThreshold` for determining which instances are removed from the training set between ACO runs in full IRL mode.

These thresholds were implemented separately for maximum future flexibility. Little use has as yet therefore been made of this flexibility in the current work, and the experiment results discussed in Chapters 6 and 7 are obtained by keeping `fitnessThreshold` and `removalThreshold` fixed to the same value (over all datasets), and varying only `constructionThreshold`.

However, results from a preliminary investigation on these fuzzy thresholds do suggest that

on some datasets setting fitnessThreshold and removalThreshold lower than constructionThreshold produces rulebases with greater classification accuracy. Reducing the value of removalThreshold, for instance, may result in another mechanism for preventing over-fitting to the training data – the lower the value, the greater the number of instances removed between ACO runs. This results in a similar effect to that provided by maxClassInstUncovered, where its purpose is to ensure that not *all* instances in the training set need be covered by a rule in the final rulebase, thereby helping to prevent over-fitting. A more thorough investigation is therefore required to fully understand the dynamics between the three parameters, and before any conclusions may be drawn.

If a rigorous experimental analysis reveals that no real benefit is obtained by having fitnessThreshold and removalThreshold, however, or that the same results may be obtained by tuning other parameters such as minInstPerRule in conjunction with constructionThreshold, then it will be possible to remove one or both parameters. On the other hand, if results strongly indicate that either of these parameters should be set lower than constructionThreshold in order to achieve maximum benefit, then *FRANTIC* may be modified so that it automatically sets these parameters itself, based on the user-defined constructionThreshold. In either of these cases, it would therefore be possible to further prevent unnecessary user intervention.

4.4 Summary

This chapter has described *FRANTIC*, a system for investigating the indcution of fuzzy linguistic rules that utilises the rule construction mechanism – an ACO algorithm – in two very different learning strategies.

The first strategy is the popular (class-dependent) iterative rule learning approach. Some algorithms explored in the literature (using GAs or other population-based algorithms as the rule discovery mechanism) adopt a simplified form of this strategy where only one rule is generated to describe each class in the training data. Other algorithms in the literature follow a fuller mode where several rules per class may be utilised in the final rulebase. *FRANTIC* may be run in either of these iterative rule learning modes.

The second strategy, introduced in this work, is that of simultaneous rule learning. It uses exactly the same rule construction mechanism as for iterative rule learning, but the rule evaluation method at the end of each ACO iteration, and consequently, how a ‘best’ rule is selected to perform pheromone updates in between iterations is distinctly different. In this approach, the rulebase evaluation inherently takes into account how well the rules describing the different classes interact during classification.

The simultaneous rule learning strategy utilises the least number of system parameters (four), whilst simplified iterative rule learning requires the setting of five parameters, and full iterative rule learning the setting of seven parameters. Two of the most important parameters – the rule construction parameters `minInstPerRule` and `constructionThreshold`, common to all strategies/modes – implement simple effective mechanisms to manage imbalanced datasets, and prevent over-fitting to the training data (as is evidenced by results in the following chapters).

These two parameters have also been designed to self-adjust during system operation, which removes the need from the user to analyse the dataset and identify valid possible combinations for the two parameters *before FRANTIC* is run, or to intervene *during* its operation if all ants in an ACO fail to construct a valid rule due to changes in the training environment. Designing these and other parameters to self-adjust is highly desirable and useful, and it should be noted that this has not yet been fully exploited (since the priority of this investigation is on developing, and gaining a real understanding of, *FRANTIC*'s potential, and not on fine-tuning small parts that provide little insight into how and why the system works).

FRANTIC is built on relatively new and/or unexplored concepts, and an explicit distinction has been made between the mechanism for constructing individual rules, and the strategy for selecting rules to form the complete rulebase. The rule discovery mechanism incrementally constructs rules term by term; the potential of this ACO-based constructionist approach to fuzzy rule discovery is rigorously investigated in Chapter 6. Analyses of the experimental results suggest exciting possibilities for even further exploitation of a very promising rule construction procedure, and these are outlined in Section 9.2.

The potential advantages of simultaneous induction of the fuzzy rules that comprise the final rulebase have not been explicitly and/or thoroughly explored in the literature. Chapter 7 therefore investigates the impact of such an approach – when compared to iterative rule learning – on the operation of the system (for instance, as observed through increased robustness to parameter values), and the quality of the induced rulebases (with respect to model accuracy and comprehensibility). Before experiment analyses are presented, Chapter 5 describes the experimental methodology employed.

Chapter 5

Experiment Preliminaries

A major consideration in the design of a testing and evaluation methodology involving different learning algorithms is to provide, as far as possible, a replicable testing environment for each dataset over all the algorithms being tested. This ensures that one can be reasonably certain that any difference in performance arises due to the learning algorithms used, and not, for instance, to different fuzzification or other transformation techniques used on a dataset.

It is not always possible to replicate exactly the testing environment for each algorithm, and this certainly includes the case when comparing crisp and fuzzy induction algorithms, where it might be natural to question the impact of the different knowledge representation languages used. However, it is considered worthwhile to compare *FRANTIC* with more well-established crisp learning techniques, especially with some that are known to produce highly accurate – even if not comprehensible – classifiers.

The first section in this chapter provides a rationale for the choice of algorithms against which *FRANTIC* is compared, and the datasets used for this purpose. The following three sections describe the evaluation criteria and the measures undertaken to replicate the testing conditions for each dataset used on each algorithm, and make explicit the situations where this is not possible. Section 5.5 describes the systematic approach adopted for setting *FRANTIC* parameter settings, while the final section gives an overview of the following results chapters, describing the main question addressed by each.

5.1 The Datasets and Other Algorithms

The datasets utilised in the following chapters were chosen partly because they are in common use by researchers when testing new learning algorithms, or modifications and extensions to existing ones, but mainly due to the range of real-world issues they cover. These are:

- heavily imbalanced datasets resulting from an uneven distribution of the instances between the classes,
- a small number of instances to describe some classes in a dataset
- no clear boundary between classes, and
- datasets where the number of attributes to describe an instance may be close to or greater than the actual number of instances.

These datasets are all derived from real-world data and cover different application areas, including bioinformatics, forensic investigation, and industrial plant monitoring. All but one are obtainable from the UCI Machine Learning Repository [7], with the exception – the Leukaemia dataset – obtainable from The Broad Institute [1]. Different datasets present different problems, including multi-class induction from small sample sizes (the Glass dataset), extreme imbalance in the distribution of instances between the classes (Water Treatment), and an approximate equal proportion of attributes and instances in a dataset (Leukaemia).

Table 5.1 on the next page lists basic features of the datasets. The number of instances, attributes, and classes in a dataset ranges from small to modest, while the distribution of the classes in a dataset varies considerably from one to another and ranges from equal, to approximately equal, to heavily imbalanced. Due to rounding the percentages in the last column of Table 5.1 may not total to exactly 100%.

The number of attributes listed is the number of conditional attributes, i.e. excludes the class attribute. All conditional attributes in all datasets are real-valued and continuous, since a focus of this work is on inducing fuzzy rules which are used to best advantage in situations where real-valued observations are made. However, no fundamental changes are necessary to enable *FRANTIC* to deal with datasets that contain a mixture of nominal and continuous attributes, since crisp sets are ultimately a specialisation of fuzzy sets and such a dataset may be formatted appropriately[1]. Developing an efficient production system to deal with large volumes of data is beyond the scope of this research. However, *FRANTIC*'s computational complexity and its potential for dealing with sizeable datasets generated in domains such as astronomy and bioinformatics is explored in Section 7.3.

Unless explicitly mentioned in this chapter, no transformation of data values, such as normalisation to transform values in a specific range, has been performed on any of the datasets, for use with any of the algorithms mentioned below. More detailed descriptions of each dataset are found in Section B.1 on page 211.

[1]For instance, an attribute with nominal values in its domain may be deemed to have as many 'fuzzy' sets as the number of nominal values, and an observation in an instance for such an attribute may be deemed to have a membership degree of 1 for the appropriate fuzzy set/nominal value, and a membership degree of 0 for all other fuzzy sets/nominal values

Table 5.1: Dataset properties

Dataset	Instances	Attributes	Classes	Comments
Wine	178	13	3	Well-posed problem with separable classes, used as a 'control' set. Domain: agriculture and cultivation. Distribution: 33% : 40% : 27%
Iris	150	4	3	Extensively used dataset for classification testing. Only one class is linearly separable from the other two. Domain: plant taxonomy. Distribution: 33.3% : 33.3% : 33.3%
Glass	214	9	6	Larger number of classes highlights important future work for *FRANTIC*. Several classes contain only a few instances. Domain: forensic science. Distribution: 33% : 35% : 8% : 6% : 4% : 14%
WT	377	5	2	Highly-imbalanced dataset with minority class containing only a few instances. Domain: industrial plant monitoring. Distribution: 98% : 2%
Leuk	72	50	2	Dataset commonly used to test machine learning techniques in bioinformatics applications. Domain: Cancer diagnosis and classification. Distribution: 65% : 35%

For datasets with a small number of attributes such as the Iris dataset, an exhaustive approach to rule construction will result in the most accurate rulebase. For instance there are 255 unique simple propositional rules to describe a class for this dataset[2], and running *FRANTIC* for 100 iterations with ten ants each creating a rule, say, might seem an overkill. However, exhaustive approaches are not viable in practice and the construction of specific rules is generally constrained by system parameter settings (the rule construction parameters in the case of *FRANTIC*) to prevent this type of search. This is the also the case with the algorithms against which *FRANTIC* is compared and which are discussed next; it is therefore still informative to compare the performance of different algorithms on datasets with a small number of attributes. The impact of *FRANTIC*'s construction parameters on the accuracy of rules and the size of the solution space searched is discussed further in Section 6.3.2 on page 101 and Section 7.2.1 on page 149 respectively.

Table 5.2 on the following page provides an overview of the algorithms against which *FRANTIC*'s performance is assessed. They include a prominent crisp decision tree learner, a probabilistic classifier, a support vector machine, and several fuzzy decision tree and rule induction algorithms. The constraints and aims that resulted in these choices are:

1. availability of a software implementation for an algorithm that enables easy replication of testing conditions;

[2]With four attributes and three values within the domain of each attribute in the Iris dataset, and considering the order of attributes to be unimportant, there are 12 different rule antecedents of size one (i.e. only one attribute represented), 54 different rule antecedents of size two (two attributes represented), 108 different antecedents of size three, and 81 different antecedents where each of the four attributes is represented.

Table 5.2: Algorithms against which *FRANTIC* is compared

Algorithm	Fuzzy/Crisp	Induction and model summary
C4.5	Crisp	Well-established decision tree induction algorithm, able to deal directly with numeric attributes. Derived model explicitly captures decision-making process.
NB	Crisp	Naïve bayes, statistical model based on Bayes' rule of conditional probabilities. Determines probabilities for each class when classifying instances.
SMO	Crisp	Support vector machine, hybrid between linear and instance-based methods. Selects subsets of training set that lie on the boundaries between different classes. Considered as generally very accurate classifiers.
QSBA	Fuzzy	Uses subsethood values to attach quantifiers to linguistic values of IF-THEN rules. No parameters to be set.
FSBA	Fuzzy	Uses subsethood values to select linguistic values for IF-THEN rules.
NEFCLASS	Fuzzy	Uses grid-based approach to determine initial set of linguistic IF-THEN rules, and then a backpropagation-like method to tune fuzzy sets.
FID3.4	Fuzzy	Uses methods from fuzzy sets and approximate reasoning to induce fuzzy decision trees.

2. a mix of crisp and fuzzy induction algorithms;

3. a diverse range of more well-established machine learning techniques.

Implementations for the first three algorithms are found as part of the *Weka* system[3], while the fourth and fifth implementations are courtesy of Dr R. Jensen of the Department of Computer Science, University of Wales, Aberystwyth. Implementations of both *NEFCLASS*[4] and *FID3.4*[5] are obtainable from the original designers of the algorithms.

The first two fuzzy induction algorithms, like *FRANTIC*, require fuzzy sets to be pre-determined and provided with the dataset. The second two fuzzy induction algorithms are able to determine their own fuzzy sets prior to and/or during the induction process – how this is managed, in the context of attempting to replicate the testing environment between different algorithms, is discussed in Section 5.4.1.

Descriptions of each of these algorithms, including the parameter settings that lead to the results reported in this research, are provided in Section B.2 on page 215 – in an attempt to obtain optimal results, each algorithm is generally run with different parameters settings and the best results obtained are the ones reported.

[3]The *Weka* machine learning tool is developed and maintained by the Department of Computer Science, University of Waikato, New Zealand, and is available from: `http://www.cs.waikato.ac.nz/~ml/weka/index.html`. An associated book on machine learning is [191].

[4]*NEFCLASS-J*, the Java-based implementation of the system: `http://fuzzy.cs.uni-magdeburg.de/nefclass/nefclass-j/_dld/`

[5]*FID3.4*: `http://www.cs.umsl.edu/~janikow/fid/`

Table 5.3: A two-class confusion matrix

		Predicted Class	
		Positives	Negatives
Actual	Positives	TP	FN
Class	Negatives	FP	TN

5.2 Evaluation Criteria

The term model here refers to the output of an induction algorithm, which is used to predict test cases (instances unseen during training). The models produced by the different algorithms are compared against each other with respect to their predictive capability, and where possible, with regards to model complexity, which is used as indication of the level of comprehensibility of the model.

5.2.1 Predictive Capability

The percentage accuracy on test sets of the models induced by all algorithms is used as a first indicator of their performance.

For two-class imbalanced datasets the additional metrics of True Positive Rate (*TP_Rate*, eq. 5.1) and False Positive Rate (*FP_Rate*, eq. 5.2) are used to evaluate how effective an algorithm is at inducing models that can describe and accurately predict the smaller class. Consider the example of a two-class confusion matrix, Table 5.3, where each ijth element of this matrix, i.e. the element in the ith row and jth column, represents the number of test instances with actual class i that have been classified as class j. The positive class is normally taken to be the minority class (the class with the least number of instances), as this is generally the most difficult class to describe for an induction algorithm.

$$TP\_Rate = \frac{TP}{TP+FN} \tag{5.1}$$

$$FP\_Rate = \frac{FP}{FP+TN} \tag{5.2}$$

The actual class distribution is captured by values that are spread over the two final rows of the confusion matrix, so that any metric that uses values from both rows is inherently sensitive to class imbalances. This includes $Accuracy = (TP+TN)/(TP+FP+TN+FN)$ where in highly imbalanced datasets it is relatively easy to maximise this value by always predicting the

majority class, and $Precision = TP/(TP+FP)$ which is commonly used in conjunction with the *Recall* measure, discussed below.

TP_Rate (known as *Recall* in the information retrieval community) and *FP_Rate* use values from only one row in the contingency table. They consider the performance of an algorithm's induced rule describing the positive class, on the actual positive and negative classes separately and respectively.

The *TP_Rate* measures how capable the rule is in recognising and classifying instances of its own class (the positive instances), while the *FP_Rate* gives an indication of how good the rule describing the positive class is at avoiding misclassifying instances belonging to other classes (the negative instances).

These two metrics are also presented in diagrammatic form as a ROC (Receiver Operating Characteristic) graph. ROC analysis [82] originates in signal detection theory but has been used extensively by the medical community where different costs associated with misclassifications are common (e.g. it is considered to be less 'costly' to classify a benign tumour as malignant and conduct further investigatory tests, than it is to classify a malignant tumour as benign and cease investigation or treatment). This approach to analysis when dealing with imbalanced datasets or unequal classification costs is increasingly being investigated[6].

5.2.2 Model Complexity

The models of the majority of the algorithms tested are in a form that may be evaluated with respect to how easily the information they contain may be understood and validated by humans – these are the models induced in the form of decision trees or rulebases. The extent to which a model is human-comprehensible may be measured by its complexity, or size – generally speaking, the smaller the model, the more likely it is to be easily assimilated by humans.

The metrics obtained from decision trees and rulebases may be compared with each other, since a path from the root of a decision tree leading to a terminal leaf may be considered a rule (so that the number of leaves in a decision tree equates to the number of rules in a rulebase), and the number of decision nodes in a tree path may be equated to the number of terms or conditions in a rule antecedent. In the results reported in this work all nodes in a path are counted. Specifically, if there is more than one node referring to the same attribue – for e.g. a node in a path $\text{AGE} \leq 50$ and another in the same path $\text{AGE} \geq 25$ – then they are all included in the count.

[6]For instance, the 1st Workshop on ROC Analysis in Artificial Intelligence, held as part of the 16th European Conference on Artificial Intelligence 2004; and the 2nd Workshop on ROC Analysis in Machine Learning, held as part of the 22nd International Conference in Machine Learning 2005

A global measurement with which to measure the size of a decision tree or rulebase is therefore the total number of non-terminal nodes in the tree, or the total number of terms in rule antecedents. However, it is often the case that many short rules are in fact easier to understand than a few very large rules, so that two additional metrics that are sometimes used are the number of rules in a rule base (or the number of paths in a tree), and the average number of terms in a rule antecedent (or the average number of non-terminal nodes in a tree path).

5.3 Accuracy Estimation and Model Selection

The results relating to model complexity and accuracy reported in the following chapters are obtained using stratified ten-fold cross-validation. In *k*-fold cross-validation a dataset is randomly split into *k* approximately equal parts, and stratification ensures that each fold contains approximately the same relative proportions of instances for the different classes, as the original complete dataset. Each of the *k* folds is used once as a test set on the model induced from the other $k-1$ folds. This produces *k* individual sets of performance statistics – such as predictive accuracy and number of rules in a rulebase – that are averaged to obtain the final estimates. Standard deviations for the *k* individual performance statistics may also be obtained.

The use of *k*-fold cross-validation as a model selection and evaluation method is supported by several studies. In [117], for instance, the use of stratified ten-fold cross-validation as an accuracy estimation method is proposed, after it has been evaluated against LOO estimation (leave-one-out estimation, *k*-fold cross-validation with *k* equal to the number of instances in the dataset), and two members of the bootstrap family [53], using medium-sized datasets (500+ instances) from the UCI Repository to induce classifiers by *Naive Bayes* and *C4.5*. For small datasets such as those arising in bioinformatics, [20] indicates that bootstrap methods provide improved performance with respect to variance over ten-fold cross-validation (lower variation in performance as a result of the ten different training sets used), but with an increase in bias (increased expected difference in classification performance between an average induced model and the true model generating the data), and high computation cost.

The bootstrap estimator investigated in [114] has a lower variance than *k*-fold cross-validation for linear discriminant classifiers, but not for classifiers produced by 1- and 3-nearest neighbour algorithms and decision trees. The authors state that *k*-fold cross-validation has consistently low bias, reasonable variance if *k* is equal to 10 or greater, and is the most consistent in this regard when compared with the other estimators over the different learning algorithms. Reference [113] also indicates that ten as the value for *k* in *k*-fold cross-validation produces the most reliable estimates, on average, of a classifier's true performance.

Each percentage accuracy and model size statistic reported in this work is therefore the average of the ten figures obtained from the ten models produced by a ten-fold cross-validation. The associated standard deviation is also reported. It should be noted that since *FRANTIC* uses an element of randomness in its rule construction, *ten* ten-fold cross-validations have been run for each dataset, and each reported *FRANTIC* figure is an average of the ten averages obtained from the ten cross-validations.

In order to reduce the variability in performance between algorithms that may arise due to different algorithms using different partitions of the same dataset, exactly the same ten folds of a dataset are used for all the algorithms, i.e. each fold *j* of dataset *i* used by algorithm *A* contains exactly the same instances as fold *j* of dataset *i* used by algorithm *B*, and all other algorithms (irrespective of whether the model induced is crisp or fuzzy). Using the same folds is also particularly useful when exploring different *FRANTIC* parameter settings in Chapters 6 and 7, since any differences arising from varying these parameters may be attributed to the system configuration, and not due to differences in the datasets.

Furthermore, the partitioning of the dataset occurs in such a way that each fold has approximately the same distribution of classes as the original complete dataset – this ensures that an algorithm is presented with a similar induction problem to tackle in all the ten training stages of a ten-fold cross-validation test, and that each test set has approximately the same class distribution as each training test.

5.4 Fuzzy Specifics

When comparing fuzzy induction algorithms, variability in performance as measured by accuracy and model complexity may arise due to reasons other than the different induction methods being compared, or the partitioning of the dataset used during cross-validation. Specifically, data fuzzification and the inference process used for classification may also have an impact on performance.

5.4.1 Data Fuzzification

In order to enhance comprehensibility of the induced fuzzy models, and to aid illustrations and discussions in later chapters, only three fuzzy sets (*low*, *medium*, *high*) are defined for each attribute of all datasets, except for the Water Treatment dataset. The fuzzy sets are determined using a simple method based on the mean and standard deviation of attribute values[7], and no

[7]The implementation of the fuzzy set generator is courtesy of Dr Richard Jenson, and obtainable from `http://users.aber.ac.uk/rkj/programs/index.php`

further optimisation is generally carried out. The fuzzy sets for the Water Treatment dataset were obtained from the authors of [169; 170] – three fuzzy sets (*low*, *medium*, *high*) for the first three attributes and two (*low*, *high*) for the remaining two.

Attributes (irrespective of the dataset) with three fuzzy sets have a left-shouldered trapezoidal set representing *low*, a triangular one representing *medium*, and a right-shouldered one for *high* (see Fig. A.2 on page 207 for an example). Attributes with two fuzzy sets have one left-shouldered and one right-shouldered trapezoidal fuzzy set representing *low* and *high* respectively.

These fuzzy sets are used for *FRANTIC*, *QSBA* and *FSBA*. *NEFCLASS* is seeded with these same sets, i.e. the initial induction of rules is done using these same sets, but is allowed to tune these sets during the different rule and rulebase pruning steps. *NEFCLASS* may therefore be deemed to have an added advantage over the other algorithms since it can optimise these sets.

FID3.4 requires user-defined fuzzy sets as input, or user-defined restrictions on the minimum and maximum numbers of fuzzy sets that it generates for each attribute. However, any user-defined sets must be normalised (i.e. the x-axis values should range from 0 to 1, and not over the range of original attribute values), which also requires the datasets to be normalised and have attribute values in the range [0,1]. In order to maintain as far as possible the same testing environment over all algorithms, all datasets except one – the Leukaemia dataset – are inputted to *FID3.4* in their original unnormalised state. The system is left to define its own fuzzy sets under the constraint, however, that the minimum and maximum numbers of fuzzy sets for each attribute are set to two or three (i.e. as for the other fuzzy induction algorithms).

The actual number of fuzzy sets generated by *FID3.4* for each attribute of each dataset is one, two, or three. If the system creates one fuzzy value for an attribute then it has decided that the attribute is irrelevant in the classification of instances. The shapes generated for two- or three-set attribute linguistic values are the same as for the other algorithms, i.e. left- and right-shouldered sets for an attribute defined by two fuzzy sets, and left-, triangular and right-shouldered sets for an attribute defined by three.

For each attribute, as well as the minimum and maximum numbers of fuzzy sets, a lowerbound and upperbound value must also be inputted. The Leukaemia dataset has negative values for several attributes and because the implementation does not allow a user to define a negative lowerbound, the dataset was normalised so that values of all attributes lie in the range [0,1]. This transformation is achieved by:

$$\tilde{att}_i^j = \frac{att_i^j + |\min_i(att^j)|}{\sum_i (att_i^j + |\min_i(att^j)|)} \tag{5.3}$$

where att_i^j is the value of attribute j in instance i, $|\min_i(att^j)|$ is the absolute value of the smallest value of attribute j over all instances, and $\tilde{att}_i^j$ is the new normalised value for attribute j in instance i. The term $|\min_i(att^j)|$ is only present in the formula if the attribute being normalised has negative values – it enables all values to be first turned into zero or positive values before normalisation, and this ensures that any values that were originally equal to zero still maintain their relative order amongst the new normalised values[8].

5.4.2 Replicating Fuzzy Rule Matching and Inferencing

As described in Appendix A, all fuzzy rules are applied to an instance requiring classification – a degree of match is found between each rule and the instance, and how a decision is reached as to which class is assigned to the instance is handled by the particular inference method.

The process of finding a degree of match between an instance and a rule, and the inference method utilised by *FRANTIC* are described fully in Sections A.3 on page 208 and A.4 on page 209. Naturally, different methods may be utilised, but these two were used to generate the results reported in the following chapters as they are also the methods utilised by *QSBA* and *FSBA* – this makes the comparison between the different algorithms equitable.

The documentation with *NEFCLASS* indicates that the *t-norm* used in determining the degree of match is the *min* operator, and it has a facility to set its *s-norm* to the same operator as that used by *FRANTIC* (the *max* operator), making the matching processes equivalent. *NEFCLASS*'s inference method is also the same as *FRANTIC*'s, a single-winner based one (described in Appendix A.3 on page 208).

FID3.4 offers several different options, and parameters have been set to make its fuzzy matching and inferencing methods equivalent to that of *FRANTIC* and the other fuzzy algorithms. Parameters relating to determining the degree of match are set to the *min* operator, the overall inference type is set to a fuzzy set-based method, with internal conflicts – arising when a leaf in the decision tree contains training examples of different classes – resolved using the majority class, and external conflicts – when a test instance ends up with non-zero degrees of match with more than one leaf – resolved by ignoring all leaves except the one in which the instance to be classified has the highest degree of membership.

[8] If this term is not included, an original attribute value of 0 is transformed to 0, while negative values are transformed to values greater than 0.

Table 5.4: *FRANTIC* simplified and full IRL mode default parameter settings for exploratory runs

Parameter	simplified IRL	full IRL
`representation`	negation	negation
`numIterations`	100	100
`numAnts`	10	10
`minInstPerRule`	50%,60%,...,90%	30%,40%,...,70%
`constructionThreshold`	0.5,0.55,...,0.95	0.5,0.55,...,0.95
`fitnessThreshold`	0.5	0.5
`removalThreshold`	n/a	0.5
`maxClassInstUncovered`	n/a	10%

5.5 Setting *FRANTIC* Parameters

FRANTIC has several parameters, and in setting the values that produce the results presented in the following chapters, a simple and systematic but certainly not comprehensive approach has been adopted. It is therefore quite possible, and in some cases probable (discussed later), that better *FRANTIC* results (in terms of accuracy) may be obtained.

For each dataset, several exploratory ten-fold cross-validation runs are carried out with default values for almost all parameters, and variations in values for the remaining two, `minInstPerRule` and `constructionThreshold`, Table 5.4. It is obvious therefore, that several of these parameters are *not* exploited to full potential in this research, mainly because early *FRANTIC* results indicated that the two rule construction parameters have the greater impact on the quality of the rulebases produced.

In these preliminary runs the parameter `minInstPerRule` is set to greater values for simplified IRL than for full IRL, since if only one rule is created to describe a class, it is reasonable to assume that a rule that is forced to cover more training instances may have greater generalisation power. With regards to `constructionThreshold`, the valid range of values is [0,1]. However, as discussed in Section 4.3.3, if the value is set too low then rules with many conditions in the antecedent are created, and if it is set too high then an overly specialised rule is generated that matches only a few instances in the training set.

When investigating how and why *FRANTIC* works in Chapters 6 and 7, `constructionThreshold` is set in the range [0.5,0.95], with a step value of 0.05, and the values used for `minInstPerRule` are the ones that produce rulebases with the highest classification accuracies in the preliminary tests. The `minInstPerRule` parameter setting is also varied in some of the experiments by selecting additional values from the preliminary runs that are close to the original one

selected, and which lead to rulebases with similar accuracy. The two construction parameters tend to be dataset-dependent, so that they often have different values for different datasets, and different values for a simplified IRL run versus a full IRL run for any one dataset.

If the preliminary runs also indicate that all ants quickly converge to creating the same rules, then `numIterations` is reduced from 100 to 50 or 70 for some of the datasets in these later experiments discussed in Chapters 6 and 7. Other parameters that are varied in these investigations are the number of ants, and the form of the hypothesis language used. These details are provided where appropriate in the results chapters and also listed for reference purposes in Section B.3 on page 220. The *FRANTIC* parameter settings that produce the rulebases used to compare with models produced by the other learning algorithms are provided in Table B.4 on page 221 – these rulebases are selected from those discussed in Chapters 6 and 7 and are generally the ones leading to the higher classification accuracies.

The parameter values for running *FRANTIC* in simplified SRL mode are determined from those set for simplified IRL mode – all values are the same with the exception of `numIteration`. Since simplified SRL conducts more evaluations than simplified IRL, and in order to make the comparison between the two approaches more equitable, `numIterations` is reduced to 30 for simplified SRL.

5.6 Overview of Results Chapters

The main aim of the experiments conducted and analysed in Chapters 6 and 7 is to gain an in-depth understanding of how *FRANTIC* behaves on real data, its strengths and current limitations. In doing so, the direction for future work is also clarified.

Chapter 6 investigates the impact of various factors in the ACO rule construction mechanism, such as the heuristic and the rule construction parameters, which affect both the iterative and simultaneous rule learning strategies. Chapter 7 explores the impact of the rule learning strategies themselves, the computational cost involved, and ways of scaling up the system to deal with larger problems.

The final results chapter, Chapter 8, places *FRANTIC*'s performance in the wider context of other learning algorithms based on more well-established approaches. *FRANTIC*'s capabilities with respect to producing rulebases that are both accurate and comprehensible are highlighted.

Chapter 6

Constructing Individual Rules

This chapter analyses results obtained from *FRANTIC* running in a simplified IRL mode, in an attempt to understand the impact of the different elements of the rule construction mechanism that guide term selection and retention – the heuristic, the pheromone levels, the construction parameters `minInstPerRule` and `constructionThreshold`, and the degree of richness of the knowledge representation/hypothesis language used to construct a rule. Some elements influence the accuracy and complexity of the rulebases induced, and others the speed at which optimal rulebases are found.

It should be remembered that rule construction is identical in *FRANTIC* irrespective of the strategy it follows, whether it is simplified iterative rule learning, or full iterative rule learning, or simultaneous rule learning. However, any differences in performance that may arise from the learning strategy used are discussed in detail in Chapter 7.

The *FRANTIC* parameter settings used to generate the results presented in this chapter are given in Table 6.1. For the parameter `representation`, 'Negation' denotes rules constructed with negated terms in the antecedent, 'Simple' denotes simple propositional rules, 'Disjunction' rules with internal disjunction between attribute values, and 'Hedges' rules with both negated terms and linguistic hedges.

Table 6.1: *FRANTIC-simpIRL* parameter settings for exploring the system's performance

	Wine	Iris	Glass	WT	Leuk
`representation`	— Negation (Simple, Disjunction, Hedges) —				
`constructionThreshold`	— 0.50, 0.55, ..., 0.90, 0.95 —				
`minInstPerRule`	(50%) 60% (70%)	(40%,50%) 60%	50% (60%,70%)	70% (80%,90%)	(70%,80%) 90%
`numAnts`	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10
`numIterations`	50	100	50	100	70
`fitnessThreshold`	0.5	0.5	0.5	0.5	0.5

In these experiments the `constructionThreshold` parameter is always set to the values in the range $[0.50, 0.95]$, with a step value of 0.05, and generally the form of the hypothesis language used is rules that contain negated terms, the number of ants is 10, and the `minInstPerRule` setting for each dataset is as indicated in the table. When these parameter settings are varied in some experiments, the additional values used are indicated in the brackets.

The value for `minInstPerRule` is dataset dependent and how this is obtained for each dataset by running preliminary tests is explained in Section 5.5 – the `minInstPerRule` value chosen from these preliminary tests is the one that appears to lead to rulebases with greater generalisation power. The two additional values in brackets used in some of the experiments are the values closest to the initial `minInstPerRule` setting that appear to lead to rulebases with similar classification accuracy. Unless otherwise stated, the setting used for each parameter is the one that is *not* in brackets in Table 6.1; when the additional parameter settings in brackets are used this is clearly indicated in the text.

Error bars on graphs denote (unless otherwise stated) the standard error of the mean for the main characteristic depicted (such as the average classification accuracy or the average number of terms in a rulebase). Tables generally detail the main characteristic, the associated standard error of the mean (SEM) and standard deviation (STD). Since 10 ten-fold cross-validations are carried out, the STD is the average of the 10 standard deviations obtained from each of the 10 cross-validations (this gives an indication of *FRANTIC*'s robustness to the different training sets used in *one* ten-fold cross-validation, though the inherent randomness in the algorithm will also have some impact), and the SEM is the standard deviation of the 10 average accuracies from each of the 10 cross-validations (this gives an estimate of the reliability of the main statistic presented in a graph or table).

6.1 The Heuristic

The use of a heuristic is standard in the application of ACO, and is considered an important element in guiding the early construction of optimal solutions, especially if no local search is used to improve solutions [50, p. 97]. The heuristic implemented in *FRANTIC* is based on the use of subsethood values, as discussed in Section 4.1.2. It is a popular measure to use in fuzzy rule discovery, and it should be remembered that two of the algorithms against which *FRANTIC* is compared in Chapter 8 – *QSBA* and *FSBA* – also use the concept of subsethood values.

QSBA and *FSBA* use the subsethood value of each term associated with each class directly – to either determine the weight to be attached to the term in the rule antecedent (*QSBA*), or, in

conjunction with a user-defined threshold, to determine whether that term should be present in the rule antecedent (*FSBA*). *FRANTIC* uses subsethood values as a *partial* guide in determining the next term to be selected by an ant during rule construction – the amount of pheromone associated with a term is also taken into account (and changes from iteration to iteration), and the selection is probabilistic so that it is not necessarily the term that has the highest combined subsethood value and pheromone level that gets selected.

It is reasonable to question, therefore, how much of *FRANTIC*'s performance may be attributed to the use of the subsethood values processed from the training set before learning, and how much to the other elements in the rule construction process, such as the reinforcement mechanism as implemented through artificial pheromone trails, or the parameters that regulate how specialised or general a rule is. The results presented in this section are an initial attempt to determine the influence of the heuristic in *FRANTIC*'s rule construction.

6.1.1 Impact on Classification Accuracy

Table 6.2 on the following page compares the accuracy, standard error of the mean accuracy, and standard deviation of rulebases induced with (+Heuristic), and without (−Heuristic), the use of heuristic information during rule construction.

It is clear that no significance difference arises in rulebases induced with or without the use of heuristic information, at least for these datasets – if, for any one dataset, there is a slight increase in the accuracy obtained using a particular `constructionThreshold` value when running without the heuristic, then there is a slight decrease in the accuracy obtained using a different `constructionThreshold` value. For the Iris dataset (Table 6.2(b)) there is no difference in figures (to 2 decimal places) for `constructionThreshold>0.70`, and for the Water Treatment dataset (Table 6.2(d)) there is no difference (to 2 d.p.) for any `constructionThreshold` value. Though not shown here, the actual rules induced when running with and without heuristic information are also very similar to each other. Possible reasons for these results are discussed later on in this chapter and the next one, when investigating the strong influence of the rule construction parameters.

6.1.2 Impact on Solution Convergence

In this work convergence is deemed to occur when all rules produced by all ants within an iteration are identical, *and* remain so in all subsequent iterations until the final iteration is completed. The interpretation here is that all ants have converged to an 'optimal' solution, as directed by the system parameters and an element of chance in the solution contruction.

Table 6.2: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced with (+Heuristic) and without (−Heuristic) use of heuristic information during rule construction

(a) Wine

	+Heuristic			-Heuristic		
construction Threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	82.95	1.80	8.77	83.52	1.88	9.47
0.55	86.86	1.87	7.51	85.73	2.68	7.43
0.60	86.05	1.45	7.61	87.11	1.35	7.82
0.65	85.35	2.35	7.51	85.09	1.31	7.06
0.70	88.09	1.59	9.28	88.66	1.47	8.15
0.75	90.47	1.32	5.31	90.70	1.35	5.20
0.80	93.01	0.77	5.44	92.61	0.57	5.10
0.85	92.20	1.04	5.13	91.88	1.52	5.23
0.90	85.67	0.80	8.38	86.04	0.65	8.47
0.95	84.74	0.58	7.32	85.29	0.58	7.54

(b) Iris

	+Heuristic			-Heuristic		
construction Threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	86.73	0.73	11.41	86.80	0.69	11.10
0.55	86.33	0.47	10.73	86.67	0.77	10.80
0.60	85.27	0.66	11.08	85.20	1.08	11.30
0.65	88.80	0.76	9.71	88.60	0.66	9.90
0.70	93.33	0.00	7.03	93.33	0.00	7.00
0.75	75.33	0.00	7.06	75.33	0.00	7.10
0.80	76.67	0.00	6.48	76.67	0.00	6.50
0.85	76.67	0.00	6.48	76.67	0.00	6.50
0.90	75.33	0.00	6.32	75.33	0.00	6.30
0.95	63.33	0.00	11.86	63.33	0.00	11.90

(c) Glass

	+Heuristic			-Heuristic		
construction Threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	53.74	1.93	8.98	54.55	1.58	8.21
0.55	53.09	2.74	11.90	53.20	1.67	11.47
0.60	55.63	1.64	9.66	55.67	0.99	9.43
0.65	49.77	0.54	11.74	50.04	0.77	12.46
0.70	45.82	1.03	15.35	45.67	1.44	15.13
0.75	42.00	0.62	16.89	41.68	0.58	17.49
0.80	47.54	0.65	16.09	47.52	0.23	16.08
0.85	44.61	0.38	15.31	44.36	0.56	15.19
0.90	42.91	0.95	11.72	43.04	0.82	11.88
0.95	45.22	0.57	14.21	45.27	0.47	14.36

Table 6.2: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced with (+Heuristic) and without (−Heuristic) use of heuristic information during rule construction (cont.)

(d) Water Treatment

	+Heuristic			-Heuristic		
construction Threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	72.24	0.00	8.00	72.24	0.00	8.00
0.55	74.35	0.00	8.39	74.35	0.00	8.39
0.60	71.89	0.00	6.90	71.89	0.00	6.90
0.65	71.89	0.00	6.90	71.89	0.00	6.90
0.70	70.84	0.00	6.50	70.84	0.00	6.50
0.75	72.16	0.00	7.38	72.16	0.00	7.38
0.80	74.83	0.00	7.72	74.83	0.00	7.72
0.85	79.02	0.00	5.92	79.02	0.00	5.92
0.90	82.78	0.00	6.11	82.78	0.00	6.11
0.95	82.78	0.00	6.11	82.78	0.00	6.11

(e) Leukaemia

	+Heuristic			-Heuristic		
construction Threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	89.29	2.03	15.99	89.39	2.31	16.00
0.55	90.38	1.13	16.24	90.52	0.73	16.15
0.60	92.68	1.04	10.05	92.93	1.23	8.99
0.65	92.58	2.16	11.01	92.24	1.99	11.86
0.70	92.18	1.87	10.30	91.27	2.59	11.98
0.75	93.57	0.79	11.54	93.40	1.23	12.09
0.80	93.78	0.99	9.60	93.65	1.29	9.56
0.85	95.63	1.55	7.07	95.00	1.60	7.68
0.90	93.35	1.21	9.95	92.76	1.53	9.94
0.95	93.11	1.09	8.52	93.13	0.74	7.90

The iteration number at which convergence occurs is here used as a measure of how quickly *FRANTIC* is able to reach an optimal solution. Note, that if there are n classes within a dataset, then there will be a convergence iteration number for each class, since one ACO is used to find a best rule per class – the average of these individual class iteration numbers is used as a convergence indicator for the dataset for that particular run. Hence, a convergence value for a dataset D run m times, is determined by finding the convergence value for each run, and then averaging over the total number of runs:

$$conV(D,m) = \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} itn_{ij} \tag{6.1}$$

where itn_{ij} is the convergence iteration number of class i in run j. A run is here defined as processing one fold of a k-fold cross-validation test. Since ten ten-fold cross-validation experiments are carried out to obtain *FRANTIC* statistics, each dataset – or 90% of each dataset – is run 100 times. All parameter settings for each run are identical.

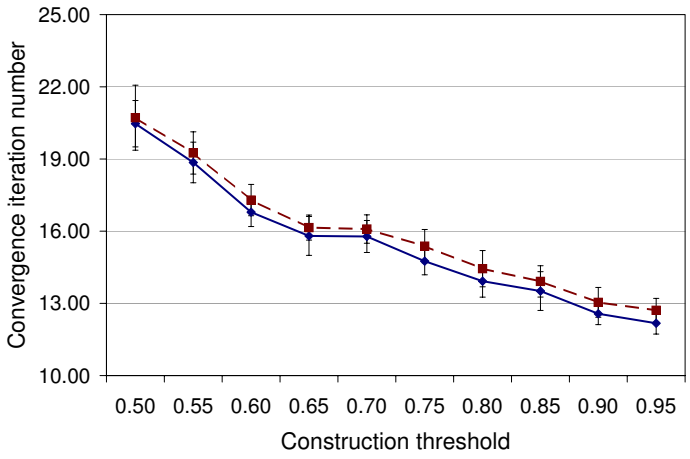
The convergence values that correspond to the classification accuracy results presented in Table 6.2 are plotted in Fig. 6.1 (with the actual values that generate the graphs in Fig. 6.1 presented in Table C.1 on page 224). It should be observed that for all datasets, although 50, 70 or 100 iterations are run for each ACO, on average convergence occurs well before the final iteration is completed. It appears that the the rule discovery mechanism generally converges to optimal rules reasonably quickly for these datasets. However, the overlap of the error bars in almost all `constructionThreshold` values for all datasets indicate that the use of the heuristic does not significantly improve speed of convergence.

Investigating individual *class* convergence reveals little additional information. A class convergence iteration value is the average of the iteration convergence numbers for class i over the total number of dataset runs m:

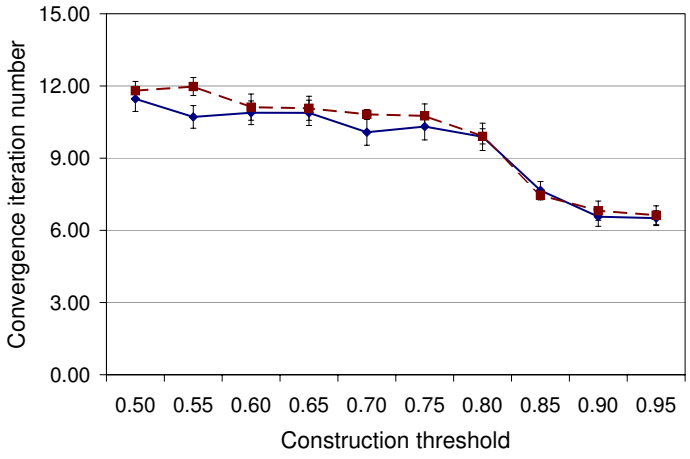
$$conV(D,i,m) = \frac{1}{m}\sum_{j=1}^{m} itn_{ji} \tag{6.2}$$

where itn_{ji} is the convergence iteration number in run j for class i.

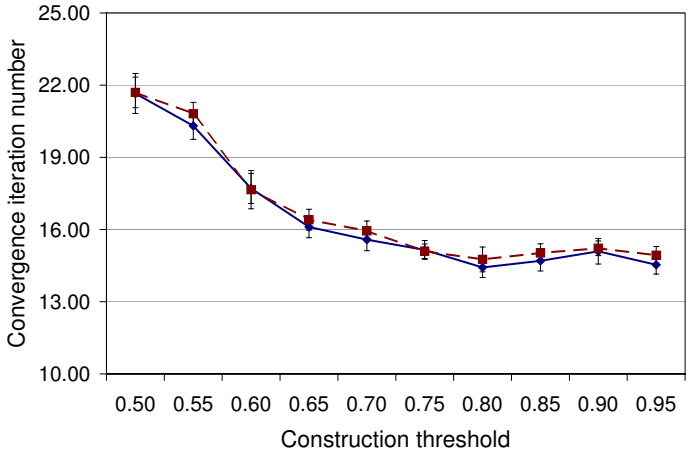
Figure 6.2 on page 92 plots the class convergence iteration values for the Wine dataset, and though error bars are not shown they do overlap for most of the `constructionThreshold` values. This dataset was chosen primarily because though not statistically significant, convergence occurred later for each `constructionThreshold` value if heuristic information was not used – since this dataset poses a simpler problem with all classes being linearly separable, it was expected that similar results regarding the impact of heuristic information on convergence would be observed for the individual classes. However, for all three classes of this dataset (Fig. 6.2), a few settings of `constructionThreshold` result in very similar convergence values irrespective of whether the heuristic is used or not. In one or two cases convergence is quicker without the



(a) Wine

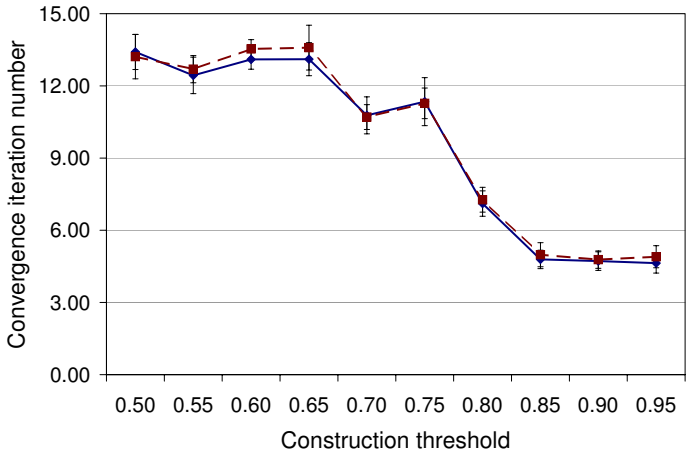


(b) Iris

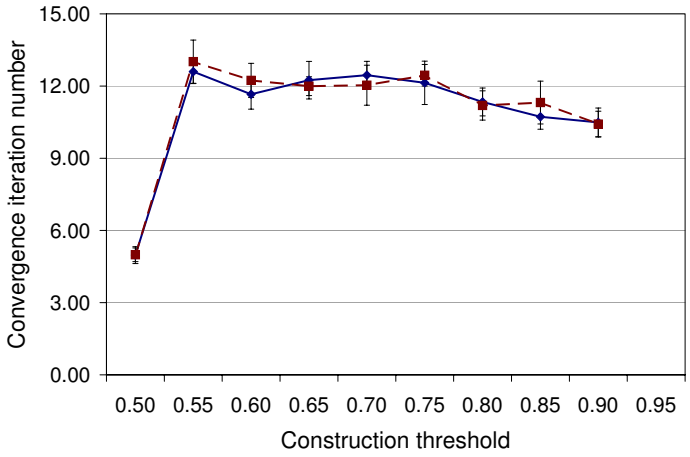


(c) Glass

Figure 6.1: Average convergence iteration values for *FRANTIC-simpIRL* rulebases induced with (solid line) and without (dashed line) use of heuristic information during rule construction

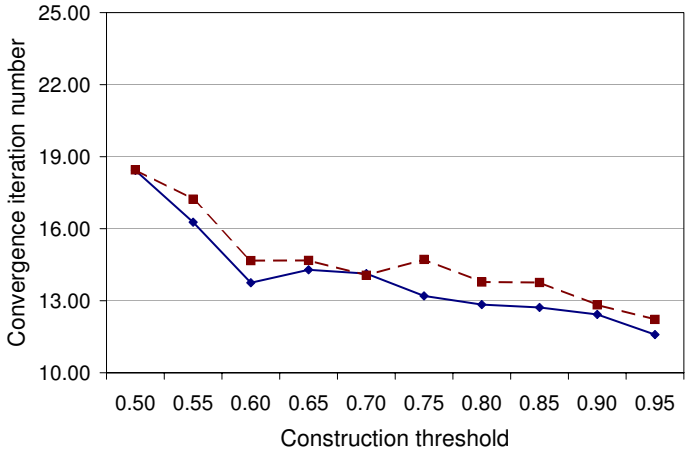


(d) Water Treatment

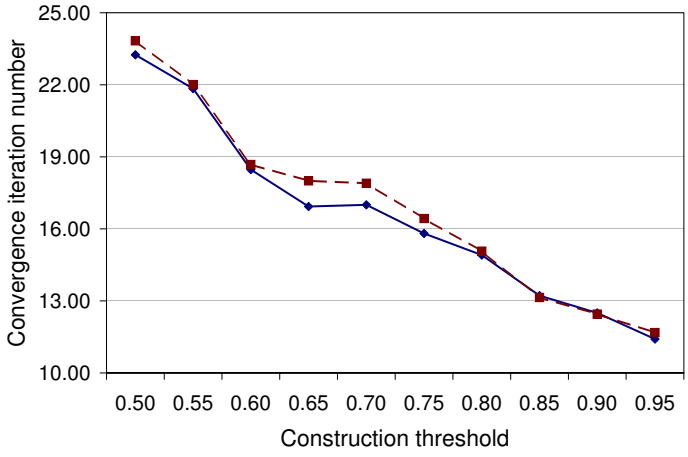


(e) Leukaemia

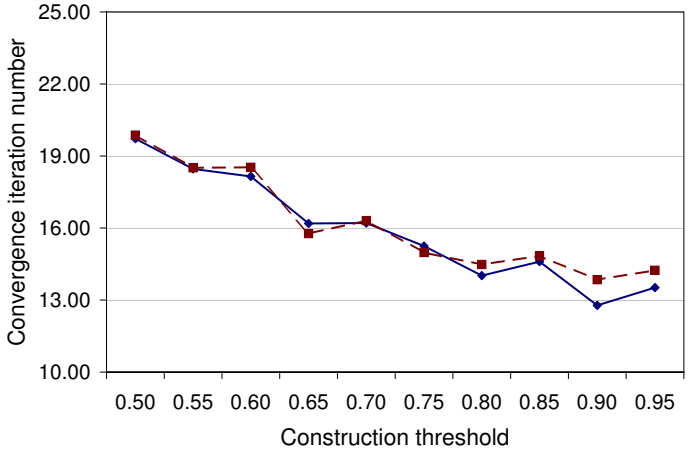
Figure 6.1: Average convergence iteration values for *FRANTIC-simpIRL* rulebases induced with (solid line) and without (dashed line) use of heuristic information during rule construction (cont.)



(a) *Cultivar 1* class



(b) *Cutlivar 2* class



(c) *Cutlivar 3* class

Figure 6.2: Wine dataset – comparison of individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information

use of the heuristic (Fig. 6.2(c)). The possible trend in Fig. 6.1(a) is therefore due to averaging of the individual class results, and not due to the main difference between the datasets, which is that the classes are linearly separable. Though not shown here, the other datasets exhibit similar results with regards to class convergence – most settings of `constructionThreshold` for *each* class within a dataset, result in a convergence value that is as small or smaller with the use of the heuristic, than without, and with no statistical significance.

This subsethood values-based heuristic therefore has minimal impact on the accuracy of the rulebases induced and on the convergence to optimal solutions. The use of the heuristic ensures that from the first iteration of an ACO terms that may be selected during rule construction do *not* have initial equal probabilities, which biases term selection towards those that have a higher heuristic value (since initial pheromone levels are equal for all terms). The most one can conclude at this stage is that this heuristic does not bias term selection *un*favourably. However, one clear trend that may be observed for each dataset in Fig. 6.1 is the tendency for the convergence iteration number to decrease as the `constructionThreshold` value increases. The impact of this parameter, and the anomaly exhibited by the Leukaemia dataset at `constructionThreshold=0.50` (Fig. 6.1(e)), is clarified in the discussions to follow regarding the rule construction parameters.

6.2 The Pheromone Levels

The pheromone levels associated with terms found in a high quality rule, as defined by the fitness function, may eventually become large enough to cause all ants within an ACO to create the same rule over and over – the greater the pheromone level for a term, the greater the probability associated with the term, and therefore the greater the chance that the term is selected. This is confirmed in part by the fact that the results in the preceding section show that all datasets converge for all values of `constructionThreshold`. For a stronger confirmation of this notion, this section investigates what happens *without* the use of pheromone levels to influence term selection.

6.2.1 Impact on Solution Convergence

Another series of *FRANTIC* experiments is run – system parameter settings are the same as used to generate the results in the preceding section, the heuristic is used, but pheromone levels are set equal to one for all terms in the first iteration, and remain so throughout the run of an ACO, i.e. no pheromone updating occurs between iterations. Term probabilities are therefore based on only heuristic values, but term selection is still probabilistic and so an element of

randomness is present in the selection process.

As may be expected, dataset convergence as defined in formula 6.1 does not occur for any `constructionThreshold` value – without the increasing influence exerted by changing pheromone levels on term selection, convergence cannot take place since all terms have the same chance of being selected as they initially did.

However, convergence *appears* to occur for some classes, of some folds in a ten-fold cross-validation run, for some `constructionThreshold` values of some datasets. For instance, with `constructionThreshold=0.85-0.90` for the Glass dataset, all ants create the same rules right from the first iteration for class 1 in fold 5. This is *not* convergence as a result of a changing environment exerting an increasing influence on ants with regards to term selection. The influence to create this one specific rule is a result of the restrictions imposed on the solution/hypothesis space by `minInstPerRule` and `constructionThreshold`, and highlights the importance of these two parameters (also discussed in detail in Section 6.3 on page 98).

Generally, only a strict subset of the total number of terms – here called a *rule construction term set*, or construction set for short – meet the `minInstPerRule` and `constructionThreshold` criteria, and may therefore be added to a rule antecedent. When constructing a rule, the order in which terms from this set are added and retained is often important for later term selections.

For example, consider an example construction set $\{A1, A2, B3, C1\}$, where $A1$ denotes the first value from the domain of attribute A, $A2$ denotes the second value of A, and so on. Each element of this construction set *individually* satisfies `minInstPerRule` and `constructionThreshold` restrictions, i.e. there are at least `minInstPerRule` instances in the training set with $A1$, $A2$, $B3$, *or* $C1$ fuzzified values meeting or exceeding the `constructionThreshold` parameter setting. If $A1$ is added to a rule antecedent, say, then it may mean that $B3$ if selected will not be retained, because $A1$ and $B3$ *together* will cause coverage of instances in the training set to fall below the required `minInstPerRule` number[1]. However, if $A2$ is selected, then $B3$ may also be added. This of course may cause further restrictions on remaining terms in the construction set. The smaller the construction set, the smaller the number of different rules that may be created that satisfy both `minInstPerRule` and `constructionThreshold` so that ants create the same rules. It is reasonable to expect that this occurs more for higher values of `minInstPerRule` and/or `constructionThreshold`, since higher parameter settings impose more stringent restrictions, creating smaller construction sets.

A case in point is the *AML* class of the Leukaemia dataset, and it should be noted that this explains an anomaly in the convergence results for this dataset when investigating the impact of the heuristic in the preceding section. The dataset and class convergence graphs shown

[1] Section A.4 on page 209 provides a detailed example of how the degree of match between a fuzzy rule and fuzzified instance is determined, as implemented in this work.

for almost all classes and all datasets in Section 6.1 (Figs. 6.1–6.2 starting on page 90) and Section C.1 (Figs. C.1–C.4 starting on page 225), show a downward trend for the convergence value as `constructionThreshold` values increase. As discussed in the preceding paragraph this might be expected since a higher setting for this parameter will result in a smaller construction set, and hence fewer different rules that may be created by ants. The impact is that an ACO is likely to converge faster.

However, Fig. C.4(b) on page 229 shows that the exception to this downward trend is for the *AML* class of the Leukaemia dataset with `minInstPerRule=90%` and `constructionThreshold=0.5` – convergence for this class at this `constructionThreshold` value occurs at the first iteration, and this is due to the stringent conditions placed on rule construction by these two parameters, resulting in an extreme case where only one rule is created that satisfies both criteria.

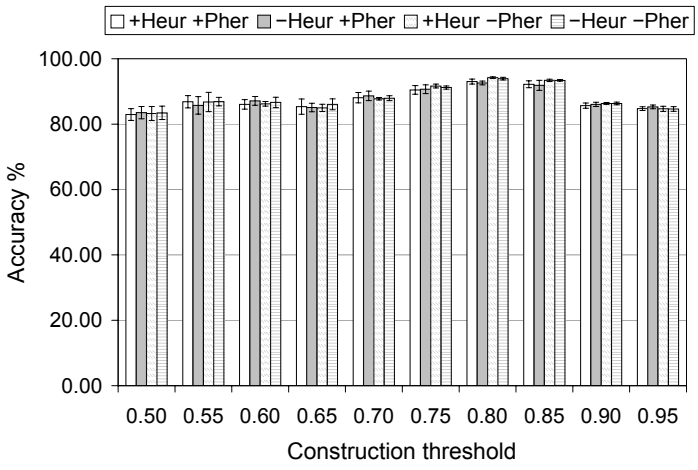
6.2.2 Impact on Classification Accuracy

Another valid question about the reinforcement mechanism implemented through artificial pheromone trails is whether it has an impact on the accuracy of the rulebases induced.

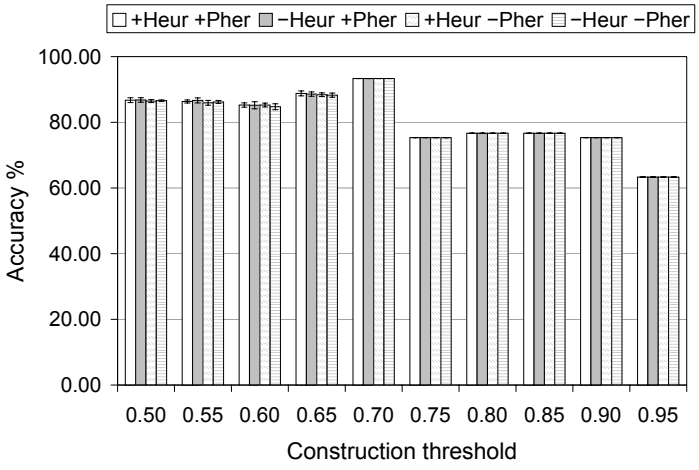
Figure 6.3 on page 97 shows for each dataset the accuracy attained by rulebases induced with and without use of the heuristic (results obtained from preceding section), with the use of the heuristic but without pheromone trails (the accuracy of rulebases induced whose convergence results have just been discussed in this section), and without the use of either the heuristic or pheromone trials. The latter set of results are obtained by setting both the heuristic and pheromone elements in formula 4.9 equal to one, rendering all term probabilities equal so that term selection is now from a uniform random distribution. Error bars denote standard error of the mean.

Figure 6.3 suggests that for these datasets at least, pheromone levels have minimal impact on the accuracy of rulebases induced. For the Iris and Water Treatment datasets (Figs. 6.3(b) and 6.3(d)), there appears to be no difference in average accuracy for almost all `constructionThreshold` settings. For the other datasets, though there may be a (not statistically significant) difference in accuracy, no pattern may be discerned – for some `constructionThreshold` values, rulebases induced without the use of the heuristic and/or pheromones may show an improved accuracy (which may be due to increased exploration possibilities), while rulebases induced with other values might show a better accuracy when induced using the heuristic and pheromone. Table C.2 on page 231 provides the accuracies that generated Fig. 6.3, and the associated standard error of the mean and standard deviation.

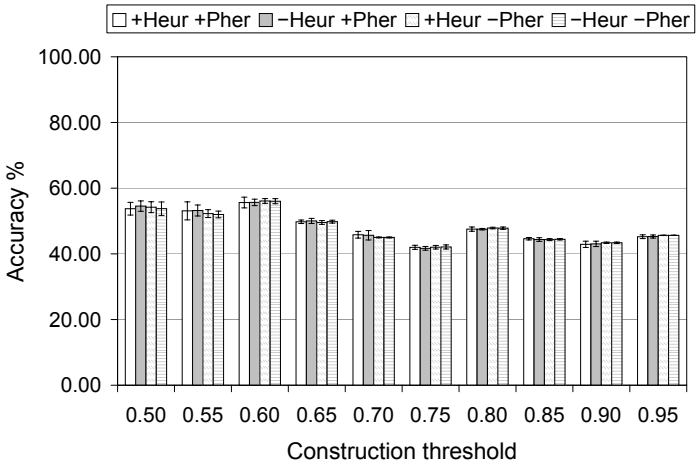
The fact that rulebases with comparable accuracy are induced with or without the use of phero-



(a) Wine

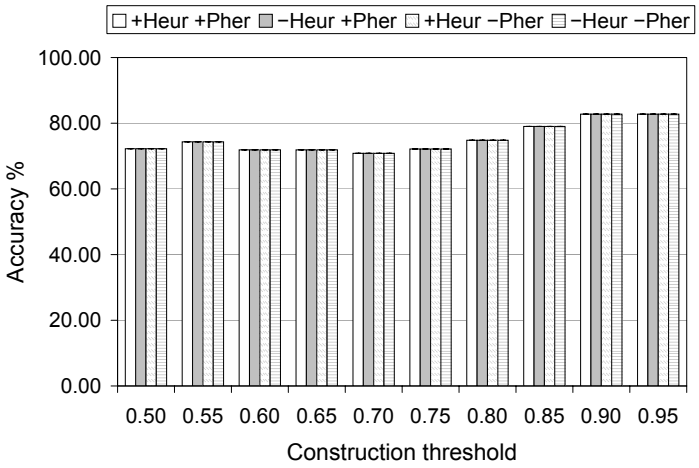


(b) Iris

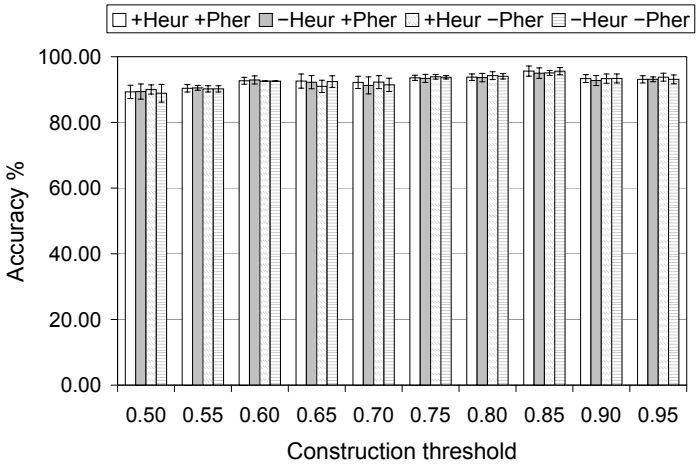


(c) Glass

Figure 6.3: Classification accuracy of *FRANTIC-simpIRL* rulebases induced with and without use of heuristic information and pheromone trails during rule construction



(d) Water Treatment



(e) Leukaemia

Figure 6.3: Classification accuracy of *FRANTIC-simpIRL* rulebases induced with and without use of heuristic information and pheromone trails during rule construction (cont.)

mones is due mainly to the rule selection method – the rule that is added to the final rulebase at the end of an ACO is not the best rule of the final iteration, but the best rule of all iterations. Therefore, if a reasonably good rule is produced at some point during an ACO, even though there is no convergence to that rule, it is included in the final rulebase. The advantage of this selection method is highlighted and discussed in Section 7.8.

Though the pheromone trails have little impact on the accuracy the rulebases induced for these datasets, it is clear that artificial pheromones are essential in helping the system to converge to good or optimal solutions. It appears therefore that the accuracy of the rulebases induced is due to a combination of an element of chance in term selection, and the construction parameters `minInstPerRule` and `constructionThreshold`. These parameters are investigated in the next section, and the usefulness of convergence in controlling computation expense is discussed in Section 7.3.2.

6.3 The Rule Construction Parameters

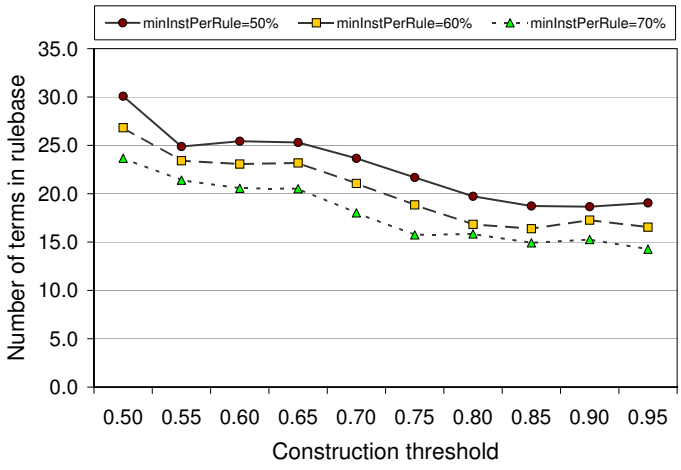
The rule construction parameters – `minInstPerRule` and `constructionThreshold` – determine which terms, if selected, are retained in an antecedent being built by an artificial ant. This section investigates their impact on both the complexity of induced rulebases, and their accuracy.

6.3.1 Impact on Model Complexity

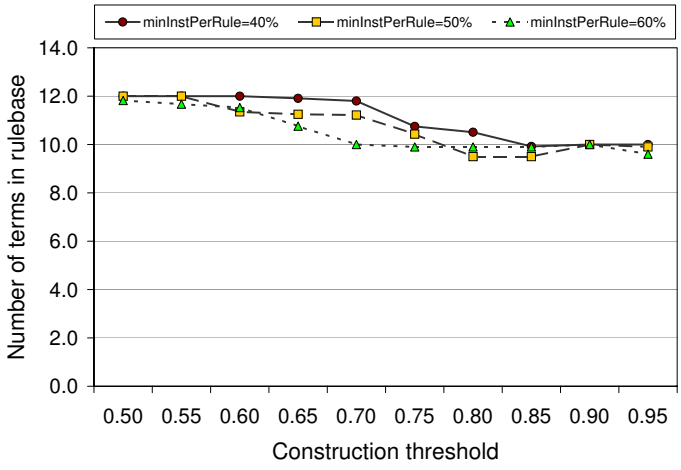
Since *FRANTIC* is here run in simplified IRL mode, the number of rules per class is preset to one. However, the construction parameters directly control which terms may be added to a rule antecedent describing a class in a dataset. As outlined in the preceding section, the higher the values for both parameters, the smaller the size of the rule construction term set, and consequently, the smaller the size of a rule antecedent. This is confirmed in part by reviewing the average number of terms per rulebase for different parameter settings.

Figure 6.4 on the following page provides the average number of terms per *FRANTIC* rulebase over different values of `constructionThreshold` and `minInstPerRule`[2]. Each plotted curve represents a different `minInstPerRule` setting for a dataset (see Table 6.1 for specific parameter settings). The empirical results presented in this figure confirm the description of the impact and interaction between these two parameters with regards to controlling the number of terms

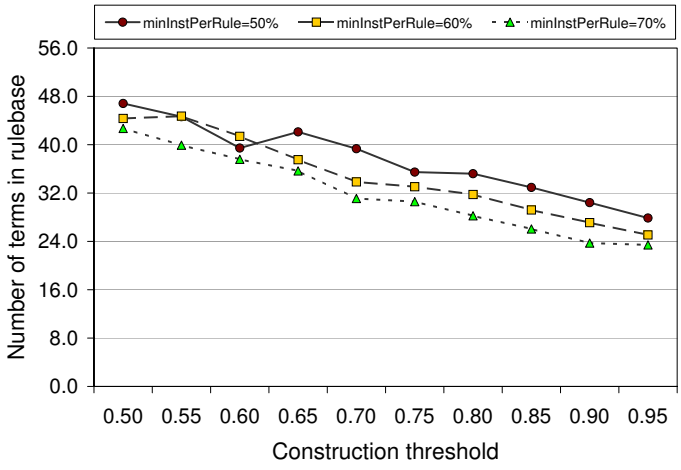
[2]The number of attributes for each dataset is listed in Table 5.1 on page 74. Apart from two attributes for the Water Treatment dataset that each have two values in their domain, all other attributes in all datasets have three values in their domain.



(a) Wine

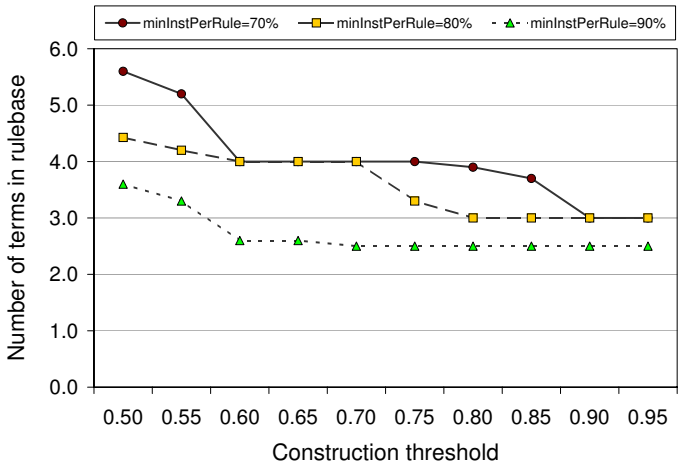


(b) Iris

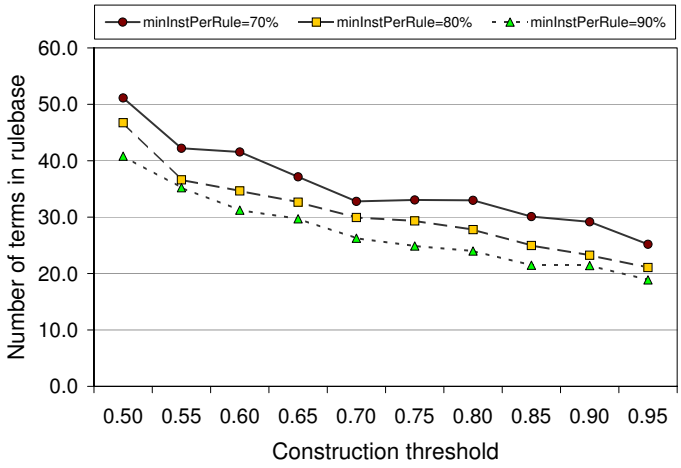


(c) Glass

Figure 6.4: Number of terms in *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`



(d) Water Treatment



(e) Leukaemia

Figure 6.4: Number of terms in *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule` (cont.)

in a rule[3] – for each curve the average number of terms generally decreases as `constructionThreshold` increases, and, the curves with lower values of `minInstPerRule` are generally situated higher up on the graph, indicating that overall they result in a greater number of terms per rulebase (or per antecedent). Though error bars are not shown on the graphs, the standard error of the mean is very small in most cases, suggesting that the difference in number of terms from one `minInstPerRule` value to another is significant.

The exact average number of terms per rulebase, associated standard error of the mean and standard deviations are provided in Table C.3 on page 234. The `minInstPerRule` parameter is also discussed in detail in Section 7.1.2, where it is seen that it has a direct influence on the number of *rules* in a rulebase, when *FRANTIC* is run in full IRL mode.

6.3.2 Impact on Classification Accuracy

The impact of these parameters on the accuracy of induced rulebases is determined by running another series of experiments. *FRANTIC* is run without the use of heuristic information or pheromones trails *and* without the influence of `minInstPerRule` and `constructionThreshold` during rule construction. This means that all terms selected are retained in the rule antecedent. Since no bias in term selection is provided by the heuristic or pheromone levels, term selection is performed from a uniform random distribution, i.e. all terms have an equal chance of being selected. For each rule antecedent, the number of terms that are selected is decided by generating a uniform random integer in the range $[1, numAtts]$, where *numAtts* is the number of conditional attributes in the dataset (i.e. different rules may have a different number of terms).

The number of iterations and number of ants per iteration is the same as for previous *FRANTIC-simpIRL* experiments. This means that exactly the same number of rules are created in this new series of experiments for each class of each dataset, as for the previous ones. The final rulebase is formed by choosing the best performing rule for each class (as defined by *FRANTIC*'s fitness function, Section 4.2.1) out of all the rules created. The results obtained from this set of experiments are referenced under the term of *Random FRANTIC*.

Table 6.3 on the next page provides a comparison between *FRANTIC-simpIRL* (*FRANTIC*) and *Random FRANTIC* (*Random*), with regards to the classification accuracy and complexity of the rulebases induced. Note that the statistics for *FRANTIC-simpIRL* are for when *FRANTIC* is run without use of heuristic information and pheromone trails, in order to ensure as far as possible that differences in results between the two versions arise out of the use of the construction parameters. *FRANTIC-simpIRL* accuracy figures are therefore obtained from the

[3] Dividing the average number of terms in a rulebase in each subfigure by the number of classes for the dataset in question, will give an average number of terms per rule

Table 6.3: Impact of construction parameters on *FRANTIC* rulebases. SEM denotes the standard error of the mean and STD the standard deviation.

(a) Classification accuracy

	FRANTIC			*Random*		
	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
Wine	93.92	0.40	4.91	87.21	1.72	7.47
Iris	93.33	0.00	7.03	94.20	0.55	5.87
Glass	56.03	0.77	10.20	43.71	3.53	12.54
WT	82.78	0.00	6.11	73.44	2.26	13.14
Leuk	95.60	1.09	7.05	79.70	2.92	17.64

(b) Total number of terms

	FRANTIC			*Random*		
	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
Wine	15.81	0.22	1.97	4.50	0.00	1.13
Iris	10.00	0.00	0.00	3.96	0.00	0.94
Glass	38.73	0.13	1.67	12.00	0.00	1.98
WT	3.00	0.00	0.00	4.31	0.00	0.89
Leuk	21.13	0.25	2.18	3.12	0.00	1.03

last three columns of Table C.2 on page 231, choosing the highest accuracy for each dataset. The corresponding rulebase complexity statistics are determined and used in Table 6.3(b).

The two most noticeable observations are that rulebases induced by *FRANTIC-simpIRL* generally have greater accuracy – compare columns 2 and 4 of Table 6.3(a), and complexity – columns 2 and 4 of Table 6.3(b), than those induced by *Random FRANTIC*. An explanation for the second observation is provided first.

Since the number of terms per rule in *Random FRANTIC* is determined by randomly and *uniformly* generating an integer in $[1, numAtts]$, there is a bias to search the space of smaller rules more thoroughly than the space of larger rules. Consider, for instance, the Iris dataset with four conditional attributes. When generating the 1000 rules (100 $iterations * 10$ $ants$) for each class, approximately 250 of them will have one term in the rule antecedent, 250 will have two terms, 250 will have three terms, and 250 will have four. The number of unique rules that may be created using r attributes for this dataset is given by

$$^{n}C_{r} \times 6^{r} \tag{6.3}$$

where $^{n}C_{r}$ is the number of unique combinations of r attributes from a total of n (i.e. the order of

attribute-values in a rule antecedent is unimportant), and 6 is due to the fact that each attribute in this dataset has domain size of six (*low*, *medium*, *high*, *not_low*, *not_medium*, *not_high*)[4]. For the Iris dataset $n = 4$ and r ranges from 1 to 4 so that there are 24 unique rules that may be created for each class using one term in the rule antecedent, 216 rules with two terms, 864 with three terms, and 1,296 with four terms (i.e. one from each attribute).

With approximately 250 opportunities each to create rules of a different size, it is clear that the solution space of smaller-sized rules is explored more than that of larger-sized rules. When selecting the best rule for each class, it may therefore be reasonable that it tends to be smaller rather than larger, resulting in a tendency for rulebases induced by *Random FRANTIC* to have a lower complexity than those induced by *FRANTIC-simpIRL*.

This also explains what may seem the surprising result of a purely random method generating a rulebase with comparable classification accuracy (94%) for the Iris dataset, when compared with *FRANTIC-simpIRL* and the algorithms utilised in Chapter 8. It appears that for this particular dataset, smaller rules – with one or two terms – have generally greater generalisation power than larger rules (average rule size is 4 $terms/3$ *classes*, Table 6.3(b)). With ample opportunity to create all possible unique one-term rules, and probably almost all of the unique two-term rules for each class, *Random FRANTIC* rulebases for this dataset therefore have a very high classification acccuracy.

In general, though, it is not known in advance whether very short rules will be much more accurate than medium-sized or large rules, and so unless a very large number of rules are generated for each possible rule size, this approach is not a viable one. For datasets with a large number of attributes some way of constraining the search space is required, and this is a role that the construction parameters `minInstPerRule` and `constructionThreshold` perform. It is also as a consequence of these parameters restricting the search space, that the standard error of the mean and standard deviations for the accuracy of *FRANTIC-simpIRL* rulebases are on the whole smaller than those for *Random FRANTIC* rulebases – columns 2, 3 and 5, 6 of Table 6.3(a).

6.4 The Richness of the Hypothesis Language

As described in Section 4.1.1, *FRANTIC*'s rule construction mechanism may be exploited so that an increasingly expressive hypothesis language is used to induce rules. This section investigates the impact of the hypothesis language on the complexity and accuracy of the induced

[4]Formula 6.3 is greatly simplified due to the fact that the domain size of each attribute is equal, as it is for all attributes in all datasets used here, with the exception of two attributes in the Water Treatment dataset that each have two values.

FRANTIC-simpIRL rulebases. Another important aspect of using a richer hypothesis language in this work – that of the computational overhead incurred in constructing richer rules – is discussed in Section 7.3. Note that all experiments in this section (and in others unless explicitly mentioned) are carried out using both the heuristic information and pheromone trails.

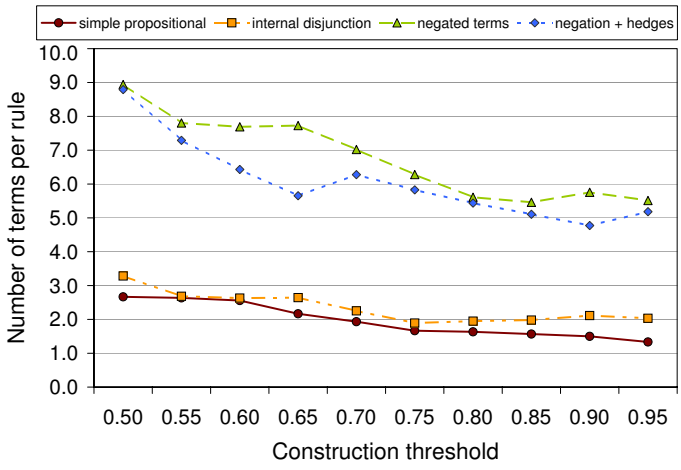
6.4.1 Impact on Model Complexity

Figure 6.5 on the next page gives the average number of terms per rule for *FRANTIC* rulebases induced using simple propositional rules, propositional rules with internal disjunction, rules with negated terms, and rules that include both negated terms and linguistic hedges. Terms combined using internal disjunction are counted separately (for e.g. OUTLOOK=*Cloudy* OR OUTLOOK=*Rain* are counted as two terms), while negated terms are counted as one (for e.g. OUTLOOK=Not-*Sunny* is counted as one). The statistics that generate the graphs in Fig. 6.5, with associated standard error of the mean and standard deviations, are provided in Table C.5 on page 239.

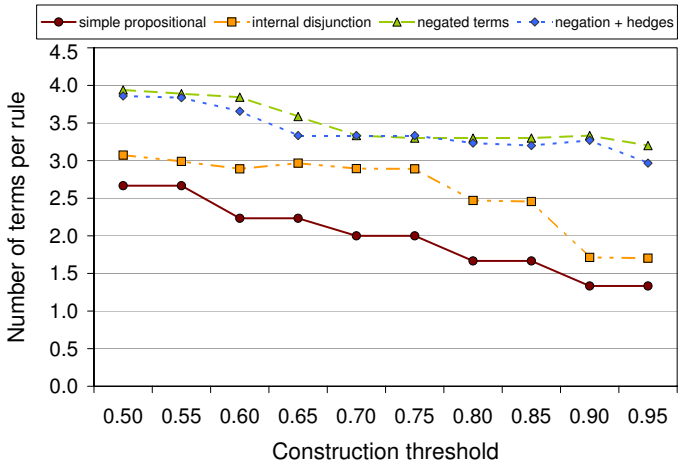
It is clear from Fig. 6.5 that constructing rules from construction graphs that have a larger number of nodes (terms), produces solutions with a greater complexity (the standard error of the mean is in most cases very small) – for each dataset, the curves representing the number of terms for rulebases using negated terms, and those using negated terms and hedges lie higher up on a graph. This is because a larger number of terms in the construction graph is more likely to lead to a larger rule construction term set, with the consequence that more terms may together satisfy the construction parameters criterion. Another possible contributing factor is that negated terms are less specific (e.g. OUTLOOK=*Not-Sunny* is OUTLOOK=*Cloudy* OR *Rain*), so that a larger number of negated terms may be included before rule construction terminates.

Apart from the graph of the Water Treatment dataset (Fig. 6.5(d)), for several or all `constructionThreshold` values the curve indicating rules with negated terms *and* linguistic hedges lies below the one indicating the use of negated terms only. It should be remembered that though the construction graph for inducing rules with both negated terms and hedges has double the number of nodes as the one for inducing rules with negated terms only, the solution validation method allows each type of rule to contain at most one value from the domain of each attribute. The number of terms for these two rule types is therefore dependent on a combination of the constraints imposed by the construction parameters, the element of randomness utilised in rule construction, and the training data itself.

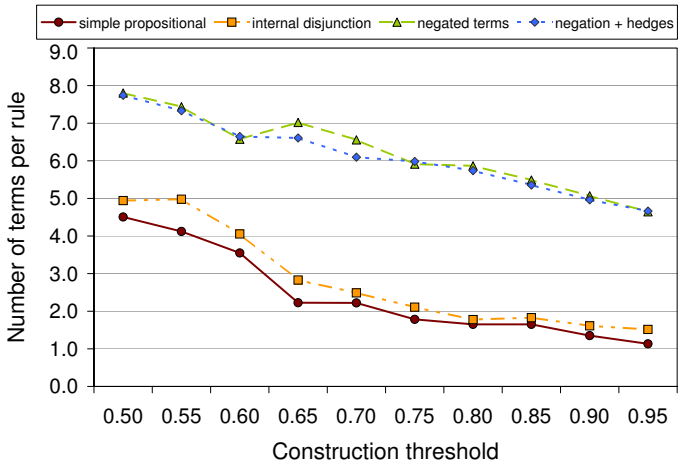
The construction graph for constructing simple propositional rules and rules with internal disjunction contain the same number of nodes. However, the solution validation method allows the latter rule type to retain in its antecedent more than one value from the domain of an attribute,



(a) Wine

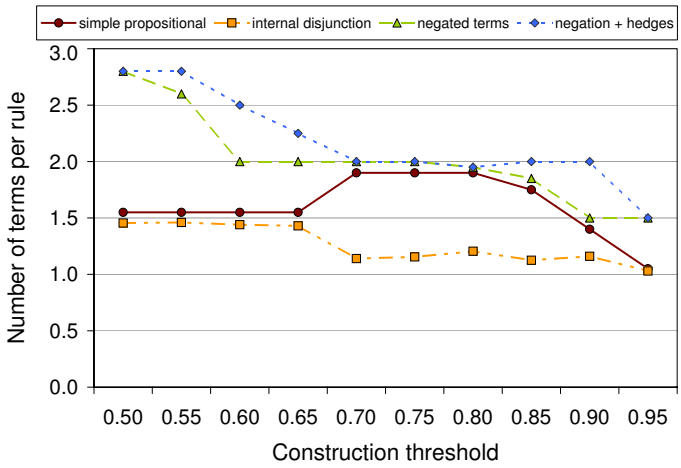


(b) Iris

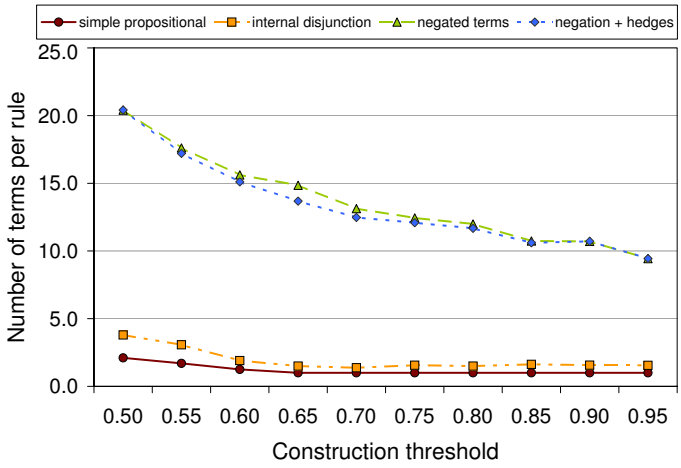


(c) Glass

Figure 6.5: Number of terms per rule in *FRANTIC-simpIRL* induced rulebases using different degrees of richness of the hypothesis language



(d) Water Treatment



(e) Leukaemia

Figure 6.5: Number of terms per rule in *FRANTIC-simpIRL* induced rulebases using different degrees of richness of the hypothesis language (cont.)

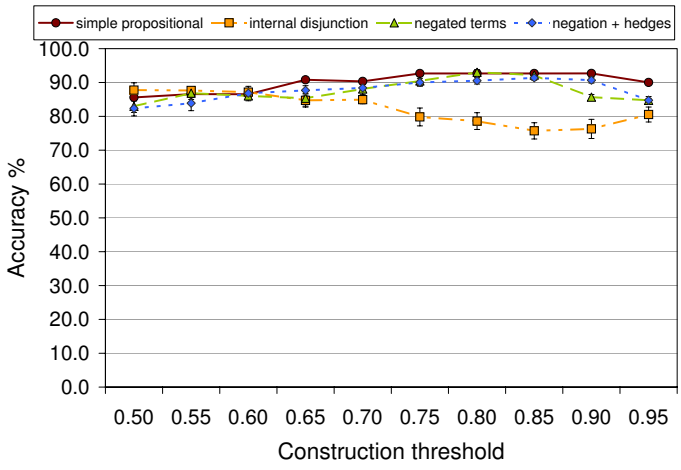
with the interpretation being that of internal disjunction for values from the same attribute. It is therefore quite possible that propositional rules with internal disjunction in general contain more terms in their antecedent than simple propositional rules. This is supported by the graphs for almost all the datasets in Fig. 6.5, apart from that of the Water Treatment dataset (Fig. 6.5(d)). This is likely to be due to the construction method for rules with internal disjunction – if all values from the domain of an attribute are added to a rule antecedent (which can happen in practice), at the end of the rule construction process they are removed, the interpretation being that this attribute provides little discrimination for the class of the rule in question. For certain datasets, this may result in propositional rules with internal disjunction actually being smaller than simple propositional rules.

6.4.2 Impact on Classification Accuracy

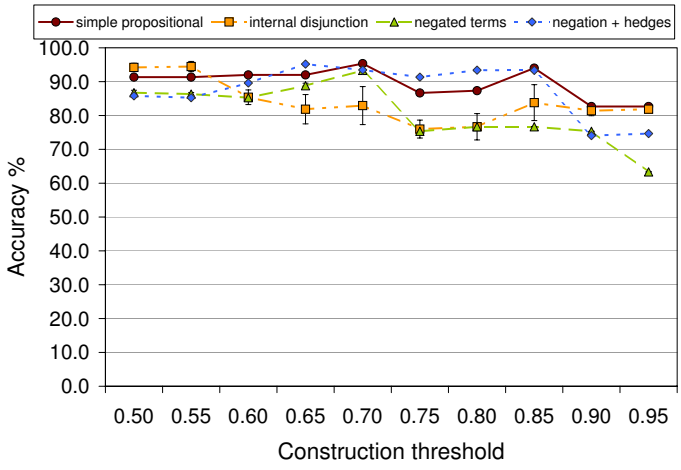
It should be observed that a large difference in the number of terms of a rulebase, as a result of using different degrees of richness in the hypothesis language, need not result in a similarly large difference in the accuracy of such rulebases. This is often the case for lower values of `constructionThreshold`, where rulebases induced using a simpler knowledge representation language may achieve a similar or even superior classification accuracy than rulebases induced using a richer form of the language. Figure 6.6 on the next page shows the accuracy of rulebases induced on all datasets using the different types of IF-THEN rules, and the error bars denote the standard error of the mean. The statistics that generate the graphs in Fig. 6.6 on the following page are provided in Table C.6 on page 241.

Smaller values of `constructionThreshold` lead to larger rule construction term sets, which in turn provides ants constructing rule antecedents with an opportunity to construct more rules, some of which may be quite accurate irrespective of the simplicity of the form of the hypothesis language. In some cases – `constructionThreshold=0.5-0.55` for the Wine, Iris, Water Treatment and Leukaemia datasets – rulebases consisting of simple propositional rules may achieve an accuracy exceeding that of rulebases consisting of rules with negated terms, or with negated terms and linguistic hedges. The greatest difference in the accuracies of rulebases consisting of different rule types appears for `constructionThreshold`$\geq$`0.65`. Beyond this value, richer forms of the hypothesis language are more likely to generate more accurate rulebases, since the construction set for the simpler forms may simply be too small to create a variety of different rules.

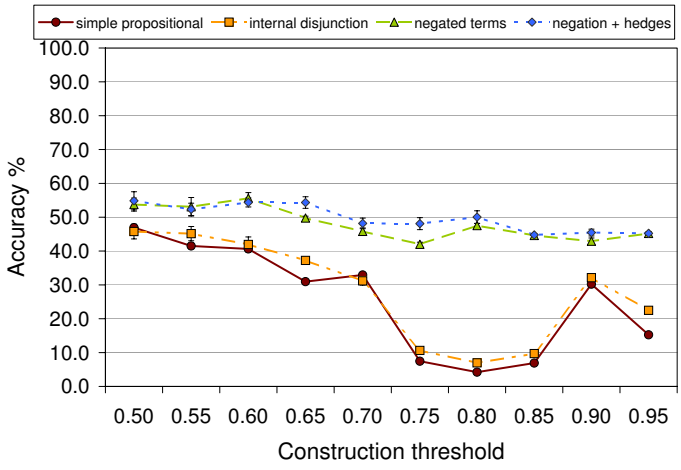
Overall, therefore, it appears that rulebases consisting of IF-THEN rules that include negated terms, or negated terms and linguistic hedges, are more robust to changes in `constructionThreshold` – the accuracies achieved by these rule types vary less from one `constructionThreshold` to another, than rulebases consisting of simple propositional rules.



(a) Wine

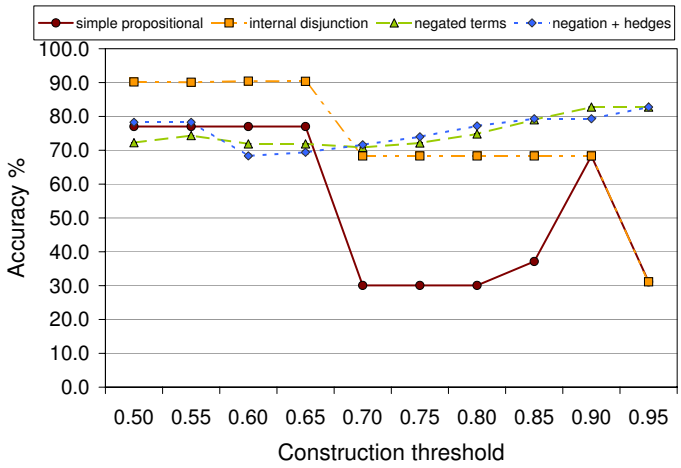


(b) Iris

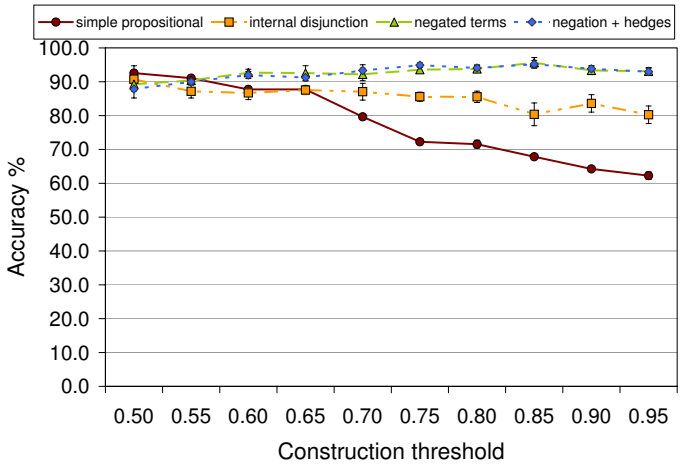


(c) Glass

Figure 6.6: Accuracy of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language. Error bars denote standard error of the mean.



(d) Water Treatment



(e) Leukaemia

Figure 6.6: Accuracy of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language. Error bars denote standard error of the mean (cont.)

`Threshold` value to another, than those achieved by simple propositional rules, or rules with internal disjunction. This is attributed in part to the size of the construction set, and in part to the greater precision of terms contained within the set.

Note that though a term may not be represented to a sufficiently high degree in enough instances in the training set to be included in the construction set for creating simple propositional rules, its negated complement may well be included in the construction set for creating rules with negated terms, or rules with negated terms and linguistic hedges. Therefore, despite higher values of `constructionThreshold` causing the size of corresponding construction sets to become smaller, the sets are still sufficiently large to enable ants to create a variety of rules, some of which can adequately describe the underlying data.

A refinement to the above observation is that rules with both negated terms and linguistic hedges are also likely to be slightly more accurate than those with just negated terms (observed in all graphs in Fig. 6.6). This is due to the fact that linguistic hedges are able to decrease or increase the precision of fuzzy terms, thus allowing ants to describe more precisely the underlying patterns in a dataset.

Some anomalies are also worthy of note. Rulebases induced from the Wine dataset consisting of simple propositional rules often achieve a higher classification accuracy than those induced using even negated terms and linguistic hedges (Fig. 6.6(a)). This may be partly due to this dataset being considered a simple problem with linearly separable classes – simple rules are therefore just as capable of distinguishing between the classes as more elaborate ones. Consequently, if simple rules are sufficient, then the observation that rulebases consisting of simple rules may achieve even *better* accuracy on this dataset is attributed to the likelihood that the construction set is smaller – ants therefore have a greater opportunity to explore and construct more of the different possible rules, enabling the system to choose the best ones. This is supported by observing that the standard deviation for the accuracy of rulebases consisting of simple propositional rules tends to be smaller than that for those consisting of rules with negated terms, or negated terms and hedges (Table C.6).

The Water Treatment dataset also presents interesting results. Figure 6.6(d) shows that for `constructionThreshold`$\leq$`0.65`, simple propositional rules and rules with internal disjunction again achieve a classification accuracy greater than or comparable with that achieved by rules with negated terms and hedges. However, for rulebases consisting of simple propositional rules, Table C.6(d) on page 242 shows that the standard deviation for almost all `constructionThreshold` values is uncommonly large, ranging from approximately 13 to 40.

Analysis of detailed results for the Water Treatment dataset shows that for these `constructionThreshold` values, models are produced that achieve reasonable to high classification accuracy

Table 6.4: Water Treatment dataset – classification accuracy per fold of a ten-fold cross-validation run for *FRANTIC-simpIRL* rulebases induced using different degrees of richness in the hypothesis language. 'Simple' denotes rulebases containing simple propositional rules, 'Disjunction' denotes those containing rules with internal disjunction between attribute values, 'Negation' denotes rules containing negated terms, and 'Hedges' denotes rules that contain both negated terms and hedges.

(a) `constructionThreshold=0.55`

Fold	Simple	Disjunction	Negation	Hedges
1	92.31	89.74	66.67	79.49
2	0.00	55.26	68.42	76.32
3	68.42	81.58	68.42	73.68
4	76.32	94.74	68.42	78.95
5	81.58	94.74	65.79	73.68
6	89.47	97.37	86.84	84.21
7	91.89	100.00	83.78	81.08
8	100.00	100.00	86.49	89.19
9	91.89	100.00	75.68	78.38
10	78.38	89.19	72.97	72.97
Average	77.03	90.26	74.35	78.80
Std dev	28.7	13.6	8.4	5.1

(b) `constructionThreshold=0.70`

Fold	Simple	Disjunction	Negation	Hedges
1	71.79	87.18	66.67	66.67
2	0.00	55.26	60.53	60.53
3	68.42	73.68	71.05	71.05
4	76.32	78.95	73.68	73.68
5	0.00	52.63	68.42	76.32
6	84.21	86.84	81.58	81.58
7	0.00	67.57	75.68	75.68
8	0.00	62.16	78.38	78.38
9	0.00	62.16	67.57	67.57
10	0.00	56.76	64.86	64.86
Average	30.07	68.32	70.84	71.63
Std dev	39.0	12.8	6.5	6.6

(68%-100%) on all folds *except* the second one – rulebases induced using this fold, containing simple propositional rules, generally achieve 0% accuracy. Table 6.4(a) on the previous page shows the results of a ten-fold cross-validation run for *FRANTIC* rulebases induced with the different types of rules, and `constructionThreshold=0.55`. Rules with internal disjunction perform better on Fold 2 than simple propositional rules, rules with negated terms achieve an even better accuracy, and those that also include linguistic hedges peform the best. However, it is also interesting to note that for all other folds, simple propositional rules and/or rules with internal disjunction achieve the higher accuracies.

The picture changes for `constructionThreshold`$\geq$`0.70`, with increasing accuracy for rulebases composed of the more descriptive rules, and decreasing accuracy for those composed of the simpler ones (Fig. 6.6(d)). Table 6.4(b) shows the results of a ten-fold cross-validation run for *FRANTIC* rulebases induced with the different types of rules and `constructionThreshold=0.70`. Due to the restriction imposed by `constructionThreshold` (and by `minInstPerRule`) on the construction set, rulebases composed of simple propositional rules and those with internal disjunction now perform worse, resulting in a considerably lower average accuracy, whilst those composed of rules with negated terms and negated terms and hedges overall perform similarly to when `constructionThreshold=0.55`.

Detailed investigation of the Leukaemia and Iris results yield similar findings. For the Leukaemia dataset with `constructionThreshold=0.50`, all forms of the hypothesis language perform similarly on all folds except for Fold 8, where rulebases composed of simple propositional rules or those with internal disjunction achieve 83% accuracy, versus those composed of rules with negated terms or negated terms and hedges which achieve only 50% accuracy. Again, for greater values of `constructionThreshold`, the larger construction sets and more precise terms enable the system to induce better rulebases. For the Iris dataset, different forms of the hypothesis language yield different results depending on the fold and `constructionThreshold` value.

It seems clear, therefore, that *FRANTIC*'s performance is dependent on the environment in which it operates, as defined by the particular dataset instances in a fold, and the system settings. The use of a more descriptive form of the hypothesis language renders the system more robust to changes over different training sets, and to changes for the `constructionThreshold` parameter. However, it may well be possible to generate more accurate rulebases using more simple forms of the hypothesis language. How this knowledge about *FRANTIC*'s performance may be used to allow the system to adapt to its learning environment is discussed in Section 9.2.1.

6.5 Summary

This chapter has explored the impact of different elements of the rule construction process on the complexity and accuracy of the rulebases induced, and the speed at which optimal rulebases are found. Though the focus of the experiments analysed and discussed in this chapter did not include the impact of the number of iterations in an ACO, the convergence results presented do indicate that for these datasets at least, convergence occurs reasonably quickly and well before the maximum number of 50, 70 or 100 iterations.

The heuristic information and pheromone trails have little impact on the quality of the rulebases, though the pheromones are essential in helping the system to converge to optimal or near-optimal solutions. A search of the ACO-based literature for research that might confirm or contradict these findings sheds little light. Several studies compare different heuristics or pheromone updating strategies, but only a few attempt a more high-level investigation, and these are generally on non-rule induction problems so that it is difficult to make any direct comparisons. Reference [15], for instance, explores running an ACO-beam search hybrid algorithm for solving open shop scheduling problems. The author observes that there is little difference in the quality of the solutions produced, and attributes the strength of the algorithm to the probabilistic beam search mechanism for constructing solutions. This is similar to the conclusions drawn here.

Work on *Ant-Miner* [148], however, also investigates running the system without the use of pheromones. On one of the six datasets used there is little change in accuracy, and on the remaining five there is an average accuracy decrease of 6% when compared with the accuracies obtained using pheromones. The authors conclude that artificial pheromones are important to the accuracy of *Ant-Miner* induced rulebases. This is different from the conclusion drawn here, but it is difficult at this stage to determine what might lead to these different results since there are important differences between the two systems in both the rule construction mechanism, and the overall strategy. A first step towards narrowing down spurious differences would be to test the two systems on the same datasets under the same conditions, though this is left for future work.

It is also possible that running *FRANTIC* on other datasets may provide different results with regards to the impact of the heuristic and pheromones – for the current datasets a small number of iterations is sufficient to produce accurate rulebases with or without the use of these features of the rule construction process. However, for larger and more complex datasets, both in terms of number of instances and number of attributes, it is conceivable that the number of iterations will need to be increased, and then the pheromones (and perhaps the heuristic) will be essential for arriving at optimal solutions as soon as possible. There may also be an additional beneficial

impact on the accuracy of the rulebases induced.

Another avenue for future investigation is the impact of simplified strategy employed (where only one rule is used to describe a class) on the utility of the heuristic information and pheromone trails. It is perhaps reasonable to assume that the first rule generated to describe a class might be the easiest one to construct, since training is carried out on a full training set; subsequent rules to describe the same class are trained on much smaller training sets since instances that are covered by the first generated rule are removed. Therefore, though the impact of the heuristic information and pheromone trails might be minimal on the accuracy of the first generated rules, they may have a greater role to play in a fuller learning strategy where more than one rule is generated to describe a class. Another potential benefit of these two elements is their use in an ACO stopping criterion based on convergence, which might prevent unnecessary computation. This is discussed in Section 7.3.2.

The rule construction parameters and the hypothesis language directly influence the complexity of induced rulebases. Lower values for both `minInstPerRule` and `constructionThreshold` lead to larger rulebases (with regards to the total number of terms in a rulebase), and the use of the richer forms of language also lead to increased complexity.

An increased facility to describe the underlying data need not necessarily be accompanied by an increase in the accuracy of the induced rulebase. Due to their larger construction sets, and the greater precision of the terms these sets may contain, rules with negated terms and those with both negated terms and hedges do achieve a reasonable performance over most `constructionThreshold` values. However, in-depth analyses reveal the influence of other factors, and there is a strong interdependency between the instances within a fold of the dataset, the `constructionThreshold` parameter, and the particular form of the hypothesis language used. It is also probable since there is a strong interaction between `constructionThreshold` and `minInstPerRule`, and since the latter parameter also exerts a direct influence on the size of the construction set, that `minInstPerRule` also has an impact on the final results.

Chapter 7

Inducing a Complete Rulebase

This chapter investigates the impact that the overall induction strategy has on the induced rulebases – the classification accuracy, model complexity, and robustness to changes in several parameters are explored. The advantages and limitations of the different strategies are discussed, as are ways of resolving their current limitations.

Section 7.1 explores the iterative rule learning (IRL) approach and compares the simplified version with the full one. Section 7.2 introduces a simplified form of the simultaneous rule learning (SRL) strategy and compares this against the simplified iterative learning strategy. Section 7.3 discusses the computational complexity of the strategies presented, compares them, and identifies several ways in which the computational performance may be improved for future *FRANTIC* enhancements and implementations.

In all tables and figures in this chapter *simpIRL* denotes results produced by *FRANTIC* running in simplified IRL mode, *fullIRL* denotes results obtained by running in full IRL mode, and *simpSRL* denotes those obtained by running in SRL mode. The *FRANTIC* parameter settings used to generate the results presented are given in Table 7.1 on the following page for full IRL mode, and in Table 7.6 on page 140 for simplified SRL modes. Parameter settings for simplified IRL mode are presented in Chapter 6, Table 6.1.

As for the results discussed in Chapter 6, in these experiments the `constructionThreshold` parameter is always set to the values in the range $[0.50, 0.95]$, with a step value of 0.05, and generally the form of the hypothesis language used is rules that contain negated terms, the number of ants is 10, and the `minInstPerRule` setting for each dataset is as indicated in the tables. The value for `minInstPerRule` is generally dataset dependent, though it appears to be less so when running in full IRL mode (at least for the datasets used here). When parameter settings are varied in some experiments, the additional values used are indicated in the brackets.

Table 7.1: *FRANTIC-fullIRL* parameter settings for exploring the system's performance

	Wine	Iris	Glass	WT	Leuk
`representation`	— Negation (Simple, Disjunction, Hedges) —				
`constructionThreshold`	— 0.50, 0.55, ..., 0.90, 0.95 —				
`minInstPerRule`	(30%,40%) 50%	(40%) 50% (60%)	(30%,40%) 50%	50% (60%,70%)	40% (50%,60%)
`numAnts`	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10
`numIterations`	50	100	50	100	70
`fitnessThreshold`	0.5	0.5	0.5	0.5	0.5
`removalThreshold`	0.5	0.5	0.5	0.5	0.5
`maxClassInstUncovered`	10%	10%	10%	10%	10%

Some parameter settings for simplified and full IRL are determined after running preliminary experiments and this is detailed in Section 5.5; other parameters are used at a default value. The parameter settings for running in simplified SRL mode are, however, in no way optimised – SRL is run using the same settings as for simplified IRL, with the exception of `numIteration` – since simplified SRL conducts more evaluations than simplified IRL, and in order to make the comparison between the two approaches more equitable, `numIterations` is reduced to 30 for simplified SRL.

Error bars on graphs depicting the accuracy of induced rulebases denote the standard error of the mean (SEM). Error bars are not generally included in graphs illustrating model complexity (since they tend to be very small), but the SEM is included in tables that detail the figures giving rise to the graphs. Tables also generally include the standard deviation (STD). Since 10 ten-fold cross-validations are carried out, the STD is the average of the 10 standard deviations obtained from each of the 10 cross-validations (this gives an indication of *FRANTIC*'s robustness to the different training sets used in *one* ten-fold cross-validation, though the inherent randomness in the algorithm will also have some impact), and the SEM is the standard deviation of the 10 average accuracies from each of the 10 cross-validations (this gives an estimate of the reliability of the main statistic presented in a graph or table).

7.1 Iterative Rule Learning Performance

This section investigates the two variants of *FRANTIC*'s iterative rule learning mode – the simplified version where only one rule is used to describe each class, and the full version where more than one rule may be generated for a class. The two variants are compared with regards to their impact on model accuracy and complexity – Sections 7.1.1 and 7.1.2 respectively.

The number of extra rules created in the full version is determined by several user-defined system settings – the rule construction parameters `minInstPerRule` and `constructionThreshold` (relevant for both simplified and full IRL), and the parameters `maxClassInstUncovered`, `fit-`

Table 7.2: Comparison of *FRANTIC* rulebases induced using the simplified (*simpIRL*) and full (*fullIRL*) modes of the iterative rule learning strategy

(a) Classification accuracy (%) and standard error of the mean (+/-)

	simpIRL		*fullIRL*	
	%	+/-	%	+/-
Wine	93.01	0.77	93.89	1.22
Iris	95.33	0.00	95.33	0.00
Glass	55.63	1.64	58.41	2.21
WT	90.42	0.62	93.07	0.00
Leuk	95.63	1.55	95.07	1.72

(b) Number of rules and number of terms in a rule

	simpIRL		*fullIRL*	
	Rules	Terms	Rules	Terms
Wine	3.00	5.61	6.00	7.58
Iris	3.00	2.00	3.40	2.03
Glass	6.00	6.58	12.00	7.67
WT	2.00	1.44	4.00	3.23
Leuk	2.00	10.74	2.82	3.65

`nessThreshold` and `removalThreshold` (necessary only for full IRL). As explained in Section 5.5, *FRANTIC* appears relatively robust to many parameters, so that the focus in this chapter is again mainly on the rule construction parameters. However, the potential of `maxClassInstUncovered` is briefly discussed in Section 8.3 when discussing *FRANTIC*'s performance on the extremely imbalanced Water Treatment dataset, and the possibility of entirely removing or automating the setting of the other two parameters is discussed in Section 4.3.4 on page 69.

Other parameters – `numAnts` and `representation` – are also varied in their settings so that more general conclusions may be drawn with regards to the limitations and advantages of these two variants. Details are provided where appropriate.

7.1.1 Classification Accuracy

This section compares the two versions of the IRL strategy with regards to classification accuracy. The influence that additional rules may have on accuracy is explored from different angles – the impact on the different forms of the hypothesis language, and how the results vary over different values for `constructionThreshold`. First, a summary of the results for the two variants is presented.

Table 7.2 shows statistics for the rulebases achieving the highest accuracies induced using the two different IRL variants – Table 7.2(a) gives the average accuracy (and standard error of the mean), and Table 7.2(b) the corresponding average number of rules in a rulebase, and the average number of terms per rule. Simplified IRL accuracy statistics are selected from those presented in Tables C.6 and C.4, with corresponding rulebase complexity statistics in Tables C.5 and C.3. The equivalent statistics for full IRL induced rulebases are found in Tables D.1 and

D.4 for accuracy details, and in Tables **??** and **??** for complexity details.

Rulebases induced using full IRL for the Wine, Glass and Water Treatment datasets achieve a higher classification accuracy than those induced using simplified IRL (with a corresponding increase in the number of rules and/or terms in a rule), rulebases induced using either method on the Iris dataset achieve the same classification accuracy (with a similar model complexity), while those induced using simplified IRL for the Leukaemia dataset achieve the greater accuracy (again with a corresponding increase in model complexity). Though on the whole extra rules in a rulebase appear to provide some benefit with regards to classification accuracy, the extent of this benefit is not clear, and other factors are taken into account in order to exactly determine the advantages offered by full IRL, and clarify where and when it fails when compared with the simplified version.

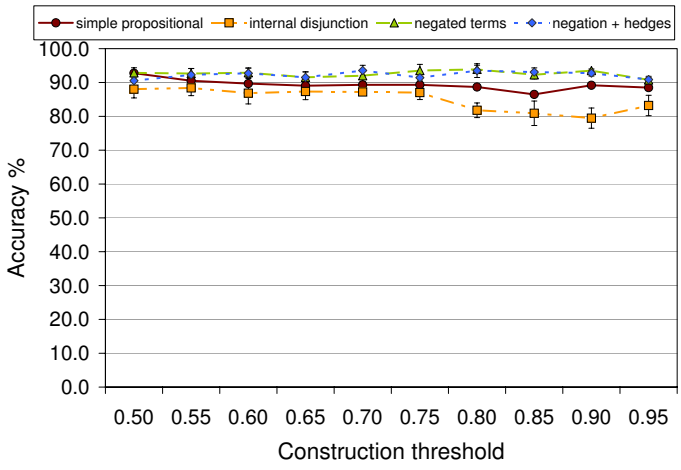
Increased robustness to `constructionThreshold` and `representation`

Figure 7.1 on the following page presents the classification accuracy achieved by each form of the hypothesis language for each dataset, for *FRANTIC-fullIRL* induced rulebases. When compared with the equivalent figure showing the results for *FRANTIC-simpIRL* induced rulebases (Fig. 6.6 on page 108), several observations may be made. The additional rules that may be induced using a full IRL approach render the system more robust to changes in two parameters:

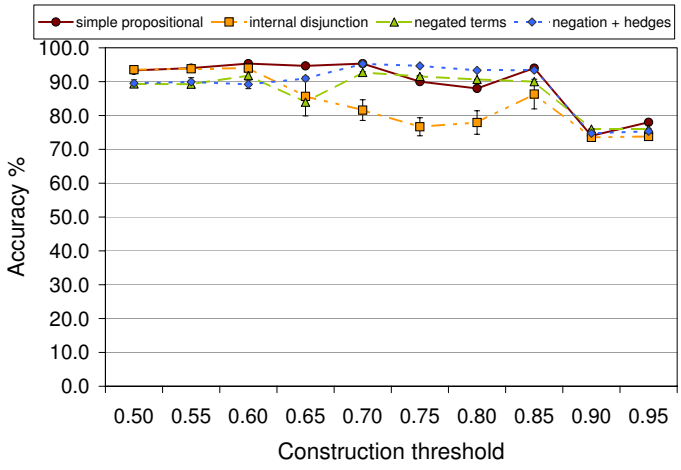
- in Fig. 7.1 the curves representing the different types of rules are closer together than those in Fig. 6.6 – there is less of a difference between the performance of rulebases induced using different forms of the hypothesis language;
- the points on most curves in Fig. 7.1 lie in a smaller vertical portion of the graph – there is less variance in the accuracies achieved for each form of the hypothesis language for the different `constructionThreshold` values.

The first point suggests that extra rules to describe a class somewhat make up for a less descriptive and/or precise form of the hypothesis language – when running *FRANTIC* in full IRL mode, rulebases induced with simple propositional rules or rules with internal disjunction achieve accuracies closer to those achieved by rulebases composed of rules with negated terms, or rules with negated terms and linguistic hedges.

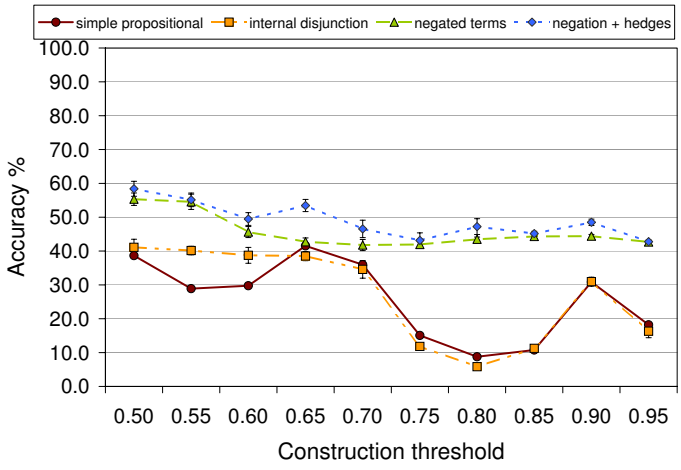
The box-and-whiskers plots in Fig. 7.2 on page 123 provide more evidence to support the second observation – they compare the level of dispersion in accuracies of simplified IRL and full IRL induced rulebases. Tables C.6 and D.1 are used to generate these plots, and they present the accuracies of the induced rulebases using different forms of the hypothesis language. The plots provide a measure of the robustness of these induced rulebases to changes in settings for



(a) Wine

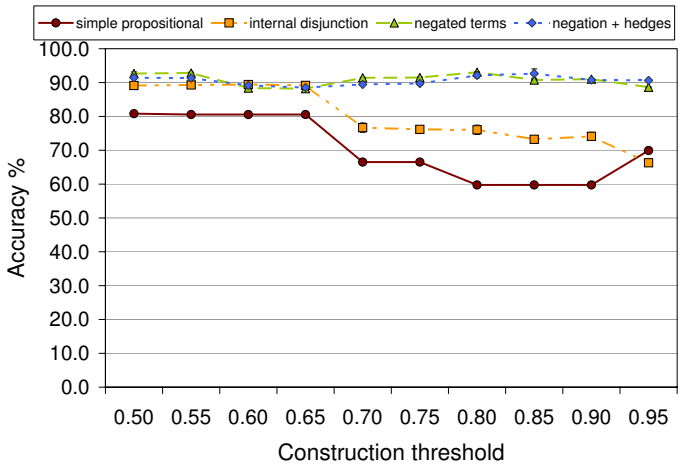


(b) Iris

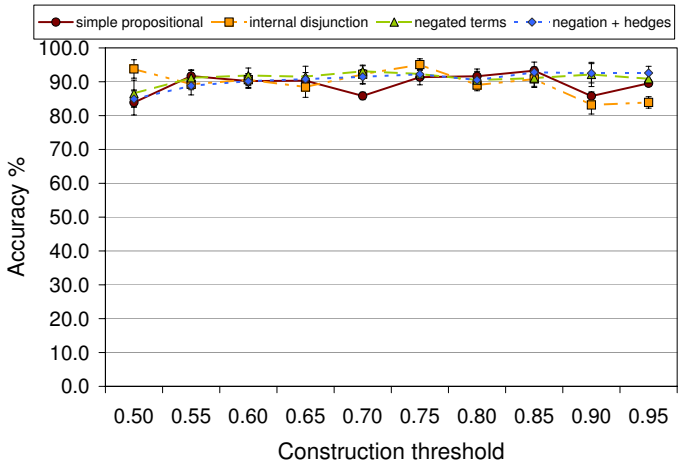


(c) Glass

Figure 7.1: Accuracy of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language



(d) Water Treatment



(e) Leukaemia

Figure 7.1: Accuracy of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

Table 7.3: *FRANTIC* induced rulebases for the Saturday Morning Problem dataset – Rulebase *A* induced using simplified IRL: R1-R3, 93.75% accuracy; Rulebase *B* induced using full IRL: 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*

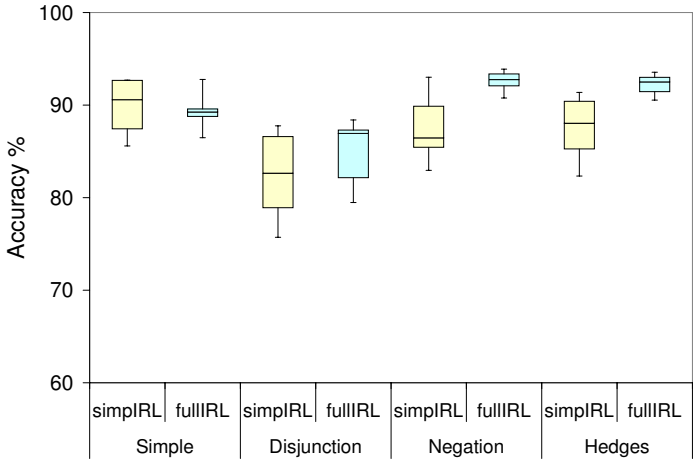
the `constructionThreshold` parameter – the top and bottom whiskers indicate respectively the maximum and minimum average accuracy achieved by rulebases for `constructionThreshold` values in the range [0.50,0.95] (and hence the overall range of accuracies), and the box indicates the range in which 50% of the accuracies lie. The smaller the overall range and the smaller the box, the more robustness to changes in this parameter is exhibited. With a few exceptions there is less variance in the accuracy of rulebases induced using full IRL, than there is for the accuracies of those induced using simplified IRL – the full range of accuracies is often smaller for full IRL rulebases, and generally, 50% of the accuracies attained lie in a smaller range.

Competing fuzzy rules

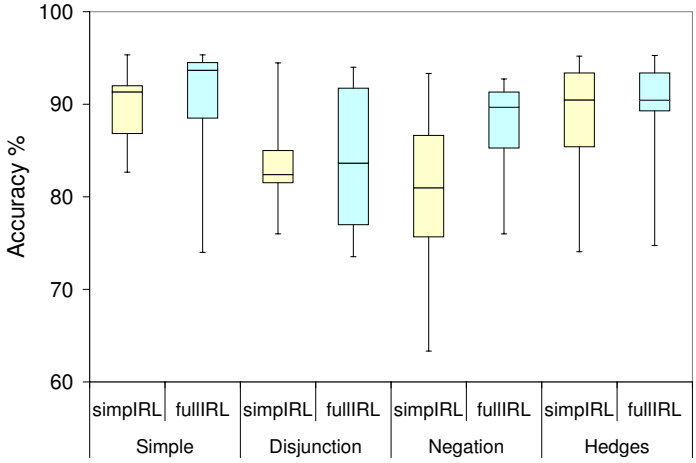
What is also made obvious by the boxplots in Fig. 7.2 on page 123 is that it is not always a full IRL induced rulebase that achieves the greatest accuracy for each different form of the hypothesis language. This is thought to be due mainly to the nature of fuzzy rules and how they interact during classification. A simple example illustrates how this potential problem often arises by inducing rules via an iterative rule learning approach.

Table 7.3 lists the rules induced by *FRANTIC* when running in IRL mode on the Saturday Morning Problem dataset. This dataset is a small artificial one originally used in the induction of crisp decision trees [157]. It has four condition attributes and sixteen instances, each classified as to which sport should be played on a Saturday morning depending on the weather conditions. It has been extensively used by both crisp and fuzzy induction algorithms when introducing new machine learning techniques. Due to its small size it has not been used in this work for comparison with other algorithms in Chapter 8, but the rulebases induced from this dataset still serve to highlight a fundamental point in fuzzy inferencing.

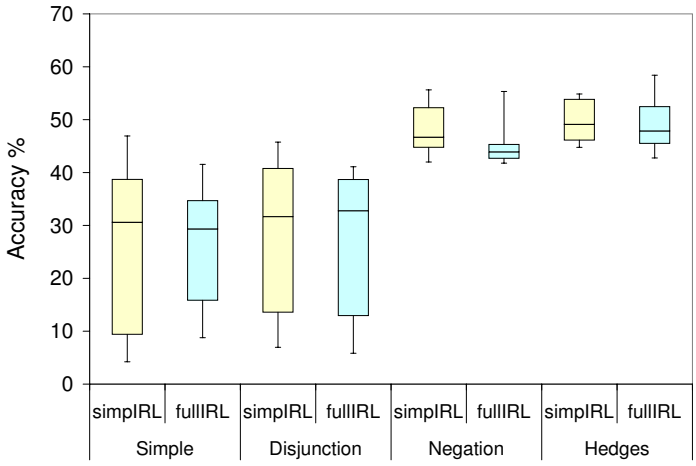
Rulebase *A* which consists of rules R1–R3 in Table 7.3 is one of the rulebases commonly



(a) Wine

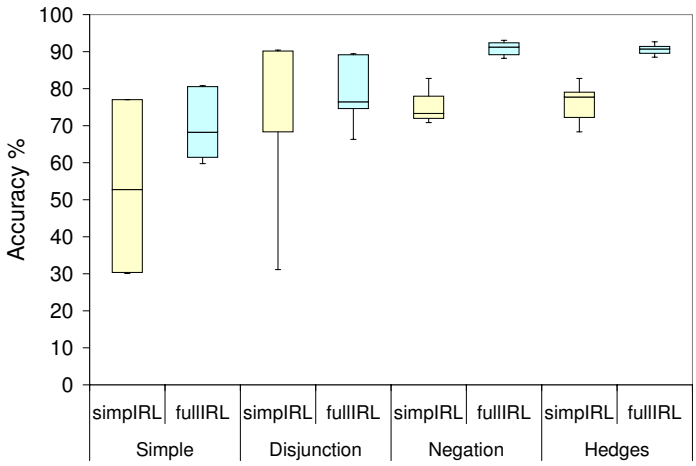


(b) Iris

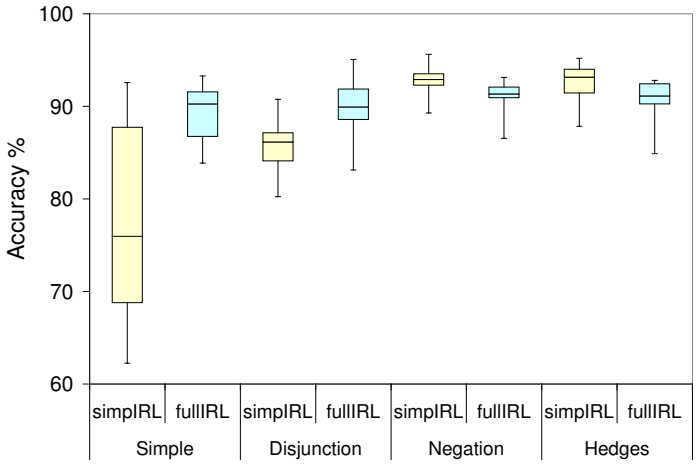


(c) Glass

Figure 7.2: Dispersion of accuracies achieved by *FRANTIC* simplified and full IRL rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote full IRL induced rulebases.



(d) Water Treatment



(e) Leukaemia

Figure 7.2: Dispersion of accuracies achieved by *FRANTIC* simplified and full IRL rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote full IRL induced rulebases. (cont.)

Table 7.4: Fuzzy rule interaction and classification

Inst.	Actual	Degree of match				Classification	
ID	Class	R1(VB)	R2(SW)	R3(WL)	R4(WL)	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*

produced when *FRANTIC* is run in simplified IRL mode, achieving an accuracy of 93.75% on the dataset. Rulebase *B* consists of rules R1–R4 and is often induced when *FRANTIC* is run in full IRL mode, achieving an accuracy of 87.50%.

Table 7.4 highlights 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 7.3 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 7.4 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.

Consider now only the instances that actually describe *Weightlifting* (instances in Table 7.4 with 'WL' in column 2) – Rulebase *B* is a closer match to the data than Rulebase *A*, since the additional rule R4 describing *Weightlifting* 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.

The example above is an extreme – though real – one. The same thing is observed when looking

at detailed results for the Glass dataset, for instance. Generally, the `minInstPerRule` value chosen from the exploratory runs described in Section 5.5 are different for running *FRANTIC* in these two modes. However, for the Glass dataset 50% appears to provide higher accuracies for both modes. The same six rules are therefore often induced for the two variants, with an additional two – one extra for the first two classes – for full IRL. These extra two rules increase the degrees of membership for several test instances whose actual class labels may or may not be the same class as the rules. The degree of membership for non-same class instances are sometimes increased to an extent that they are misclassified by one of the extra rules, causing the overall accuracy of full IRL-induced rulebases to be less than that achieved by simplified IRL-induced ones. For the Leukaemia dataset, the same rules are not induced by the simplified and full IRL modes, but again, extra rules induced by the full version often end up competing with rules that describe other classes, and win in the misclassification of non-same class instances.

Though one is able to determine the reason why extra rules may not always result in improved accuracy, it is difficult to generalise these findings over the datasets, i.e. to predict whether one dataset is more likely to require additional rules for some classes than another dataset. For the Wine and Water Treatment datasets there is some benefit obtained from extra rules for all forms of the hypothesis language used. For the Iris dataset this is less obvious though the differences in maximum accuracy achieved are small. However, the Iris dataset has an exactly equal distribution between the classes, the Wine dataset an approximately equal distribution, but the Water Treatment dataset is highly imbalanced. The classes in the Wine dataset are known to be linearly separable, while two classes in the Iris dataset are not. Furthermore, simple clustering techniques used on the Leukaemia dataset suggest that the two classes are also highly separable, yet additional rules for this dataset often lead to inferior average accuracies per `constructionThreshold` value for rules with negated terms, and those with negated terms and linguistic hedges. No consistent patterns, either, may be discerned in the number of instances in a dataset, the number of conditional attributes, or the number of class labels.

It is thought, therefore, that the advantage – or disadvantage – conferred by additional rules is dependent mainly on the environment in which rules are constructed, as defined by the particular form of the hypothesis language used, the settings for `minInstPerRule` and `constructionThreshold`, and the probabilistic construction process. A beneficial side effect of the strong influence between these elements is that it appears to make *FRANTIC* – whether operating in simplified or full IRL mode – robust to the inherent characteristics of the datasets, such as the degree of class separability and distribution. This robustness is discussed again in Chapter 8, when comparing *FRANTIC* with other learning algorithms.

7.1.2 Model Complexity

This section investigates the impact that *FRANTIC*'s full IRL strategy has on the number of rules in the final rulebase, and the size of the rule antecedents.

Impact of `minInstPerRule`

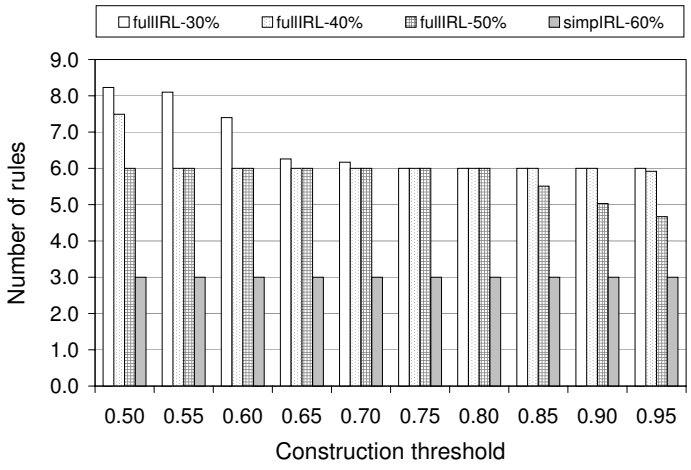
Figure 7.3 on page 128 illustrates the impact of `minInstPerRule` on the number of rules in a rulebase, over a range of `constructionThreshold` values. The number of rules induced using simplified IRL is always equal to the number of classes in a dataset, but is included in the figure as a convenient reference point.

It is clear from Fig. 7.3 that for each `constructionThreshold` value, higher values of `minInstPerRule` generally lead to an equal or smaller average number of rules in a rulebase, than lower values of `minInstPerRule`. This is because the higher the value of `minInstPerRule`, the greater the number of instances a constructed rule must cover. Consequently, in full IRL, in between ACO runs aimed at discovering rules to describe the same class, the greater is the number of instances that are removed from the training set. Hence, if during construction each rule is constrained to cover a larger number of instances for a particular class, then fewer rules are likely to be required to cover all the instances for that class (excluding the proportion set by `maxClassInstUncovered`).

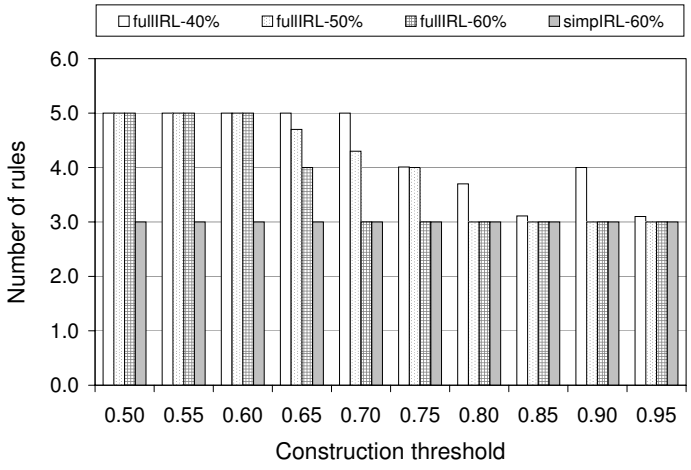
As Section 4.2.1 outlines, the parameters `maxClassInstUncovered` and `removalThreshold` are also highly likely to have a direct impact on the number of rules in a rulebase: for any specific `minInstPerRule` setting, the greater the value of `maxClassInstUncovered` the fewer the rules likely to be necessary to cover the required proportion of class instances; similarly, the lower the value of `removalThreshold`, the greater the number of class instances removed in between ACO runs, leading again to a likelihood of fewer rules necessary to cover the required number of class instances.

Section 6.3 explored the impact of the rule construction parameters on the number of *terms* in a rule (antecedent), though the rulebases investigated there are induced using simplified IRL. Figure 7.4 on page 130 illustrates the average number of terms per rule in rulebases induced using full IRL for different `minInstPerRule` and `constructionThreshold` values. For the Iris, Glass and Water Treatment datasets (Figs. 7.4(b)–7.4(d)), the results are very similar to those shown in Figs. 6.4(b)–6.4(d) on page 99 for simplified IRL rulebases – the lower the values for both `minInstPerRule` and `constructionThreshold`, the greater the number of terms in a rule antecedent.

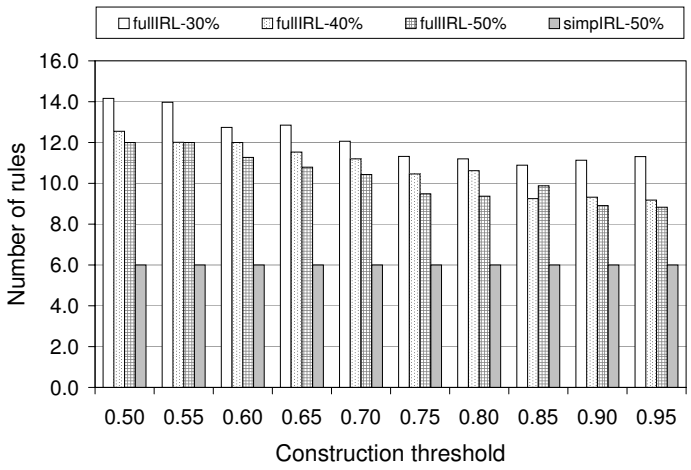
However, this is not the case for full IRL induced rulebases for the Wine and Leukaemia



(a) Wine

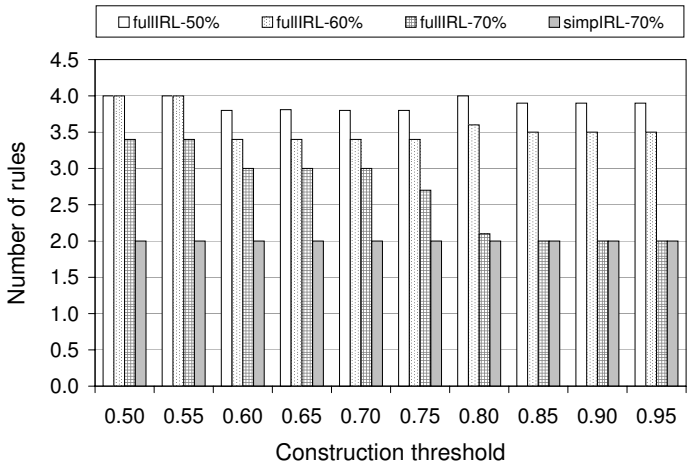


(b) Iris

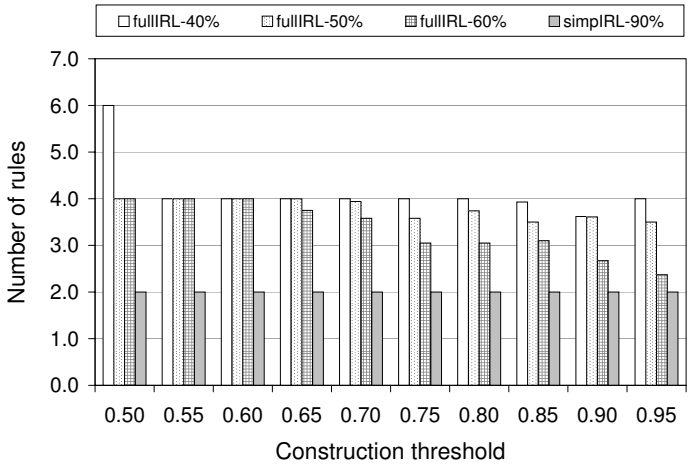


(c) Glass

Figure 7.3: Impact of the `minInstPerRule` parameter on the number of rules in *FRANTIC-fullIRL* induced rulebases



(d) Water Treatment



(e) Leukaemia

Figure 7.3: Impact of the `minInstPerRule` parameter on the number of rules in *FRANTIC-fullIRL* induced rulebases (cont.)

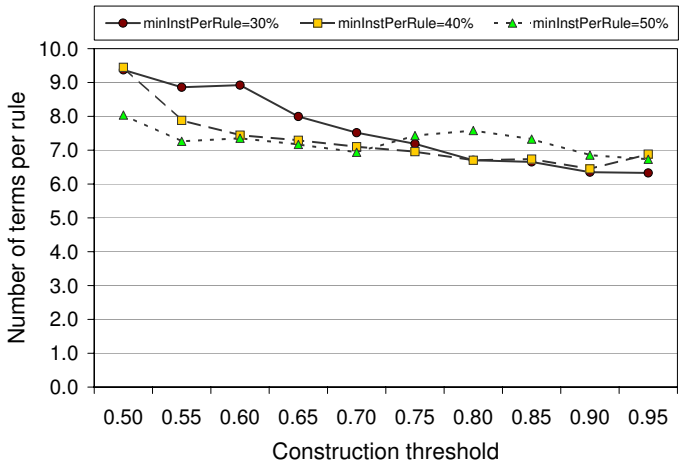
Table 7.5: Change of `minInstPerRule` values in between ACO runs during full IRL

Wine dataset, `constructionThreshold=0.75`		
`minInstPerRule=`	$p\%$ (m)	$p'\%$ (m')
class A	30% (16)	72% (13)
class B	30% (19)	72% (16)
class C	30% (13)	69% (9)
class A	40% (21)	64% (9)
class B	40% (25)	62% (10)
class C	40% (17)	63% (7)
class A	50% (27)	29% (2)
class B	50% (32)	54% (7)
class C	50% (22)	50% (4)

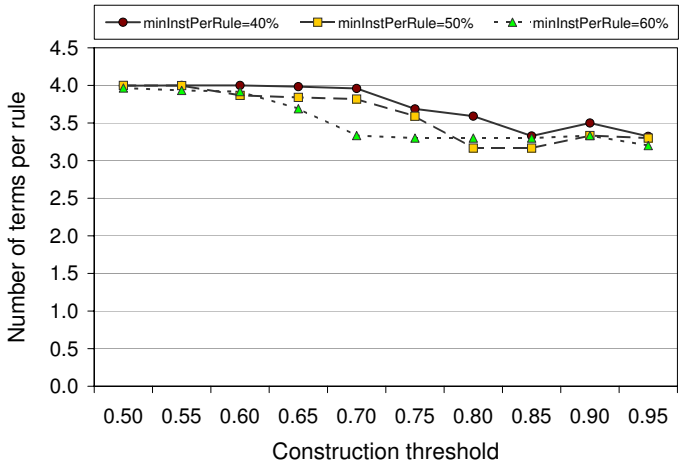
datasets. Figure 7.4(a) depicting the results for the Wine dataset, shows that for `constructionThreshold`$\geq$`0.80`, the average number of terms per rule induced using `minInstPerRule=30%` is actually lower than the numbers for `minInstPerRule=40%` or `minInstPerRule=50%`. Similarly for the Leukaemia dataset (Fig. 7.4(e)), for `constructionThreshold`$\geq$`0.55`.

These anomalies are thought to occur due to the dynamic nature of the `minInstPerRule` parameter. During full IRL, at the beginning of the first ACO run to find a rule to describe a class, the original `minInstPerRule` percentage value, say $p\%$ of the class instances, is turned into an integral value, say m class instances. Once the best rule produced by the first ACO has been determined, and the class instances covered by this rule removed from the training set, the remaining number of class instances may be fewer than the system required total of at least m plus the number specified by `maxClassInstUncovered` (Fig. 4.9 on page 68 line (4)). A new absolute value for `minInstPerRule` is therefore determined according to line (5), say m', and used in the next ACO. m' will be lower in value than the original m instances, but the corresponding new $p'\%$, i.e. the proportion of remaining class instances that m' represents, may be the same as, less or greater than the original $p\%$.

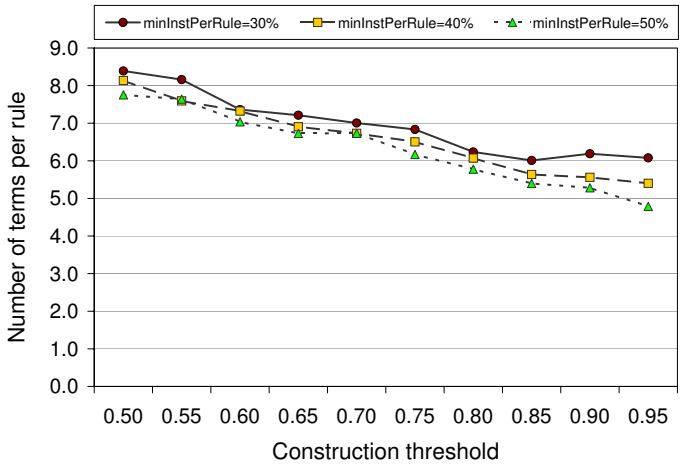
In the particular cases mentioned above for the Wine and Leukaemia datasets, $p'\%$ tended to be such that the balance between the two rule construction parameters changed considerably from its original definition. For instance, Table 7.5 presents detailed results for the Wine dataset gathered during the induction of full IRL rulebases with `constructionThreshold=0.75` and for different `minInstPerRule` values. The first column indicates a specific class, the second its original `minInstPerRule` percentage and absolute values, and the third its new `minInstPerRule` values after training instances covered by the best rule of the first ACO have been removed. *FRANTIC* IRL runs for this dataset that start with `minInstPerRule` and `constructionThreshold` values of 30% and 0.75 respectively, end up running with values of 71% ($p'\%$ averaged



(a) Wine

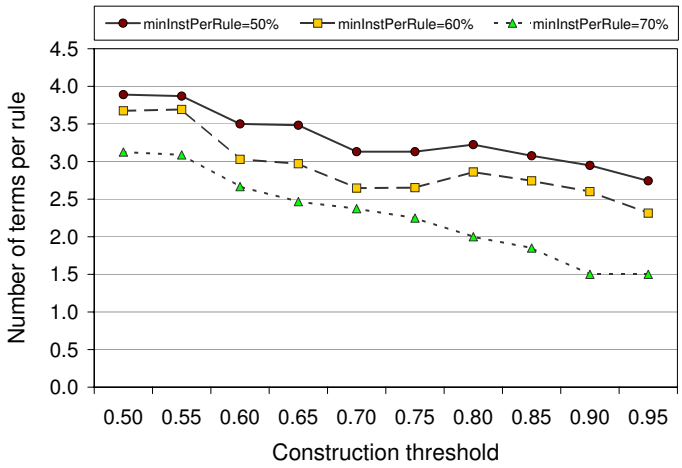


(b) Iris

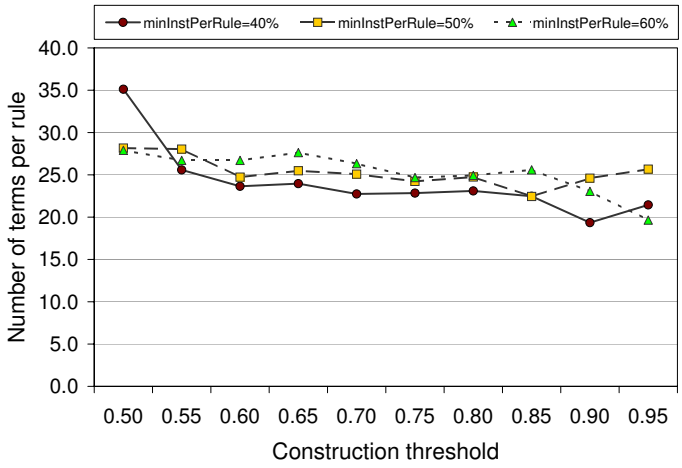


(c) Glass

Figure 7.4: Impact of the `minInstPerRule` parameter on number of terms in a rule in *FRANTIC-fullIRL* induced rulebases



(d) Water Treatment



(e) Leukaemia

Figure 7.4: Impact of the `minInstPerRule` parameter on number of terms in a rule in *FRANTIC-fullIRL* induced rulebases (cont.)

over the three classes) and 0.75, those starting with 40% and 0.75 change to 63% and 0.75, and those with 50% and 0.75 to 44% and 0.75. This is what causes the average number of terms in a rule for rulebases induced using `minInstPerRule=50%` to be greater than those induced with `minInstPerRule=30%`, for instance, when one would normally expect the average value to be lower (Fig. 7.4(a), `constructionThreshold`$\geq$`0.75`).

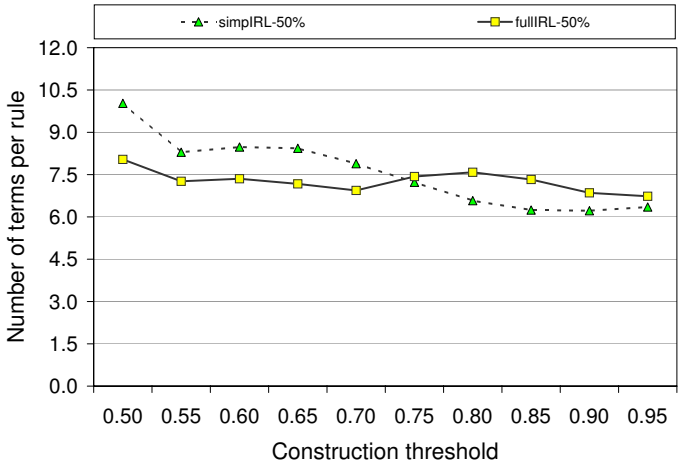
This conclusion is reinforced by looking at the results obtained from *simplified* and *full* IRL induced rulebases induced using the same `minInstPerRule` and `constructionThreshold` settings, Fig. 7.5 on page 134 – the difference for the average number of terms in a rule between simplified and full IRL induced rulebases is small (note the scale on the y-axis for each dataset). However, where the number of terms per rule produced is the same for full IRL as for simplified IRL induced rulebases (Fig. 7.5(b) Iris dataset `constructionThreshold`$\geq$`0.70`, and Fig. 7.5(d) Water Treatment dataset `constructionThreshold`$\geq$`0.85`), and hence when there is definitely no change in `minInstPerRule` values, the average number of terms per rule is practically identical.

The dynamic nature of this parameter also accounts for two further observations that may be made about Fig. 7.3, regarding the number of *rules* in a full IRL induced rulebase. For some datasets and some `constructionThreshold` settings, different values of `minInstPerRule` lead to the same number of rules (e.g. the Wine and Iris datasets), and in one case a higher `minInstPerRule` value leads to a larger number of rules than a lower `minInstPerRule` value (Fig. 7.3(c) Glass dataset, `constructionThreshold=0.85`). Both these may be attributed to a changing `minInstPerRule`, which as Table 7.5 demonstrates, may go down as well as up.

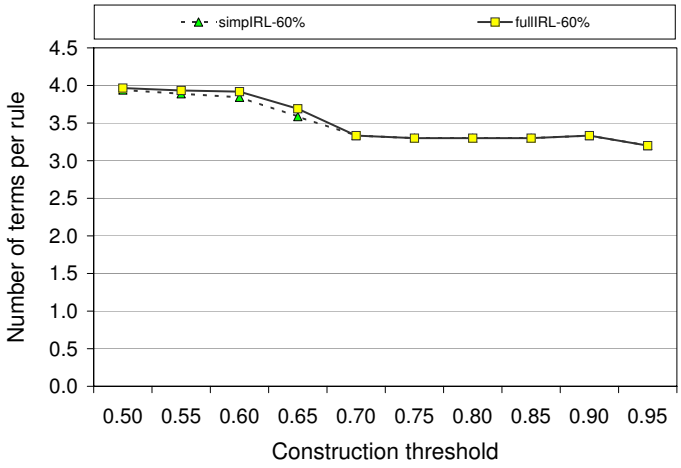
The fact that `minInstPerRule` may increase considerably from its original value, when inducing extra rules to describe a class (e.g. from 30% to 70% in Table 7.5), may also provide a partial explanation as to why extra rules end up competing with rules describing different classes – a form of over-fitting may be occurring during the construction of these additional rules due to the high `minInstPerRule` value. The automatic updating of this parameter may be amended so that its original value remains valid throughout induction. This could be achieved by setting new absolute values based on the original percentage (p%) and the remaining number of class instances in the training set (r), i.e.

$$m' = (p\% \times r) \tag{7.1}$$

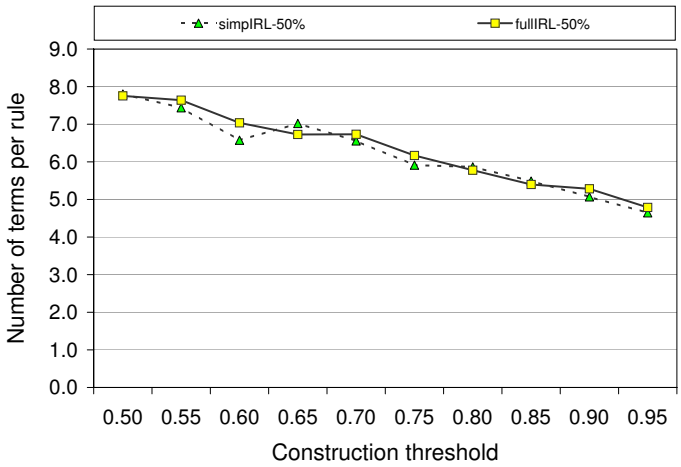
This would maintain a more consistent control over the complexity of the induced rulebase, and may result in improved accuracy. The possibility of exploring this in future work is discussed in Section 9.2.2.



(a) Wine

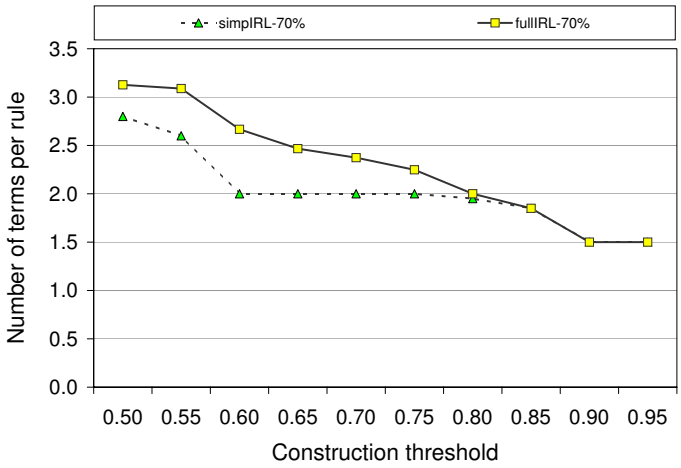


(b) Iris

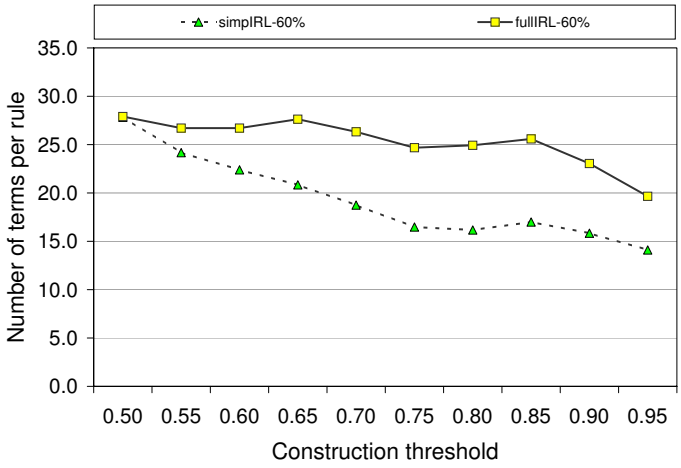


(c) Glass

Figure 7.5: Number of terms per rule in simplified and full IRL induced *FRANTIC* rulebases

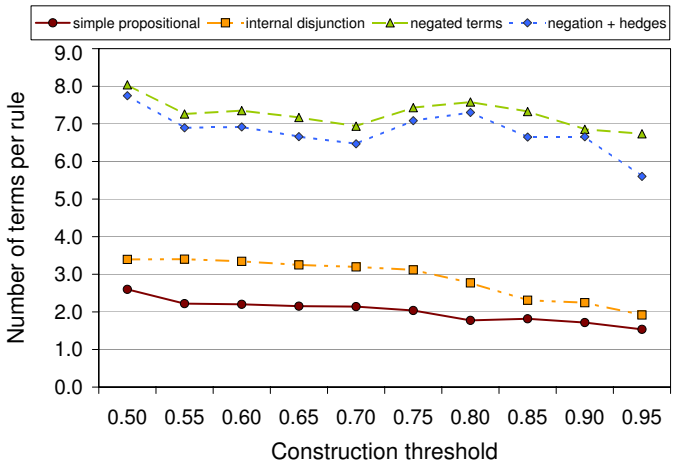


(d) Water Treatment

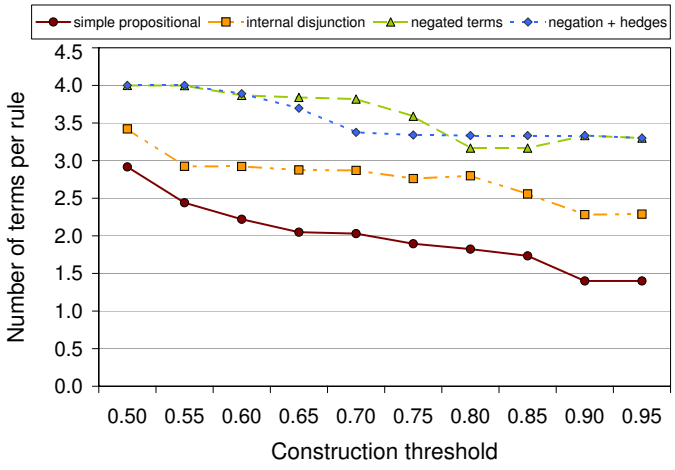


(e) Leukaemia

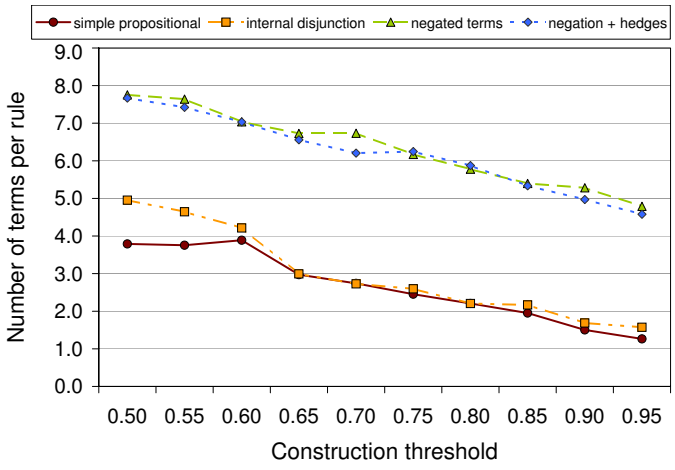
Figure 7.5: Number of terms per rule in simplified and full IRL induced *FRANTIC* rulebases (cont.)



(a) Wine

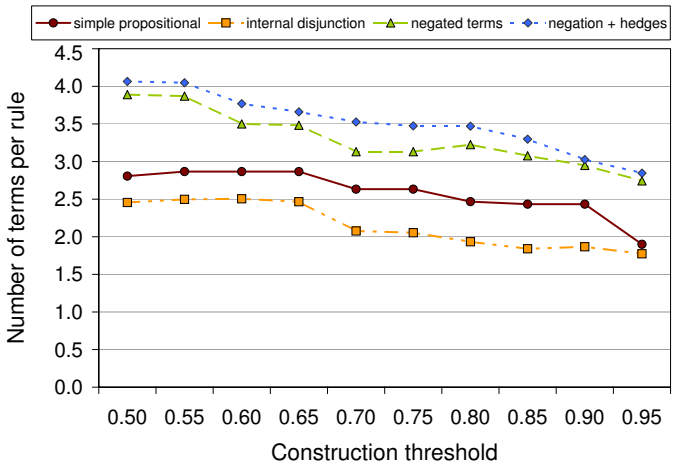


(b) Iris

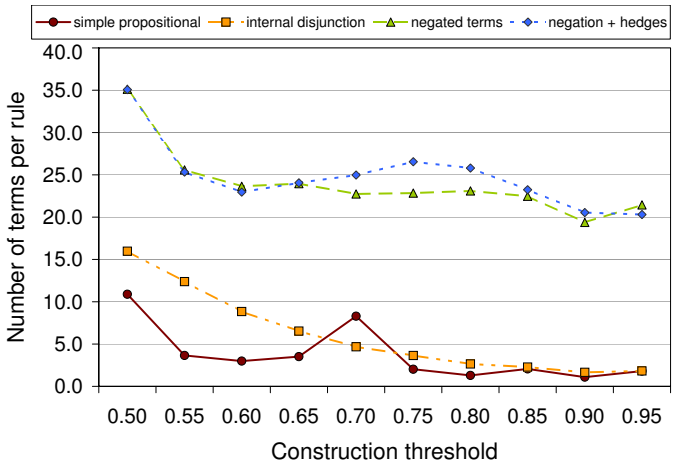


(c) Glass

Figure 7.6: Number of terms per rule in *FRANTIC-fullIRL* induced rulebases using different forms of the hypothesis language



(d) Water Treatment



(e) Leukaemia

Figure 7.6: Number of terms per rule in *FRANTIC-fullIRL* induced rulebases using different forms of the hypothesis language (cont.)

Impact of the Hypothesis Language

Figure 7.6 on page 135 shows the average number of terms per rule for *FRANTIC-fullIRL* induced rulebases using different forms of the hypothesis language. The equivalent figure for *FRANTIC-simpIRL* induced rulebases is Fig. 6.5 on page 105.

It is clear that the form of the hypothesis language used has the same general effect on each of the two modes of the learning strategy, for all the datasets tested. As explained in Section 6.4, rulebases induced using the larger problem graphs as a result of including negated terms and linguistic hedges, generate larger rule antecedents and so their curves lie higher up on the graph of each dataset. And, due to the size of the construction term set that is determined by the settings for `minInstPerRule` and `constructionThreshold`, all forms of the hypothesis language follow a general downward trend in rule antecedent size as the value of `constructionThreshold` increases.

The similarities between the two versions of IRL are observed despite the fact that different `minInstPerRule` values are often used for a dataset during simplified and full IRL (the parameter tends to be set to a lower value for full IRL), and its value often changes between ACO runs during full IRL. This therefore serves to reinforce the conclusions reached in Section 6.4, about the influence of the form of the hypothesis language on rule construction.

Since in full IRL more than one rule may be created to describe a class, it is informative to investigate whether the form of the hypothesis language used also has an impact on the number of *rules* in a rulebase. Figure 7.7 on the next page shows the average number of rules induced for *FRANTIC-fullIRL* rulebases using different forms of the hypothesis language (the graphs relate to the same rulebases whose average terms per rule have been shown in the preceding Fig. 7.6).

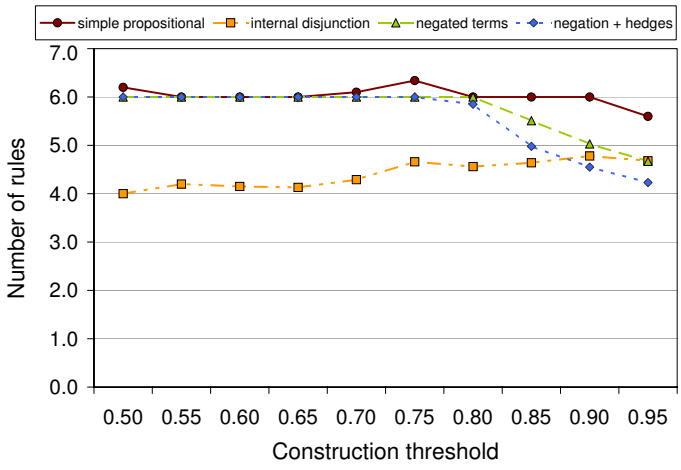
The only clear generalisation that may be made over all datasets is related to rulebases that are made up of propositional rules with internal disjunction between values of the same attribute – they are often smaller in size than rulebases containing rules constructed using other forms of the hypothesis language, and there appears to be a slight upward trend in the rulebase size as `constructionThreshold` increases.

The first observation is attributed directly to the use of internal disjunction between attribute values. For instance, the rule:

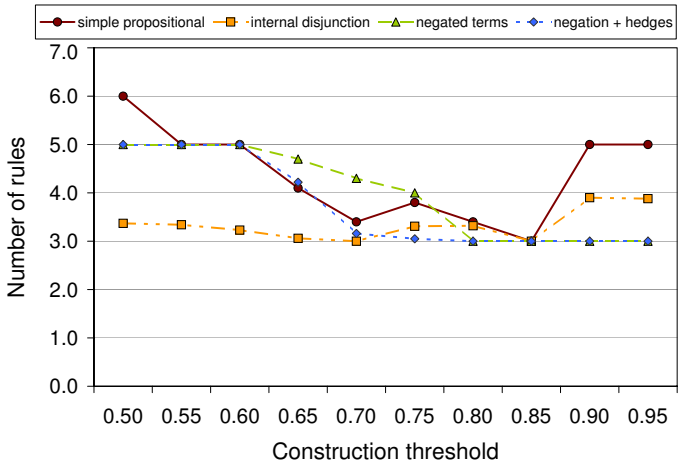
IF OUTLOOK is Cloudy OR Rain AND TEMPERATURE is Cool THEN *Weightlifting*

is likely to cover more instances in the training set than, for instance,

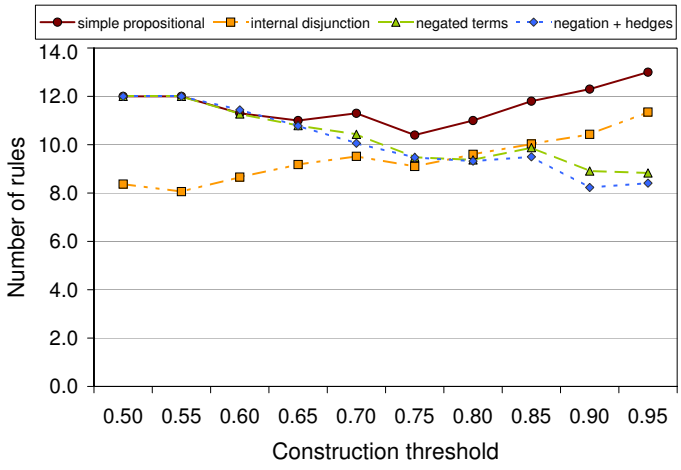
IF OUTLOOK is Cloudy AND TEMPERATURE is Cool THEN *Weightlifting*



(a) Wine

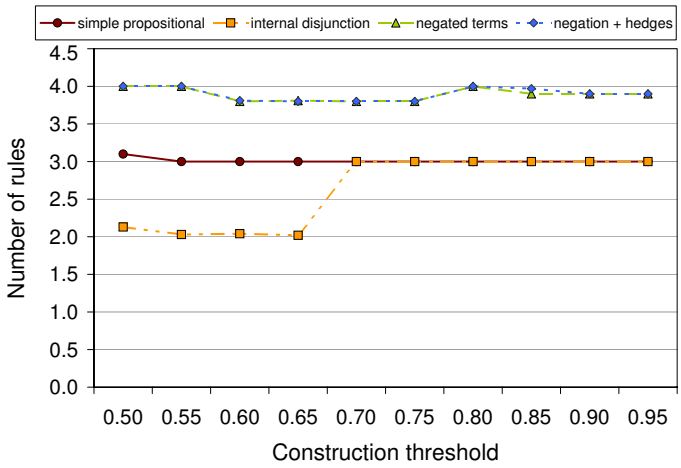


(b) Iris

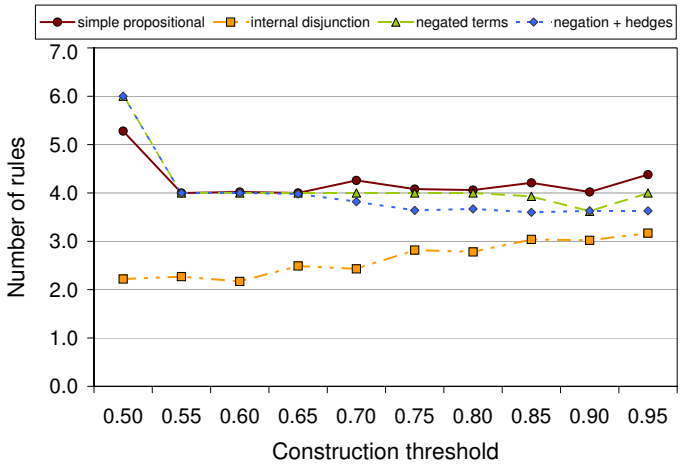


(c) Glass

Figure 7.7: Number of rules in *FRANTIC-fullIRL* induced rulebases using different forms of the hypothesis language



(d) Water Treatment



(e) Leukaemia

Figure 7.7: Number of rules in *FRANTIC-fullIRL* induced rulebases using different forms of the hypothesis language (cont.)

Table 7.6: *FRANTIC-simpSRL* parameter settings for exploring the system's performance

	Wine	Iris	Glass	WT	Leuk
`representation`	— Negation (Simple, Disjunction, Hedges) —				
`constructionThreshold`	— 0.50, 0.55, ..., 0.90, 0.95 —				
`minInstPerRule`	(50%) 60% (70%)	(40%,50%) 60%	50% (60%,70%)	70% (80%,90%)	(70%,80%) 90%
`numAnts`	(2,6) 10	(2,6) 10	(2,6) 10	(2,4) 6	(2,6) 10
`numIterations`	30	30	30	30	70

and probably at least as many as

IF OUTLOOK is NOT_Sunny AND TEMPERATURE is Cool THEN *Weightlifting*[1]

The more instances that are covered by a chosen best rule, the more that are removed from the training set in between ACO runs, and hence the fewer rules necessary to cover the required class instances. The total number of rules in a rulebase is therefore smaller.

The second observation is attributed to the pressure that an increasing `constructionThreshold` value exerts on the rule construction term set. As discussed in Sections 6.2 and 6.3, the greater the value the fewer the terms likely to meet the constraints imposed by the construction parameters. As explained, this also results in smaller rule antecedents, including fewer disjunctions between attribute values, and hence, fewer instances removed in between ACO runs. It appears that for the other forms of the hypothesis language, it is the `minInstPerRule` parameter that has direct and sole control over the number of rules in a rulebase.

7.2 Simultaneous Rule Learning Performance

As illustrated in the preceding section, more rules in a rulebase need not result in more accurate rulebases. This is attributed to one or two reasons:

1. rules induced later on to describe the same class may be overfitting the training data due to an inappropriately controlled changing `minInstPerRule`, and/or
2. the rulebase selection method, which determines and adds rules to the final rulebase without taking into account how the rules that are added later interact with rules already in the rulebase.

This section therefore explores a different rule learning strategy where rules are constructed and evaluated simultaneously, and compares the performance of rulebases induced in this manner

[1]Note that in finding the degree of match between a fuzzy rule and a fuzzy instance, the rule antecedents *IF* OUTLOOK *is Cloudy* OR *Rain* and *IF* OUTLOOK *is NOT_Sunny* are not necessarily exactly equivalent – the resulting degree of match between the first compound conditon (i.e. with internal disjunction) and the fuzzy values in an instance is likely to be different from the degree of match between the negated term in the second antecedent and the fuzzy values in the same instance. Section A.4 describes how these degrees are determined.

with that of those produced using the more common iterative rule learning strategy. In the current implementation of SRL only one rule is produced to describe each class, and so SRL induced rulebases are compared against those produced by simplified IRL. Restricting each of the different *FRANTIC* learning strategies to produce only one rule per class allows one to attribute any difference in performance directly to the rule learning approach, and not to any additional rules.

Note that the parameter settings for running in simplified SRL mode (Table 7.6) have in no way been optimised – they have been deliberately set (almost) the same as those for running *FRANTIC* in simplified IRL mode (Table 6.1). The SRL approach is more computationally expensive with regards to the rule evaluation process than the IRL approach, and so in order to test whether this approach may achieve reasonable results at a lower computational cost the number of iterations is reduced for most datasets from 50 or 70 or 100 to 30. This parameter setting for the Leukaemia dataset was inadvertently set to the same value as for when inducing rulebases using simplified IRL strategy. The results in the following Section 7.3 that discuss convergence for the two strategies, however, strongly indicate that very similar results are likely to have been attained if `numIterations` had also been set to the lower value used for the other datasets. The number of ants for the Glass dataset is reduced from 10 to 6 to speed up the generation of results – this issue of increased computational expense using an SRL rulebase evaluation strategy, and ways of resolving it, are discussed in detail in Section 7.3. It is likely that in general some effort at tuning parameters specifically for the SRL strategy is necessary and will yield better results.

The following Section 7.2.1 compares the simplified iterative rule learning strategy with the simplified simultaneous rule learning one with regards to classification accuracy, and investigates why and when simplified SRL produces improved results, and why and when it does not. This is found to be not only dependent on the actual rule learning strategy, but also on the way the selection of the final rules or rulebases is made, and highlights the consequent impact on pheromone trails for both IRL and SRL.

Section 7.2.2 compares the size of the rule antecedents produced by the simplified IRL and SRL approaches. As might be expected, since the rule discovery mechanism is identical in both strategies, very similar results are obtained and these are presented here in brief for the sake of completeness.

7.2.1 Classification Accuracy

Table 7.7 on the following page shows statistics for the rulebases achieving the highest accuracies induced using the two different rule learning strategies – Table 7.7(a) gives the average

Table 7.7: Comparison of *FRANTIC* rulebases induced using iterative (*simpIRL*) and simultaneous (*simpSRL*) rule learning strategies

(a) Classification accuracy (%) and standard error of the mean (+/-)

	simpIRL		*simpSRL*	
	%	+/-	%	+/-
Wine	93.01	0.77	93.07	1.30
Iris	95.33	0.00	96.00	0.00
Glass	55.63	1.64	61.81	1.28
WT	90.42	0.62	85.25	0.00
Leuk	95.63	1.55	94.29	1.29

(b) Number of rules and number of terms in a rule

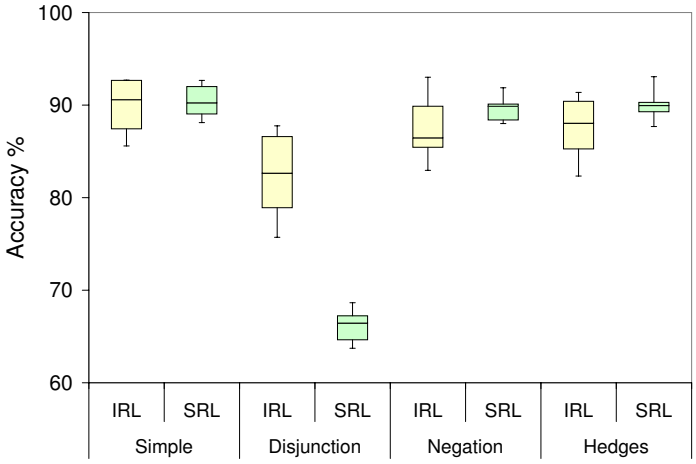
	simpIRL		*simpSRL*	
	Rules	Terms	Rules	Terms
Wine	3.00	5.61	3.00	5.48
Iris	3.00	2.00	3.00	1.67
Glass	6.00	6.58	6.00	6.55
WT	2.00	1.44	2.00	2.00
Leuk	2.00	10.74	2.00	12.62

accuracy (and standard deviation), and Table 7.7(b) the corresponding average number of rules in a rulebase, and the average number of terms per rule. Note that the results presented for the IRL strategy are the same as those presented in Table 7.2 on page 117. The accuracy statistics for simplified SRL induced rulebases are selected from Tables D.7 and D.9, and from Tables D.8 and D.10 for complexity details.

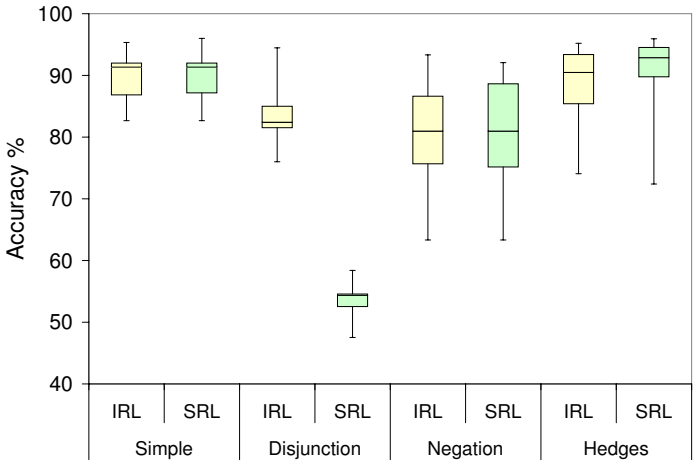
Table 7.7(a) shows that in three datasets (Wine, Iris and Glass) rulebases induced using the simultaneous rule learning approach perform slightly better than those produced using the iterative approach, and in the remaining two datasets the reverse is true. For the Glass dataset the accuracy improvement afforded by the SRL approach is non-negligible, but for the Water Treatment dataset the decrease in accuracy obtained using this approach is also significant. However, these are very high level results and an in-depth analysis yields much more useful and interesting information.

Figure 7.8 on the following page is produced using the data in Table C.6 for IRL values, and Table D.7 for SRL values. The tables present the accuracies of the induced rulebases using different forms of the knowledge representation language. The box-and-whisker plots in Fig. 7.8 give an indication of how robust these induced rulebases are with respect to changes in settings for the `constructionThreshold` parameter. The top and bottom whiskers indicate respectively the maximum and minimum average accuracy achieved by rulebases for `constructionThreshold` values in the range [0.50,0.95] (and hence the overall range of accuracies), and the box indicates the range in which 50% of the accuracies lie. The smaller the overall range and the smaller the box, the more robustness to changes in this parameter is exhibited. Naturally, where on the vertical scale these plots lie is also important – a plot may exhibit extreme robustness to a parameter, but may also achieve a very low accuracy.

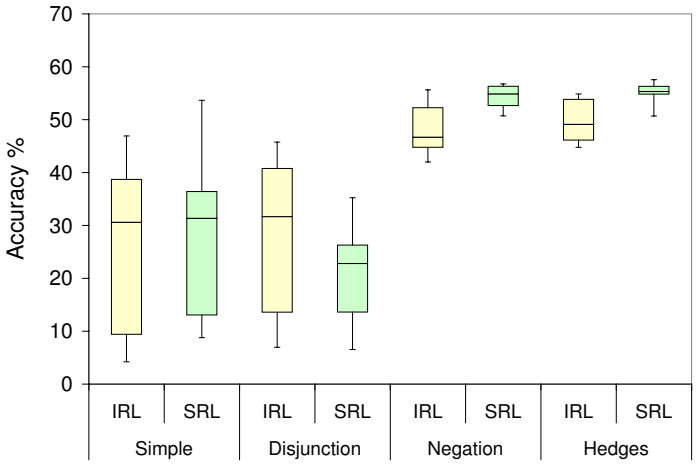
Several observations may be made from Fig. 7.8:



(a) Wine

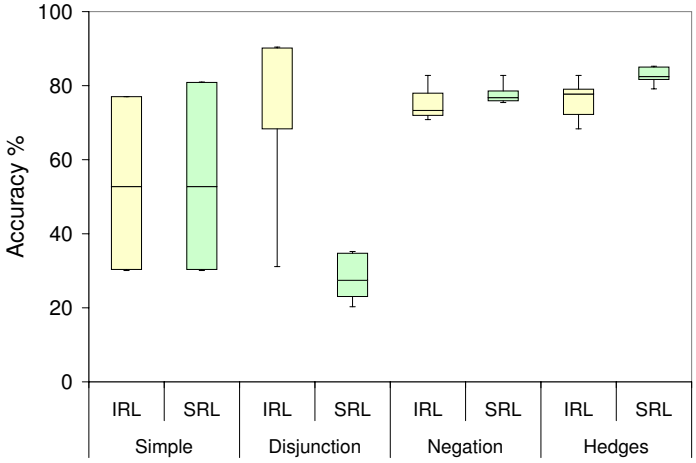


(b) Iris

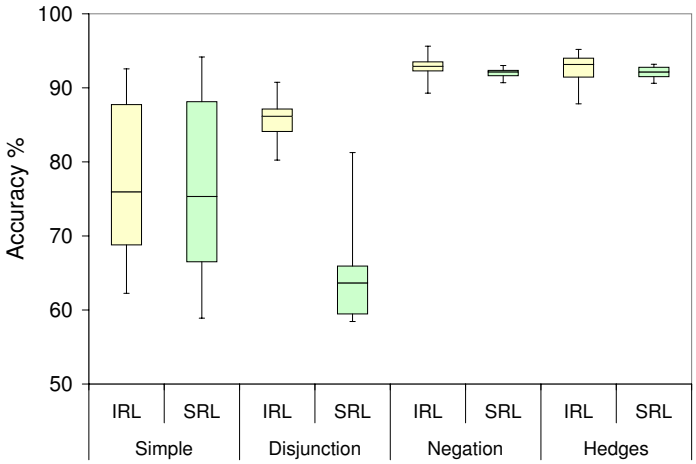


(c) Glass

Figure 7.8: Dispersion of accuracies achieved by *FRANTIC* rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote simplified SRL induced rulebases.



(d) Water Treatment



(e) Leukaemia

Figure 7.8: Dispersion of accuracies achieved by *FRANTIC* rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote simplified SRL induced rulebases. (cont.)

- Rulebases induced using the SRL approach often exhibit an increased robustness to the `constructionThreshold` value – the overall range is smaller than for rulebases induced using the IRL approach, and/or 50% of these accuracies lie in a smaller range;
- The highest accuracy is as often achieved by an IRL induced rulebase, as by an SRL induced rulebase, with the exception that,
- Rulebases containing rules with internal disjunction consistently perform better if induced using the IRL approach, than if induced using the SRL approach.

This section analyses these observations in more detail, explores reasons for their generation, and provides preliminary but strong evidence that further improvement using a simultaneous rule learning approach is possible.

Selection of iteration 'best' rule for pheromone updates

The more successful SRL rulebases are those that show both an increased robustness, and a comparable or improved accuracy when compared with IRL induced rulebases (e.g. rulebases for the Wine, Glass and Water Treatment datasets that contain rules with negated terms or with negated terms and linguistic hedges). These rulebases provide evidence to support the theory that constructing and evaluating rules describing different classes simultaneously, leads to complementary rather than competing rules in the final rulebase.

A common observation is made in such cases – during SRL, the rules in the best rulebase 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 of the Water Treatment dataset with `constructionThreshold=0.60`, and rules induced using negated terms and hedges, the first iteration of both IRL and SRL induced rulebases produces the same rules to describe the *Normal* class. During IRL, the rule with the highest fitness of 0.52 is chosen at the end of the iteration for pheromone updates.

However, this same rule is not the one that is found in the best performing rulebase at the end of the SRL first iteration – another rule that is ignored during IRL because of its low quality (a value of 0.28 when measured by the IRL fitness function (4.10)), is found to interact better with rules describing the other classes. It is therefore this low individual scoring rule that is used to update pheromones for its class at the end of the iteration. The final IRL induced rulebase achieves an accuracy of 59% on the test set, while the SRL induced rulebase (induced from the same training set), achieves an accuracy of 85% on the same test set.

Similarly for SRL rulebases that show an increased robustness but apparently at a slight cost to accuracy – the rules during SRL are chosen with a requirement to performing optimally

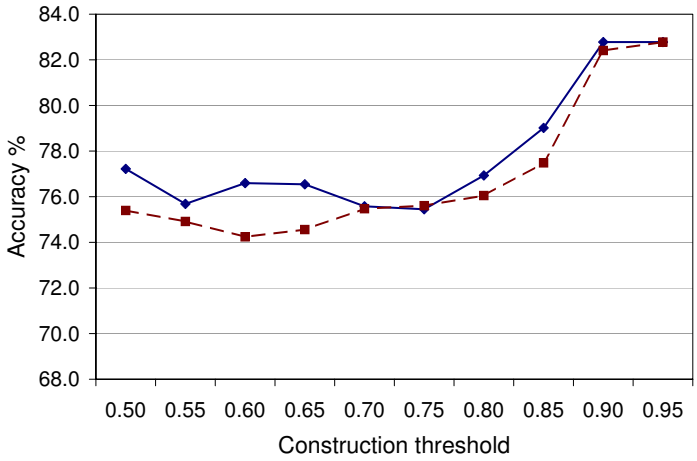


Figure 7.9: Accuracy achieved by *FRANTIC-simpSRL* induced Water Treatment rulebases with (solid line) and without (dashed line) the use of pheromones

together, so why is there not a consequential improvement in accuracy? The following subsections explore possible answers to this question.

Selection of the final rulebase

In investigating the impact of pheromone trails in Section 6.2, it was determined that they had minimal effect on the accuracy of simplified IRL induced rulebases. However, analysis of the fitness statistics from each iteration generated during SRL runs suggests that improved accuracy is obtained from SRL induced rulebases when convergence occurs to an optimal rulebase, while comparable or slightly decreased accuracy appears to be a result of convergence to a near-optimal rulebase[2]. This suggests that pheromone plays a different role in the two strategies, and highlights an important difference that is not necessarily immediately obvious.

In designing the SRL extension of *FRANTIC* the common assumption was adopted that pheromones trails will lead the system to a good solution. Therefore, in SRL, the final rulebase is the best one of the final iteration, on the assumption that good information is obtained from earlier iterations and is used by ants in later iterations to construct increasingly better solutions. In IRL, the best rule added to the rulebase is the best out of all rules produced by all iterations, (Fig. 4.5 on page 58). Therefore, if at some point this rule has been constructed, the system need not necessarily rely on pheromone trails to converge to it – the best rule is chosen irrespective of whether it ends up in the final iteration or not.

On the other hand, the current SRL version is entirely dependent on pheromone trails and their

[2]Convergence for SRL induced rulebases is determined differently than for IRL induced rulebases. This is clarified in Section 7.3.2 but for now it suffices to think of SRL convergence in a similar manner to that defined for IRL, only it is rulebases that are being evaluated in each iteration and not individual rules.

ability to converge to an optimum – or near-optimum – rulebase. There is also the possibility, however, that though convergence may not occur, a good rulebase is created in the final iteration and is selected. Preliminary evidence that provides initial support for this theory is presented, where the results pertain to rulebases induced on the Water Treatment dataset, and where the rule antecedents may contain negated terms. Figure 7.9 shows the accuracy achieved by rulebases when they are induced with the use of pheromone (solid line) and without (dashed line). In looking at the convergence details for the two types of induced rulebases, it is observed that convergence always – and only – occurs when pheromones are used.

The difference in accuracy is small, and this is because even though convergence is unable to occur, good rulebases are induced and often end up in the final iteration where they are selected as the output of the system. However, the difference is consistent over the threshold values and is especially significant when contrasted with the equivalent figure for IRL induced rulebases – Fig. 6.3(d) on page 97, where there is no difference for rulebases induced on this dataset, since the best rule is chosen irrespective of whether pheromones are used or not.

The results relating to SRL induced rulebases using rules with internal disjunction, however, are extreme and very noticeable. This is a consequence of an inability on the system's part to reach convergence at all, or to produce a good rulebase in the final iteration. The next subsection explores why this situation is aggravated for rulebases containing this particular type of rule.

Solution construction validation method

Fitness statistics for SRL induced rulebases with rules containing internal disjunction between attribute values, show that convergence does not take place within the allotted 30 iterations, and though good rulebases are produced, they rarely turn up again in the final iteration. Hence, the consistently low accuracy rulebases outputted by the system when compared with IRL rulebases.[3]

The lack of convergence with this form of the hypothesis language was initially attributed to the impact that the solution validation method has on the size of the hypothesis or solution space, and that this space may be larger for rules with internal disjunction than for the other forms of the hypothesis language.

The hypothesis space is here defined as the space of unique rules (or solutions) that may be created by *FRANTIC*, unique meaning that the relative position of terms in an antecedent is not important, so that rules with the same terms but positioned differently are considered as

[3]Convergence for rulebases induced with this type of rules also rarely occurs for IRL induced rulebases, but here, the method of selecting the class rule to add to the final rulebase compensates for the inability of the system to converge. These convergence findings for both IRL and SRL induced rulebases are discussed in detail in Section 7.3.2.

the same rule. The size of this hypothesis space, i.e. the number of uniques rules that may be constructed is determined by two factors:

- the size of the problem graph, and
- the solution construction validation method.

Simple propositional rules, propositional rules with negated terms, and those with negated terms and linguistic hedges use the same solution validation method – at most, only one value from an attribute may be included in the rule antecedent. Hence, the number of unique rules for these forms of the hypothesis language is defined by the same formula, a function of n – the number of conditional attributes in a dataset, and d – the domain size of an attribute (here assuming all attributes have the same size)[4]:

$$\begin{aligned} numR1(d,n) &= \sum_{r=1}^{n} ({}^{n}C_r \times d^r) \\ &= (d+1)^n - 1 \end{aligned} \tag{7.2}$$

The inner formula ${}^{n}C_r \times d^r$ was introduced in Section 6.3. The first part – ${}^{n}C_r$ – is the number of unique combinations of r attributes from a total of n. For each selection of attributes, each attribute has d values in its domain from which one is chosen, and there are r attributes, hence d^r. The summation gives the total number of unique rules that may be created where the rule antecedent size varies from one, to the number of attributes in the dataset. Note that this is the number of unique rules to describe *one* class of the dataset. This number is multiplied by the number of classes in the dataset to give the total number of rules that may be used to describe all classes.

Furthermore, for any one dataset the number of unique rules for a class varies with each form of the hypothesis language used due to the different size of the problem graph. In the datasets used here, $d = 3$ for simple propositional rules, since there are three values in the domain of each attribute (e.g. {*high*, *medium*, *low*})[5], $d = 6$ for rules with negated terms (e.g. {*high*, *not-high*, *medium*, *not-medium*, ...}), and $d = 12$ for rules with negated terms and linguistic hedges (e.g. {*high*, *not-high*, *very-high*, *more_or_less-high*, ...}).

The number of unique rules with internal disjunction that may be created is also dependent on the number of attributes and their domain size. However, in this form of the hypothesis language the solution validation method permits more than one value from the domain of an

[4]Note that since the rule construction mechanism is identical in IRL and SRL, formulas (7.2) and (7.3) which determine the size of the hypothesis space for different forms of the hypothesis language, are applicable to both strategies.

[5]The Water Treatment dataset has two attributes with two values in their domain, but the simplifying assumption that domain size is the same for all attributes in all datasets is useful here for illustrative purposes.

attribute to be included in the rule antecedent. Hence:

$$\begin{aligned} numR2(d,n) &= \sum_{s=1}^{nd} {}^{nd}C_s \\ &= 2^{nd} - 1 \end{aligned} \tag{7.3}$$

The inner formula ${}^{nd}C_s$ is the number of unique combinations of s attribute-*values* from a total of nd. These combinations are summed over all possible rule antecedent sizes, from one to the total number of attribute-values.

For any dataset, the higher the value of d in formula (7.2) the greater the number of unique rules. Therefore, rules that contain negated terms and linguistic hedges ($d = 12$) have a larger hypothesis space than simple propositional rules or propositional rules with just negated terms. In order to determine when rules with internal disjunction have a larger hypothesis space than rules with negated terms and hedges, the inequality $2^{nd} - 1 \geq 13^n - 1$ is solved, giving $d \geq 3.7$. This means that the attributes in a dataset must have an average domain size of 3.7 or greater.

In these datasets the domain size is restricted to three, so that the number of rules with negated terms and hedges is actually greater than the number of rules with internal disjunction. Yet, as will be demonstrated in the following section, experiments on these datasets run with rules containing negated terms and hedges often converge, while those run with rules containing internal disjunction between attribute-values never converge. The current inability to converge for this type of rule appears to be a result of the impact of the solution construction validation method on the rule construction parameter `minInstPerRule`. An explanation follows.

The rule construction parameters restrict the creation (not search) of rules to those that satisfy both their settings. For forms of the hypothesis language other than the one with internal disjunction, the more terms added to the rule antecedent, the more likely it is that the current partial rule antecedent covers fewer instances in the training set, and that the selected term may not be retained. This restricts the construction of rule antecedents to those with sizes that are more likely to be at the lower end of the possible range, rather than the upper end ($r = n$ in formula (7.2)).

In constructing rules with internal disjunction, if a term is selected for inclusion in an antecedent that does not contain any other terms of the same attribute (e.g. if current rule antecedent is *IF* TEMPERATURE *is Hot*, and the currently selected term is OUTLOOK *is Cloudy*), then this term is subject to the same constraints as mentioned for the other forms of the hypothesis language – the more terms already present in the antecedent, the less likely it will be retained. However, if the selected term has the same attribute as another term in the rule antecedent (e.g. antecedent is *IF* OUTLOOK *is Rain*, and term is OUTLOOK *is Cloudy*), then the inclusion in the antecedent of the new term will *not* lower the number of instances in the

training set covered by the new partial rule antecedent, and, it will be retained.

It is this that allows ants creating rules with internal disjunction to construct rules from a much larger portion of the hypothesis space than is possible for the other forms of the hypothesis language – it allows ants to also create rules that come from the upper end of the range with respect to rule antecedent sizes.

Is accuracy improvement possible?

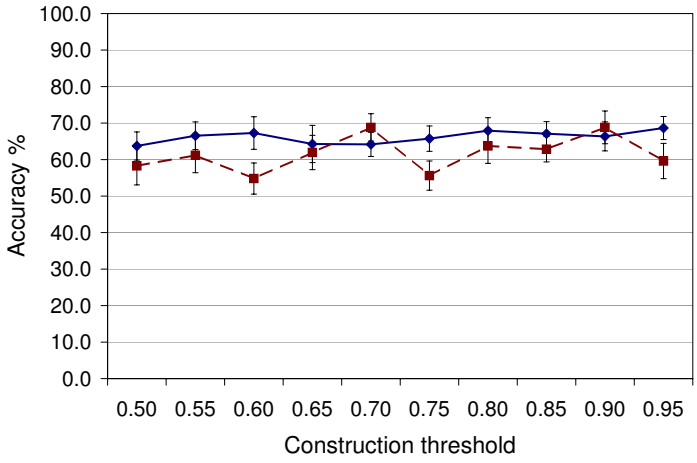
Simultaneous construction and evaluation of fuzzy rules can provide benefit, but this benefit is often negated by the way the final rulebase is chosen, and is particularly sensitive to rules with internal disjunction between attribute-values.

The following paragraphs and figures provide preliminary evidence that the SRL approach may be improved, by presenting early results of additional experiments on all datasets inducing rules with internal disjunction. The statistics pertaining to these new experiments are obtained using one ten-fold cross-validation for all datasets[6] except the Water Treatment one (where the figures are still based on ten ten-fold cross-validations).

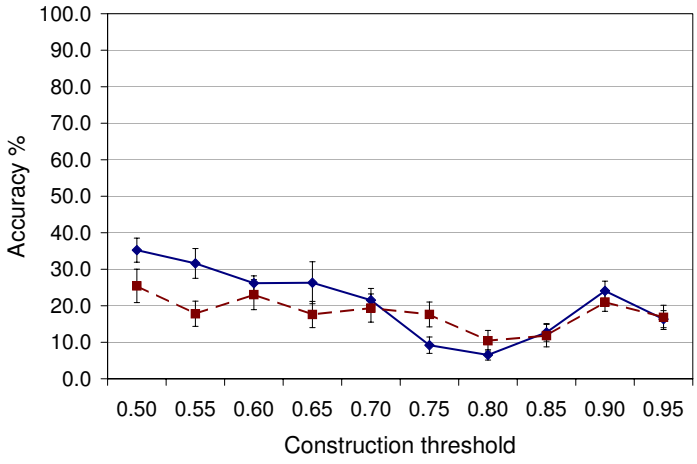
Since the number of iterations for inducing SRL rulebases was reduced to 30, one exploratory set of experiments is to rerun tests on three of the datasets with `numIterations=100`. Figure 7.10 on the next page presents the results. For the Water Treatment dataset, Fig. 7.10(c), there appears to be a negligible increase in the accuracy of the rulebases induced for several `constructionThreshold` values. For the other two datasets, although the error bars do not overlap for some `constructionThreshold` values it is likely that the greater change in accuracy – both positive and negative – is as a result of comparing the data from *one* ten-fold cross-validation, against the data from *ten* ten-fold cross-validations. It is possible that a much greater number of iterations may permit convergence for this form of the language, but since computational expense may be a consideration, and in order to attempt to confirm the analyses in the preceding paragraphs, a minor change to this rule learning strategy is implemented instead.

Figure 7.11 on page 152 compares the results on all datasets of the original SRL induced rulebases using 30 iterations, with the performance of new rulebases induced using 30 iterations, but where the final rulebase selected by the system is the best of all iterations (and not the best of the final iteration). Though the new results for all datasets except the Water Treatment one are based on one ten-fold cross-validation, the difference from the original accuracies is such that any negative or positive change in difference that might occur due to averaging over more

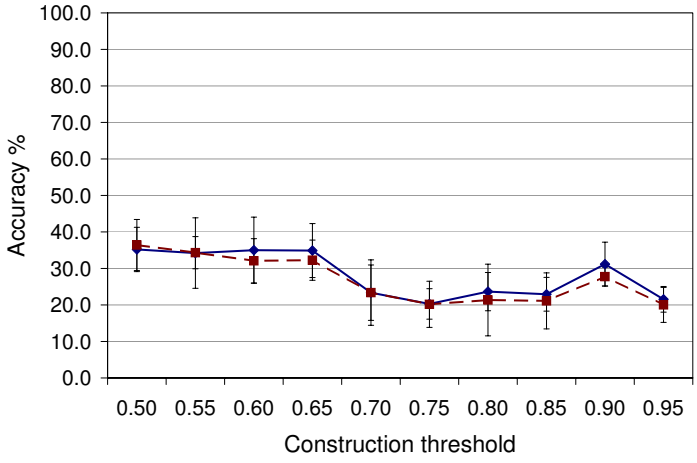
[6] The standard error of the mean in this case is calculated as the standard deviation of the accuracies of the 10 models produced, divided by the square root of the sample size, i.e. divided by $\sqrt{10}$.



(a) Wine

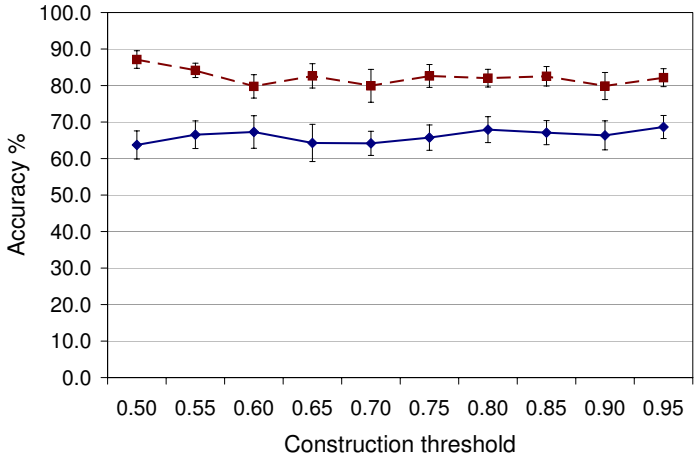


(b) Glass

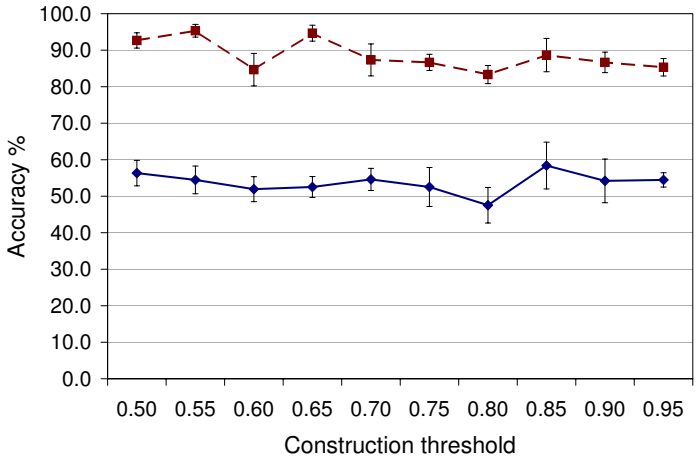


(c) Water Treatment

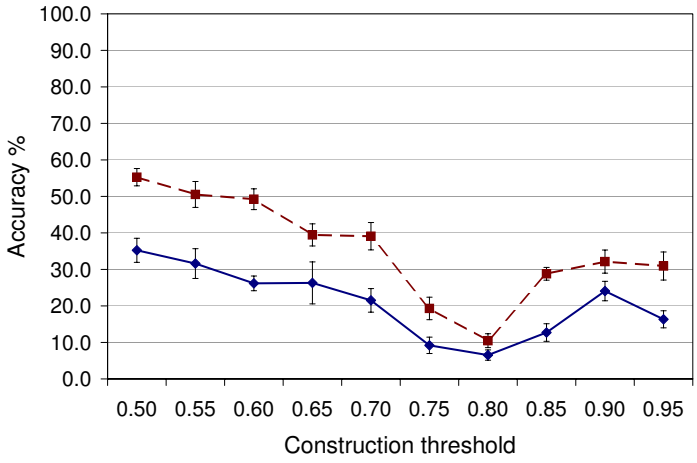
Figure 7.10: Accuracy of *FRANTIC-simpSRL* rulebases induced using 30 iterations (solid line) and 100 iterations (dashed line). Rules contain internal disjunction between attribute values.



(a) Wine

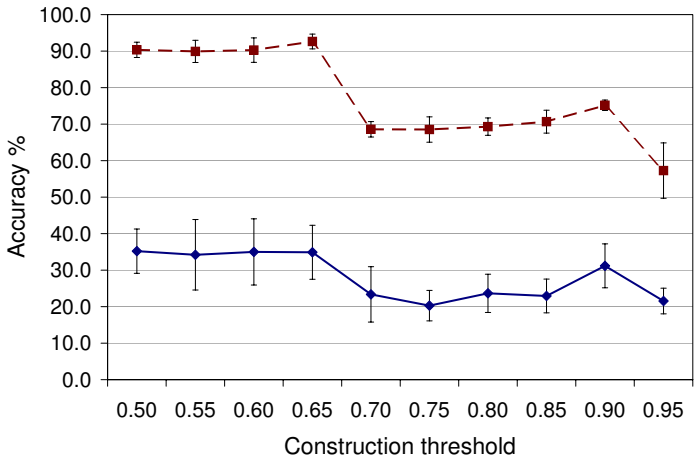


(b) Iris

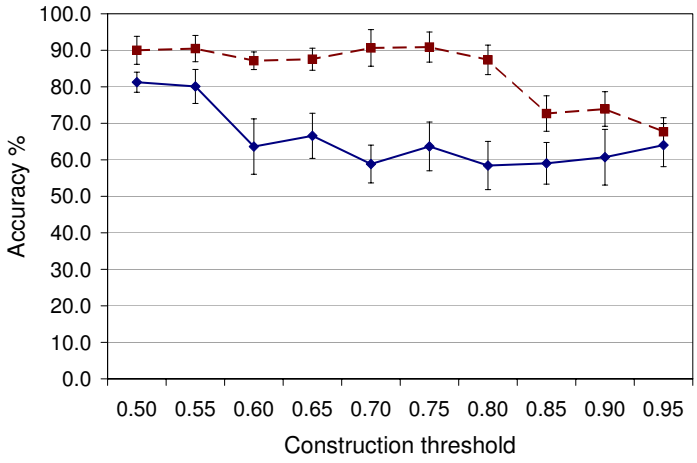


(c) Glass

Figure 7.11: Accuracy of *FRANTIC-simpSRL* induced rulebases where final rulebase is the best in the final iteration (solid line), and where final rulebase is the best of all iterations (dashed line). Rules contain internal disjunction between attribute values.



(d) Water Treatment



(e) Leukaemia

Figure 7.11: Accuracy of *FRANTIC-simpSRL* induced rulebases, where final rulebase is the best in the final iteration (solid line), and where final rulebase is the best of all iterations (dashed line). Rules contain internal disjunction between attribute values (cont.)

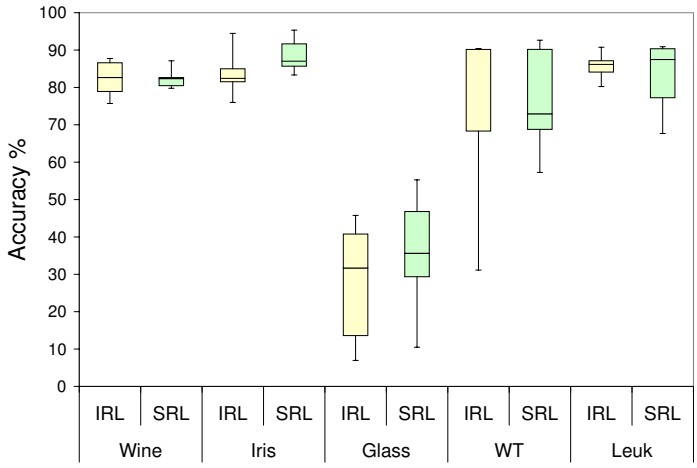


Figure 7.12: Revised accuracy comparison for *FRANTIC* IRL and SRL induced rulebases containing rules with internal disjunction. SLR induced rulebases are the best of all iterations.

cross-validations, is unlikely to completely negate the apparent beneficial effect of selecting the final rulebase out of all those created.

The power of keeping track of the best rulebase is appreciated by noting that the positive increase in accuracy is perceived across the range of `constructionThreshold` values, across all the datasets, and as already noted, that there is no improvement for this form of the language even when the number of iterations is tripled.

Figure 7.12 provides a revised comparison of IRL and SRL induced rulebases for rules with internal disjunction. It compares the data relating to the new SRL rulebases presented in the previous Fig. 7.11 (where the final rulebase is the best of all iterations), with the IRL rulebase data presented in Fig. 7.8. The SRL induced rulebases show an improvement in robustness to the `constructionThreshold` parameter, and/or in accuracy over the IRL induced rulebases. Table 7.8 on the following page presents the highest accuracy achieved for each dataset by the two modes.

The improvement in robustness is sometimes considerable (even for SRL induced rulebases where the selection is based on the final iteration, Fig. 7.8), though the difference in accuracy is often negligible. Increased robustness to parameter settings is extremely useful, as less effort may be necessary to determine optimum settings for experiments. It might be possible to improve SRL accuracy results with parameter tuning – the settings used for SRL experiments are the same as those used for the simplified IRL experiments. These settings were based on IRL exploratory runs and so it is conceivable that SRL induced rulebases require their own exploratory runs and parameter setting.

Disjunctive rules were selected as the form of the hypothesis language to use for these initial

Table 7.8: Highest accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC* IRL and SRL induced rulebases containing rules with internal disjunction. SLR induced rulebases are the best of all iterations.

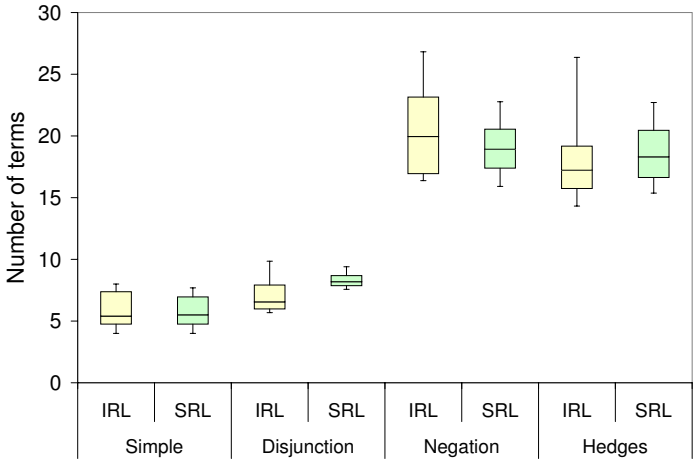
	IRL			SRL		
	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
Wine	87.76	2.17	7.00	87.13	2.41	7.62
Iris	94.47	1.51	6.18	95.33	1.74	5.49
Glass	45.76	2.17	10.27	55.27	2.37	7.50
WT	90.42	0.62	13.50	92.64	2.03	7.85
Leuk	90.76	2.65	11.99	90.89	4.13	13.05

experiments exploring the impact of the SRL rulebase selection method, as they performed the worst on all datasets when compared with IRL induced rulebases of the same form of the language (Fig. 7.8). However, a possibility is that the level of accuracy improvement achievable by the SRL approach is limited by the richness of the hypothesis language used, and the number of different rules that may be created. In these experiments, the hypothesis space of rules with internal disjunction is greater than that for rules with negated terms. The former type of rules have the ability to describe ranges over a variable, but they lack the facility to add new information to a rule (as negated terms do), or to increase or decrease the precision of the terms (as linguistic hedges do). More rules that are more expressive and precise may well enable the SRL evaluation process to make subtle distinctions between rules that result in significant gains with regard to accuracy as well as robustness. Due to time constraints, testing this hypothesis, and running more extensive experiments with the new rulebase selection method is left for important future work.

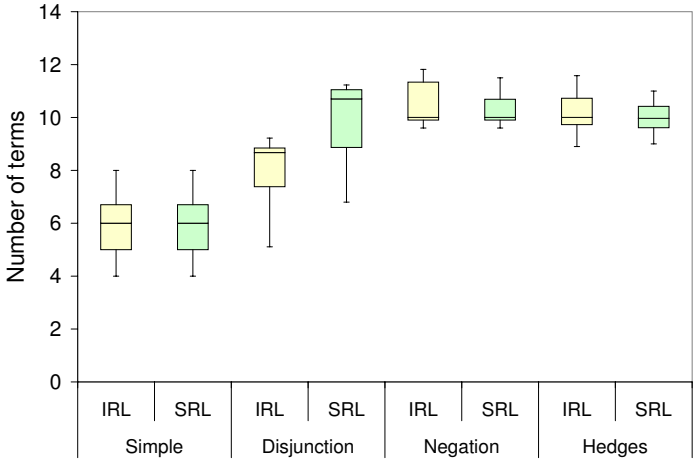
7.2.2 Model Complexity

Figure 7.13 on the next page presents box-and-whisker plots for each of the datasets, comparing the model complexity of IRL and SRL induced rulebases for each form of the hypothesis language. The statistics that generate these graphs are in Tables C.5 and D.8 for IRL and SRL induced rulebases respectively. These model complexity statistics correspond to the rulebase accuracy statistics presented in Fig. 7.8.

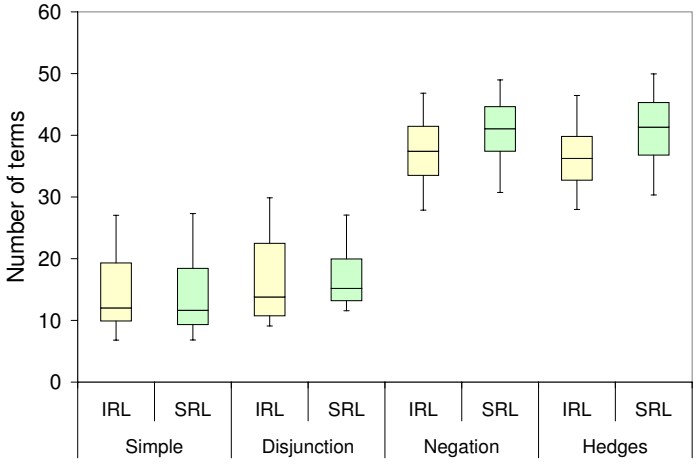
The number of terms in the rulebases induced by the simplified IRL and SRL strategies is very similar (using SRL results where the final rulebase is selected as the best of the final iteration). This is due to the fact that the rule discovery mechanism and the settings used for the rule



(a) Wine

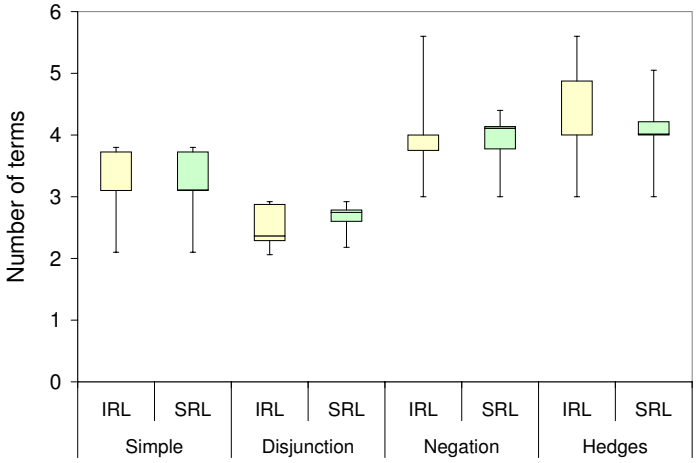


(b) Iris

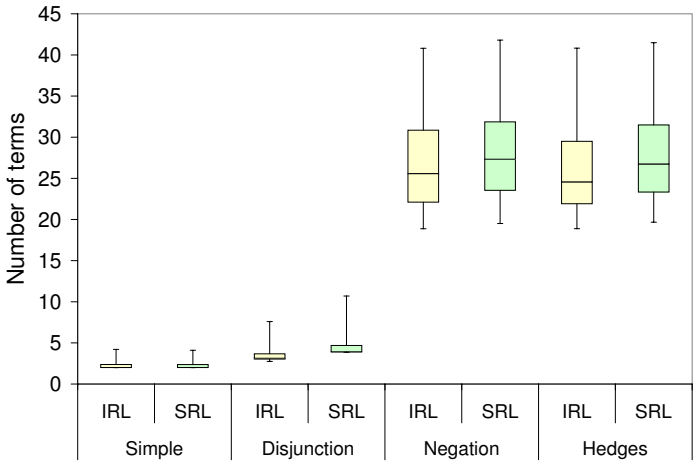


(c) Glass

Figure 7.13: Number of terms in *FRANTIC* IRL and SRL rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote simplified SRL induced rulebases.



(d) Water Treatment



(e) Leukaemia

Figure 7.13: Number of terms in *FRANTIC* IRL and SRL rulebases induced over different `constructionThreshold` values. Yellow box plots denote simplified IRL induced rulebases, and green box plots denote simplified SRL induced rulebases (cont.)

construction parameters for the two strategies are identical in the experiments. This similarity in complexity is evident even when there are significant differences in the accuracies achieved by the IRL and SRL rulebases (such as for all datasets when the rules have internal disjunction between attribute values). Rulebases containing simple propositional rules show the greatest similarity in complexity results for all datasets. The complexity of rulebases containing rules of the other forms of the hypothesis language depict a greater difference in size between the two strategies over some or many `constructionThreshold` values (the tables referred to in the preceding paragraph show detailed results). This is thought to be due to the larger size of the hypothesis spaces formed by these forms of the language, and consequently, the larger number of rules of different sizes that may be created.

7.3 Computational Cost

This section looks at the complexity of the two learning strategies, and then explores ways of reducing the computational cost. In pursuit of this second aim, both the literature and *FRANTIC*'s current implementation are reviewed. The main goal directing the design of the system's current version was flexibility, i.e. the ease with which new ideas may be explored, and hence how quickly extensions and modifications to the system may be made. However, it is demonstrated that several opportunities exist that will render the system more amenable to the scaling up of problems.

7.3.1 Complexity of the Learning Strategies

This subsection analyses and compares the complexity of the two different learning strategies. Rule construction is identical in both, but rule evaluation is not and initially appears much more computationally expensive for the SRL approach. Suggestions are made as to how the computational cost may be made more equal between the two strategies.

The notation for different elements used in the analysis of *FRANTIC*'s computational complexity is presented in Table 7.9 on the following page. The problem (dataset) size is defined by the number of attributes in the dataset, the average domain size, the number of instances, and the number of classes. These impact on *FRANTIC*'s complexity, as do the system settings, primarily, the number of iterations and the number of ants within an iteration. The worst case complexity of running *FRANTIC* in simplified IRL mode is determined first, and the complexity of other variants of the system is then described in terms of similarities and differences to *FRANTIC-simpIRL*.

Table 7.9: Notation used in *FRANTIC* complexity analysis

Notation	Description
n	Number of attributes in dataset
d	Domain size for each attribute
k	Number of classes in dataset
m	Number of total instances in dataset
m_{k_i}	Number of instances in dataset of class k_i
q	Number of iterations in an ACO
p	Number of ants per iteration of ACO

FRANTIC-simpIRL

Figure 7.14 on the next page provides an algorithm outline for *FRANTIC-simpIRL*, with enough detail to highlight the more significant parts affecting its complexity, and to maintain coherence in the description.

Consider first the construction of one rule antecedent by an ant, Fig. 7.14 lines (6)–(10). Line (7) conceptualises the calculation of term probabilities as defined in formula 4.9; at the beginning of the rule construction process there are $n \cdot d$ such probabilities to calculate. Selecting a term, line (8), and flagging a term after checking whether it meets the rule construction criteria and therefore whether it should be retained or not, line (10), is here considered to take constant time. Checking the rule construction criteria, line (9), requires going over the instances in the training set that belong to the same class as the rule being constructed, to check whether a sufficient number of them are covered by the new rule antecedent to a sufficiently high degree. At best, only `minInstPerRule` class instances need be checked. At worst, all class instances in the training set are checked.

This process is repeated until all terms in the construction graph have been flagged, though not all need to have been *selected* to be flagged. This is dependent on the form of the hypothesis language used, specifically, on the solution construction validation method. For rules that may contain internal disjunction between attribute values, all terms in the construction graph are selected and checked to see whether they may be retained in the rule antecedent. For the other forms of the language, if a term is selected and retained in the rule antecedent, then not only is *it* flagged so that it is not reselected, but other terms that belong to the same attribute domain are also flagged. For these additionally flagged terms, no renormalisation of term probabilities is required, and no check is made to determine whether the rule construction parameters are satisfied, which may lead to a reduction in actual run time. For these forms of the hypothesis language, at best only n selections are made (with the consequent probability calculation and

```
(1)    Determine heuristic values
(2)    FOR k classes
(3)        Initialise pheromone levels
(4)        FOR q iterations
(5)            FOR p ants
(6)                While terms available
(7)                    Determine term probabilities
(8)                    Select term
(9)                    Check construction criteria
(10)                   Flag terms as appropriate
(11)               FOR m instances
(12)                   Find degree of match between rule and instance
(13)                   Update fitness statistics for rule
(14)               ENDFOR
(15)           ENDFOR
(16)           Determine best rule
(17)           Update pheromone levels using best rule
(18)       ENDFOR
(19)       Add best rule to final rulebase
(20)   ENDFOR
(21)   Output final rulebase
```

Figure 7.14: Pseudocode for *FRANTIC* simplified class-dependent iterative rule learning

criteria checks), and at worst $n \cdot d$ such selections are made. In practice, the result is somewhere in between and a useful topic for future work will be exploring this empirical complexity.

It should also be remembered that the construction graph for creating rules with negated terms is twice the size of that for creating simple propositional rules or rules with internal disjunction, and that the graph for creating rules with negated terms and linguistic hedges is four times the size. It is often observed in practice, therefore, that though during the construction of rules with internal disjunction *all* terms are *always* selected, rules that include negated terms, and negated terms and linguistic hedges, may take up to twice and four times as long, respectively, as do rules with internal disjunction.

Ignoring constant time operations and assuming worst case scenarios, therefore, constructing one rule is of the order:

$$\begin{aligned} O\left(\frac{n \cdot d(n \cdot d+1)}{2} + n \cdot d \cdot m_{k_i}\right) &= O\left(max((n \cdot d)^2, n \cdot d \cdot m_{k_i})\right) \\ &= O\left((n \cdot d)^2\right) \end{aligned} \tag{7.4}$$

The part $\frac{n \cdot d(n \cdot d+1)}{2}$ is the sum of the arithmetic progression arising from considering $n \cdot d$ terms in the first step of the rule construction process, $(n \cdot d) - 1$ in the next step, then $(n \cdot d) - 2$ and

so on. The number of instances with the same class as the rule being constructed, m_{k_i}, remains the same throughout the construction process for a particular rule. In practice it is possible that in some cases the dominant operation becomes the calculation of term probabilities (rather than checking whether the current partial rule antecedent satisfies the construction parameters) – this is appropriate in circumstances where the construction graph is very large in proportion to the number of instances in the training set (such as for tests run on the Leukaemia dataset where the graph for rules with negated terms and linguistic hedges has 600 nodes, and the training set has 65 instances). Informative future work will include investigating empirically the run time for datasets with both a large number of attribute values and number of instances, and finding the possible breakpoint at which checking the construction criteria might become the more dominant operation.

Rule evaluation, Fig. 7.14 lines (11–14), is dependent on the number of terms in the rule antecedent and the number of total instances in the dataset; updating the fitness statistics as in lines (4)–(16) of Fig. 4.7 on page 61, is the less dominant operation and takes constant time $O(1)$. For simple propositional rules, rules with negated terms, and those with both negated terms and hedges, the best case is where there is only one term in the antecedent, and the worst is when there are n terms, i.e. one value from the domain of each attribute has been selected and retained.

For rules with internal disjunction between attribute values, the best case again has one term in the antecedent. The worst case has $n \cdot (d-1)$ terms, which is the greatest number of terms possible in a rule as permitted by this solution validation method; if all terms for an attribute are added to the rule antecedent – d terms – the interpretation is that the attribute is irrelevant for classification and the terms are removed at the end of the rule construction process. This leaves at most $(d-1)$ terms that may be added to the antecedent for each of n attributes.

In the worst case, therefore, taking into account rules with internal disjunction and m instances in the training set, rule evaluation is of the order:

$$\begin{aligned} O(m \cdot (n \cdot (d-1) + O(1))) &= O(max(n \cdot d \cdot m, O(m))) \\ &= O(n \cdot d \cdot m) \end{aligned} \tag{7.5}$$

Note, though, that the rule construction parameters provide a balance between creating very specialised and very generalised rules, and so unless their values are set very low, the actual number of terms in an antecedent is somewhere between the best and the worst case figures. Also, though both the number of attributes and number of instances may be large, in order to maximise human comprehensibility of the induced knowledge the number of fuzzy sets tends to be rather small; for instance, in the tests run on all datasets during this research, and depending on the form of the hypothesis language used, the value of d is 3, 6 or 12.

Determining the best rule out of all the rules created by ants in an iteration, line (16), is of the order $O(p)$, and updating the pheromone levels on nodes of the construction graph using the best rule, line (17), is of the order $O(n \cdot d)$. The complexity of the work involved in running one iteration with p ants is:

$$O\left(p \cdot \left((n \cdot d)^2 + n \cdot d \cdot m\right)\right) \tag{7.6}$$

Other operations necessary to run a complete ACO that eventually adds a best rule to the final rulebase, lines (3)–(19), are: initialising pheromone levels (line (3), order of $O(n \cdot d)$), and adding the best rule to the final rulebase (line (19), $O(1)$). At the beginning of a *FRANTIC-simpIRL* run heuristic levels are calculated (line (1), $O(n \cdot d \cdot m)$), and at the end the final rulebase is outputted (line (21), $O(k)$ since k ACO runs each produce a rule). None of these operations are dominant, so that the final complexity to produce a rulebase with one rule to describe each of k classes, where each such rule is produced as a result of running p ants within q iterations is:

$$O\left(k \cdot q \cdot p \cdot \left((n \cdot d)^2 + n \cdot d \cdot m\right)\right) \tag{7.7}$$

FRANTIC-fullIRL

When running *FRANTIC* in full IRL mode more than one rule may be created to describe each class so that the complexity becomes:

$$O\left(r \cdot q \cdot p \cdot \left((n \cdot d)^2 + n \cdot d \cdot m\right)\right) \tag{7.8}$$

where r is the number of rules actually created. As discussed, the parameter `minInstPerRule` impacts on this factor and the higher the value the fewer the rules. For the datasets used in this work, higher accuracy rulebases are obtained with this parameter set to 40% or 50%, and in these cases the average number of rules is approximately double the number of classes, varying slightly with the value for the `constructionThreshold` parameter, and generally lower for rules with internal disjunction (Table **??** on page ??). Though it is not possible to predict the number of rules generated over all possible datasets, the complexity of the two IRL variants is comparable.

FRANTIC-simpSRL

Figure 7.15 on the following page provides an outline of *FRANTIC* simultaneous rule learning. Rule construction is identical in iterative and simultaneous rule learning, but rule and rulebase evaluation are different, and this becomes the dominant factor in SRL.

```
(1)   Determine heuristic values
(2)   Initialise pheromone levels
(3)   FOR q iterations
(4)       FOR k classes
(5)           FOR p ants
(6)               While terms available
(7)                   Determine term probabilities
(8)                   Select term
(9)                   Check construction criteria
(10)                  Flag terms as appropriate
(11)          ENDFOR
(12)      ENDFOR
(13)      FOR p^k rulebases
(14)          FOR m instances
(15)              FOR k rules
(16)                  Find degree of match between rule and instance
(17)              ENDFOR
(18)              Classify instance
(19)          ENDFOR
(20)      ENDFOR
(21)      Determine best rulebase
(22)      Update pheromone levels using best rulebase
(23)  ENDFOR
(24)  Output final rulebase
```

Figure 7.15: Pseudocode for *FRANTIC* simplified class-dependent simultaneous rule learning

During rulebase evaluation, Fig. 7.15 lines (13)–(20), a rulebase is formed by including one rule that describes each class in the dataset, and using it to classify the training set. In each iteration of SRL, therefore, there are p^k unique rulebases that are formed and must be evaluated, assuming that each rule created by an ant to describe a particular class is different from all other rules created to describe that class. For each rule in a rulebase, the degree of match with m instances in the training set must be determined line (16); this is similar to the first and most time consuming step of rule evaluation in IRL so that the complexity for gathering degree of match data for a rule in SRL for all m instances is also $O(n \cdot d \cdot m)$, formula (7.5). Classifying an instance after degrees of matches have been found to it for all rules in a rulebase, line (18), is of the order $O(k)$.

Evaluating p^k rulebases in one iteration therefore leads to a complexity of:

$$\begin{aligned} O\left(p^k \cdot m \cdot (k \cdot n \cdot d + O(k))\right) &= O\left(max\left(p^k \cdot m \cdot k \cdot n \cdot d, O(p^k \cdot m \cdot k)\right)\right) \\ &= O\left(p^k \cdot m \cdot k \cdot n \cdot d\right) \end{aligned} \tag{7.9}$$

Taking into account q iterations, the order of complexity for *FRANTIC-simpSRL* is:

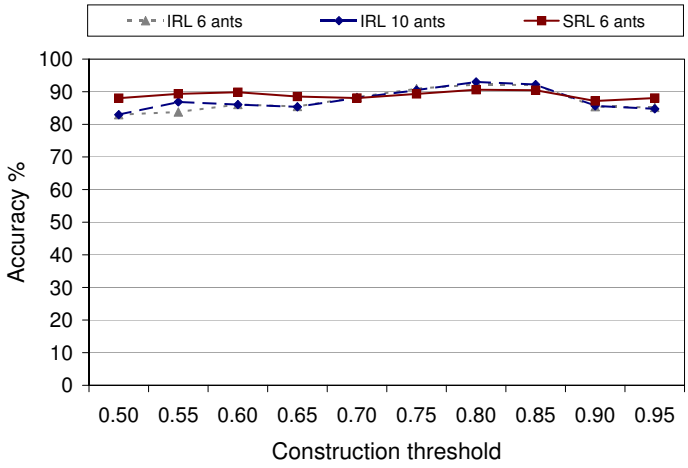
$$O\left(q \cdot p^k \cdot m \cdot k \cdot n \cdot d\right) \tag{7.10}$$

The above complexity function for SRL is more computationally expensive than the one for IRL, function (7.7). The scaling of problems for IRL is affected mainly by the number of attributes and the size of their domain, while SRL is more impacted by the number of classes in a dataset, and the number of ants and iterations. It should be pointed out that the values for ants and classes both tend to be upper bounded by a small constant, however, it is certainly worth investigating possible ways to reduce the complexity and/or actual run time. The following subsections describe possible ways of making SRL cost more comparable with that of IRL.

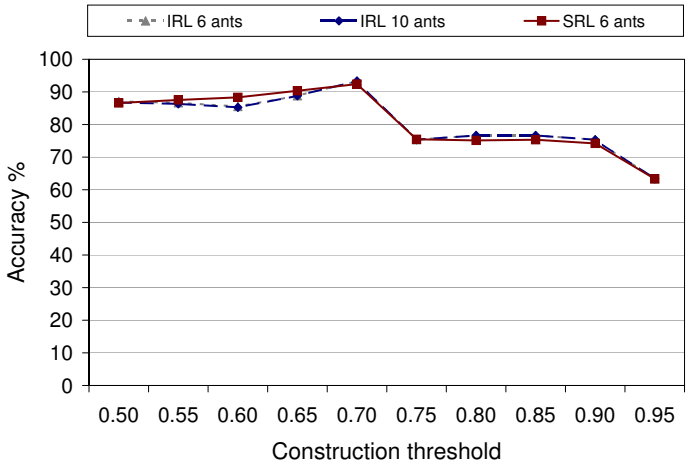
Fewer ants for *FRANTIC* SRL

Preliminary experiments suggest that *FRANTIC-simpSRL* may be able to induce rulebases with comparable accuracy results to those induced by *FRANTIC-simpIRL*, with fewer ants run within an iteration.

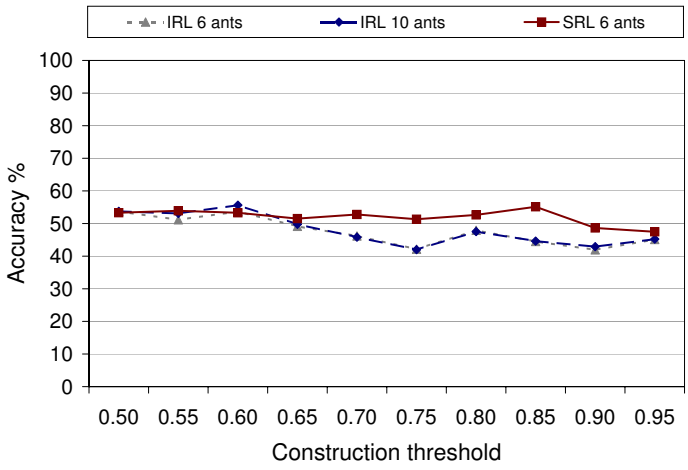
Figure 7.16 on the next page shows some of the results of additional experiments for both SRL and IRL, where the number of ants is set equal to 2 and 6, as opposed to the original value of 10. The rules are induced with the possibility of containing negated terms in their antecedent, and all other parameter settings are as before.



(a) Wine

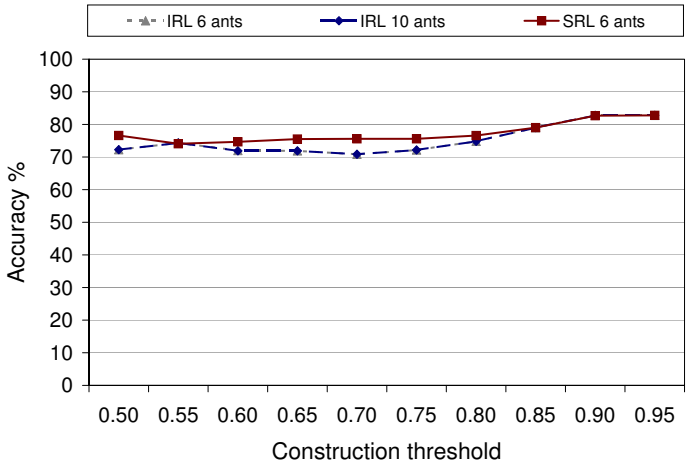


(b) Iris

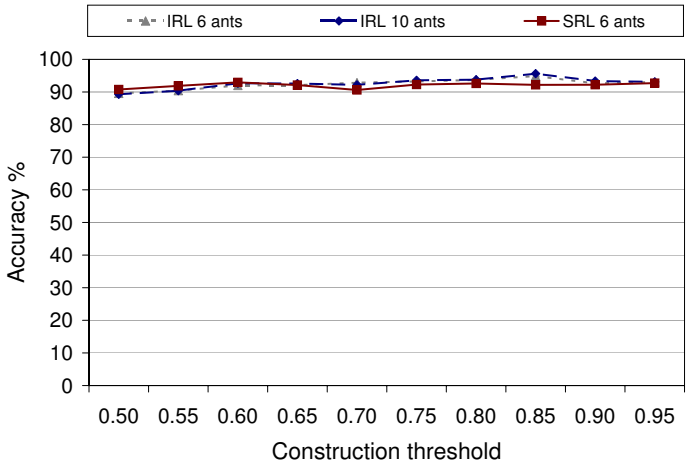


(c) Glass

Figure 7.16: Impact of *FRANTIC* rule learning strategy on number of ants per iteration



(d) Water Treatment



(e) Leukaemia

Figure 7.16: Impact of *FRANTIC* rule learning strategy on number of ants per iteration (cont.)

For all datasets, the accuracy obtained using fewer ants and SRL, is comparable to the accuracy obtained using more ants and IRL – the curves for IRL runs with 6 and 10 ants are often indistinguisable from each other and the accuracy attained is lower than the accuracy of rulebases induced using SRL with 6 ants. Furthermore, it should be remembered that the results for SRL are the same as those intially presented in Fig. 7.8 and Table D.7, i.e. where the rulebase selection method for SRL is not optimal, but relies entirely on pheromone levels and convergence to optimal or near-optimal solutions. It is therefore quite possible that better SRL results are obtainable with even fewer ants.

Though not actively reducing the number of ants that *create* a rule within an iteration, a useful observation is that in later iterations ants constructing rules to describe the same class often come up with the same rules in the same iteration. A simple check may therefore be made to evaluate only unique rules (note that the order of the antecedents is irrelevant) and rulebases. This is beneficial to individual rule evaluation in IRL by saving unnecessary time evaluating the same rules, but its impact is likely to be greater on rulebase evaluation in SRL, since removing only a few ants from each class leads to significantly fewer rulebases to evaluate.

Improved rulebase evaluation

There are several ways in which *FRANTIC* may be guided to conduct fewer rulebase evaluations during SRL, including ignoring duplicate rules and forming only unique rulebases as discussed in the preceding paragraph. Another possibility arises from the literature on multi-population co-operative co-evolution to induce both a rulebase and associated membership functions. Generally, only a few representatives from each population are used to form different knowledge bases (i.e. a rulebase and membership functions) that are to be evaluated. The representatives may be chosen according to fitness, randomly, or a combination of both.

Yet another possibility is a combined IRL and SRL evaluation process, where rulebase evaluations are conducted not at the end of *each* iteration, but every few iterations; individual rule evaluations are carried out during the iterations where rulebase evaluation is not. Both these ideas suggest a potentially useful avenue for further investigation into decreasing the computational expense of the SRL evaluation process, though it would change the current nature of the SRL strategy and rigorous experimentation and analysis will be required to determine and understand the consequent differences in the induced rulebases.

Research has been carried out to decrease the cost of matching rules to instances, which constitutes a major part of *FRANTIC*'s current SRL evaluation process. In [60], for instance, the author devises an algorithm for the fast matching of database samples to rules in a production system [141]. The production rules are transformed into a tree structure – called a Rete network

```
(1)    FOR p·k rules
(2)        FOR m instances
(3)            Determine degree of match between rule and instance
(4)        ENDFOR
(5)    ENDFOR
(6)    FOR p^k rulebases
(7)        FOR m instances
(8)            Classify instance
(9)        ENDFOR
(10)   ENDFOR
```

Figure 7.17: Alternative *FRANTIC* SRL rulebase evaluation process

– that eliminates redundancies (i.e. commonalities) in the antecedents of rules. Matching each instance to each rule therefore takes advantage of such commonalities in rule antecedents. Pan et al. [147] extend this work to the area of fuzzy matching and inferencing in expert systems. In [132] the authors propose a *Selective Inference Engine* that based on the sample to be processed, predicts the fuzzy rules that will be affected, and performs matching and inferencing with only these rules. In the area of Inductive Logic Programming [166] proposes a method that avoids considering all matchings between instances and the first-order logic hypothesis being constructed – it uses a stochastic sampling mechanism where the user decides on the number of instances to be sampled. It may be possible to utilise elements of such work in future *FRANTIC* modifications.

A more immediate *FRANTIC* modification that may yield substantial savings in run time, involves a simple change to the process of finding degrees of match between all rules and all instances during rulebase evaluation. The current implementation of SRL rulebase evaluation is as described in lines (13)–(20) of Fig. 7.15 – it is conceptually clear but involves much redundancy in determining degrees of match. This redundancy arises from two observations:

- any one rule participates in several rulebases that are evaluated, in p^{k-1} rulebases to be exact; and,
- such a rule has the same degrees of match with instances in the training set, irrespective of the rulebase it forms part of.

In the current rulebase evaluation implementation, therefore, a rule is matched with each instance in the training set $(p^{k-1}-1)$ more times than is necessary. An alternative to the current approach is outlined in Fig. 7.17. At most there are $p \cdot k$ unique rules produced within an iteration, p rules for each of k classes, and these need to be matched with all m instances in the training set, lines (1)–(5). This is of the order $O(p \cdot k \cdot m \cdot n \cdot d)$.

The degrees of match for each rule with each instance may be stored in a vector (one per rule). The classification process is now a simpler process of checking which of k rules in a rulebase has the highest degree of match with an instance, lines (6)–(10). There are p^k rulebases to be evaluated so that the classification process is of the order $O(p^k \cdot m \cdot k)$. There are still p^k operations to be carried out, but these operations are far simpler and less time consuming than matching all rules to each instances of the training set p^k times – the process of determining degrees of match is now carried out only $p \cdot k$ times.

Using this alternative approach, the complexity for SRL rulebase evaluation is therefore of the order:

$$O\left(p \cdot k \cdot m \cdot n \cdot d + p^k \cdot m \cdot k\right) \tag{7.11}$$

An empirical investigation is left for future work, which will also seek to determine the impact of the size of the training set, i.e. at what point p^k simple classifications may require less time than $p \cdot k$ full matchings between rules and the training set. This is potentially useful since k is generally a small integer, but the number of instances in a dataset may be quite large. Investigating the impact in practice of the number of attributes n in a dataset, and the average domain size d of an attribute will also provide useful information.

7.3.2 Improving Execution Time

This subsection looks at possible ways for improving *FRANTIC* run time that are applicable to both rule learning strategies, i.e. IRL and SRL.

Smaller graphs for rule construction

Reducing the time required to construct rules is expected to lead to a considerable saving in run time.

FRANTIC has a tendency to produce very short rules, i.e. only a few terms from the construction graph are eventually retained in a rule antecedent. Analysis of *FRANTIC* results indicates that the average number of terms per rule is fairly constant throughout all iterations. A procedure could therefore be implemented that determines the average number of terms within a rule for the first several iterations, and in subsequent iterations ants could stop building a rule antecedent when they have reached this average value. With this approach there is an added element of chance that is introduced in the rule construction procedure, since the average value used may occasionally restrict a longer but valid rule from being constructed. This may lead to

less accurate rules, or even act as a preventive measure for over-fitting and lead to better rules. This remains to be investigated.

A more immediate solution that does not change the current nature of the rule construction mechanism in any way involves the rule construction parameters. These parameters have two very important roles: to provide a balance between creating generalised or specialised rules, and, to make the search for solutions more tractable by restricting the parts of the hypothesis space that are searched. The latter role has not been exploited to full potential.

In the current *FRANTIC* implementation *all* terms form part of the construction graph, and all terms have a chance of being selected during rule construction[7]. However, the size of the construction graph may be reduced by removing terms that have *no* chance of being retained in the rule antecedent being constructed by an ant – these are terms that do not satisfy the `minInstPerRule` and `constructionThreshold`.

A pre-processing step could scan the training set and identify terms that individually meet both criteria. Consider for example a fuzzy term *Petal_Length is Low* – for this term to be part of the construction graph for creating rules to describe a particular class, its fuzzified value in the training set needs to be equal to or greater than the value specified by `constructionThreshold`, and, this needs to occur in at least the number of instances specified by `minInstPerRule`, where these instances are also of the same class as the rules to be constructed.

In essence, this step produces the rule construction term set (one for each class), and this term set may therefore be used to initialise the construction graph (for each class). It should be remembered that not all terms in the construction set will be present in a rule antecedent; the pre-processing only identifies terms that meet the construction criteria individually – for a term to be retained in the rule antecedent, it needs to be able to satisfy the construction criteria *together* with terms already present in the antecedent.

It is difficult to predict the size of the resulting construction sets. However, an idea of the potential run time gains of this approach may be obtained from Table 7.10 on the following page. This table shows the size of the construction term sets for the Leukaemia dataset, for constructing rules with negated terms and linguistic hedges, for different `minInstPerRule` and `constructionThreshold` settings. The total number of terms is six hundred[8], and it is clear that a significant gain is possible in reducing the time necessary to create a rule, by instead using a graph consisting only of terms that have a chance of being retained in a rule antecedent. The

[7] The only reason a term is not selected, is if another term from the same attribute domain value has been selected and *retained* in the antecedent. This is valid for all forms of the hypothesis language used apart from rules with internal disjunction – in this form, the solution validation method ensures all terms are selected during rule construction

[8] There are 50 attributes in the Leukaemia dataset, and each is defined by three fuzzy sets. For each fuzzy set, say, *low*, there are additional terms *Not_low*, *Very_low*, and *More-or-less_low*. Each attribute therefore is represented by 12 nodes in the construction graph.

Table 7.10: Size of construction term sets for the Leukaemia dataset for different `minInstPerRule` and `constructionThreshold` settings. The construction sets are for inducing rules with negated terms and linguistic hedges, and the total number of terms in the original construction graph is 600.

	Class 1			Class 2		
`construction`	`minInstPerRule`			`minInstPerRule`		
`Threshold`	30%	50%	70%	30%	50%	70%
0.50	388	229	105	355	248	136
0.55	349	195	81	339	234	122
0.60	322	159	67	326	220	112
0.65	291	142	56	309	205	98
0.70	258	114	48	295	191	83
0.75	207	85	33	281	170	71
0.80	172	69	30	268	147	57
0.85	134	63	30	252	132	49
0.90	103	60	27	233	95	44
0.95	91	55	27	208	71	39

full exploitation of the construction parameters in this way shows a potential for tackling very large problems such as those in bioinformatics – a very large number of attributes (e.g. genes) need not necessarily mean an intractable construction graph.

There is potentially another advantage to initialising construction graphs based solely on the terms in the construction set. As discussed in Section 7.2.1 on page 149, during the construction of rules with internal disjunction additional selected terms that are from the same domain of a term that is already in the rule antecedent will be retained. This means that an additional term from the same domain need not necessarily – individually or otherwise – satisfy the criteria, and if not, may be considered a spurious addition, i.e. making no real contribution to the knowledge being conceptualised by the rule. If only terms from the construction set are used during construction, and if an additional term from the same domain is selected, one can be certain that at least it meets the construction criteria on an individual basis. Whether this leads to improved or worse rulebases is left for future work to determine.

Convergence criterion for stopping an ACO algorithm

It may be possible in certain situations to dynamically reduce the number of iterations run within an ACO. *Ant-Miner* [148] has a feature by which an ACO algorithm stops if a certain

user-specified number of successive iterations produce identical rules (the population size in each iteration is one). If this does not occur, then the ACO stops when it has completed a pre-specified maximum number of iterations.

In *Ant-Miner* an iteration runs only one ant, but this same concept may be adapted for *FRANTIC* – an ACO may stop running iterations if the best rule from each of b successive iterations is identical. For the set system parameters and for these datasets, convergence is not observed when rules with internal disjunction are induced, though note that this does not necessarily detract from finding reasonably accurate rulebases when the rulebase selection method is amended. However, for the other forms of rule that may be induced, analysis of *FRANTIC* results indicate that convergence may be used as an additional stopping criterion, thereby preventing the execution of unnecessary iterations.

Table 7.11 on the next page shows the convergence results for rulebases induced using the simplified IRL and SRL strategies. These convergence results are for rulebases whose accuracies are presented in Tables C.6 and D.7 for IRL and SRL rulebases respectively. The convergence iteration values for IRL induced rulebases are determined according to formula (6.1). Convergence for SRL induced rulebases occurs if and when *all* rulebases produced within an iteration have the same fitness value, and, this continues in all subsequent iterations until the final iteration is completed. The average SRL iteration convergence value is given by:

$$conV\_SRL(D,m) = \frac{1}{m}\sum_{j=1}^{m} itn_j \tag{7.12}$$

where itn_j is the iteration number at which convergence starts for run j. As for IRL convergence, a run is defined as processing one fold of a k-fold cross-validation test, so that if v k-fold cross-validation tests are carried out, $j = v \cdot k$. Again, the parameter settings are assumed to be the same over all runs.

The number in brackets beside each statistic in Table 7.11 gives the number of ten-fold cross-validation tests (out of a total of ten ten-fold cross-validation tests) that have converged. All ten individual folds processed within a particular ten-fold cross-validation test need to each have converged; if one fold within a ten-fold cross-validation does not achieve convergence, then that ten-fold cross-validation is not included in the statistic. ' – ' instead of a convergence value indicates that none of the ten ten-fold cross-validation tests have converged.

For both IRL and SRL induced rulebases, convergence generally occurs for all forms of the hypothesis language apart from rules with internal disjunction. As discussed earlier, though the construction graph for this form is smaller than for other forms, the solution validation method allows ants constructing rules with internal disjunction to search a larger part of the hypothesis space. Since more exploration is happening, more ants and/or more iterations are required for convergence to occur. Note that in line with this thinking, convergence values for

Table 7.11: Convergence values for simplified IRL and SRL *FRANTIC* induced rulebases. Figures in brackets denote the number of ten-fold cross-validations, out of ten ten-fold cross-validation tests, that have converged.

(a) Wine

construction	simplified IRL				simplified SRL			
threshold	Simple	Negation	Hedges	Disjunct.	Simple	Negation	Hedges	Disjunct.
0.50	10.7 (10)	20.5 (10)	24.4 (10)	– (0)	12.2 (10)	19.6 (9)	21.8 (8)	– (0)
0.55	9.3 (10)	18.9 (10)	21.9 (10)	– (0)	11.2 (10)	18.7 (10)	20.5 (8)	– (0)
0.60	9.3 (10)	16.8 (10)	20.1 (10)	– (0)	10.3 (10)	17.5 (10)	19.3 (10)	– (0)
0.65	9.1 (10)	15.8 (10)	18.7 (10)	– (0)	10.0 (10)	17.9 (10)	19.8 (9)	– (0)
0.70	8.6 (10)	15.8 (10)	17.8 (10)	– (0)	9.4 (10)	16.8 (10)	19.0 (9)	– (0)
0.75	8.8 (10)	14.8 (10)	16.7 (10)	– (0)	9.9 (10)	16.9 (10)	19.1 (9)	– (0)
0.80	5.8 (10)	13.9 (10)	15.1 (10)	– (0)	8.4 (10)	16.1 (9)	17.6 (10)	– (0)
0.85	4.7 (10)	13.5 (10)	14.4 (10)	39.0 (1)	7.9 (10)	15.3 (10)	17.0 (10)	– (0)
0.90	1.7 (10)	12.6 (10)	13.6 (10)	18.2 (4)	6.4 (10)	14.3 (10)	16.2 (10)	– (0)
0.95	1.4 (10)	12.2 (10)	13.0 (10)	10.6 (7)	7.5 (10)	14.6 (10)	16.4 (10)	– (0)

(b) Iris

construction	simplified IRL				simplified SRL			
threshold	Simple	Negation	Hedges	Disjunct.	Simple	Negation	Hedges	Disjunct.
0.50	4.2 (10)	11.5 (10)	14.4 (10)	– (0)	7.4 (10)	12.7 (10)	15.9 (10)	– (0)
0.55	4.3 (10)	10.7 (10)	13.7 (10)	– (0)	7.4 (10)	12.7 (10)	15.9 (10)	– (0)
0.60	3.4 (10)	10.9 (10)	13.6 (10)	– (0)	6.4 (10)	12.5 (10)	14.3 (10)	– (0)
0.65	3.2 (10)	10.9 (10)	11.6 (10)	– (0)	6.9 (10)	12.4 (10)	13.8 (10)	– (0)
0.70	2.9 (10)	10.1 (10)	11.4 (10)	– (0)	6.3 (10)	12.2 (10)	12.9 (10)	– (0)
0.75	1.9 (10)	10.3 (10)	10.5 (10)	– (0)	7.2 (10)	12.9 (10)	12.4 (10)	– (0)
0.80	1.8 (10)	9.9 (10)	10.2 (10)	– (0)	7.5 (10)	12.8 (10)	12.2 (10)	– (0)
0.85	1.6 (10)	7.7 (10)	9.7 (10)	– (0)	7.1 (10)	12.4 (10)	12.4 (10)	– (0)
0.90	1.2 (10)	6.6 (10)	7.7 (10)	9.3 (8)	6.1 (10)	12.0 (10)	14.1 (10)	– (0)
0.95	1.0 (10)	6.5 (10)	6.9 (10)	8.7 (10)	2.2 (10)	10.4 (10)	11.6 (10)	– (0)

(c) Glass

construction	simplified IRL				simplified SRL			
threshold	Simple	Negation	Hedges	Disjunct.	Simple	Negation	Hedges	Disjunct.
0.50	13.5 (10)	21.5 (9)	25.4 (8)	– (0)	17.9 (10)	25.5 (4)	– (0)	– (0)
0.55	12.7 (10)	20.3 (10)	24.5 (8)	– (0)	18.1 (10)	23.6 (4)	26.3 (1)	– (0)
0.60	8.6 (10)	17.7 (10)	21.2 (10)	– (0)	20.4 (9)	25.2 (4)	25.9 (2)	– (0)
0.65	8.5 (10)	16.1 (10)	19.4 (10)	– (0)	20.1 (9)	22.7 (8)	– (0)	– (0)
0.70	6.0 (10)	15.6 (10)	18.8 (10)	– (0)	18.8 (9)	21.8 (4)	25.6 (5)	– (0)
0.75	5.4 (10)	15.2 (10)	17.3 (10)	– (0)	– (0)	22.3 (7)	24.7 (1)	– (0)
0.80	2.0 (10)	14.4 (10)	16.4 (10)	– (0)	20.1 (1)	21.7 (8)	23.4 (6)	– (0)
0.85	1.1 (10)	14.7 (10)	16.3 (10)	– (0)	11.5 (4)	21.8 (6)	24.0 (7)	– (0)
0.90	1.5 (10)	15.1 (10)	16.1 (10)	– (0)	19.0 (4)	21.8 (4)	23.0 (5)	– (0)
0.95	1.3 (10)	14.5 (10)	15.7 (10)	– (0)	26.8 (1)	23.6 (10)	24.6 (2)	– (0)

Table 7.11: Convergence values for simplified IRL and SRL *FRANTIC* induced rulebases. Figures in brackets denote the number of ten-fold cross-validations, out of ten ten-fold cross-validation tests, that have converged (cont.)

(d) Water Treatment

construction	simplified IRL				simplified SRL			
threshold	Simple	Negation	Hedges	Disjunct.	Simple	Negation	Hedges	Disjunct.
0.50	5.6 (10)	13.4 (10)	14.6 (10)	– (0)	7.6 (10)	12.7 (10)	12.6 (10)	– (0)
0.55	5.7 (10)	12.4 (10)	13.6 (10)	– (0)	7.4 (10)	12.3 (10)	12.4 (10)	– (0)
0.60	5.5 (10)	13.1 (10)	13.9 (10)	– (0)	7.4 (10)	11.4 (10)	12.8 (10)	– (0)
0.65	5.7 (10)	13.1 (10)	14.8 (10)	– (0)	7.6 (10)	11.6 (10)	13.1 (10)	– (0)
0.70	3.5 (10)	10.8 (10)	14.3 (10)	– (0)	3.8 (10)	9.7 (10)	10.4 (10)	– (0)
0.75	3.2 (10)	11.3 (10)	14.5 (10)	– (0)	3.6 (10)	9.4 (10)	10.9 (10)	– (0)
0.80	3.3 (10)	7.1 (10)	14.5 (10)	– (0)	3.8 (10)	8.1 (10)	10.2 (10)	– (0)
0.85	3.3 (10)	4.8 (10)	11.4 (10)	– (0)	3.6 (10)	6.2 (10)	8.6 (10)	– (0)
0.90	3.2 (10)	4.7 (10)	5.4 (10)	– (0)	3.4 (10)	5.8 (10)	7.0 (10)	– (0)
0.95	2.9 (10)	4.6 (10)	15.7 (10)	– (0)	3.1 (10)	5.6 (10)	6.6 (10)	– (0)

(e) Leukaemia

construction	simplified IRL				simplified SRL			
threshold	Simple	Negation	Hedges	Disjunct.	Simple	Negation	Hedges	Disjunct.
0.50	2.7 (10)	5.0 (10)	5.0 (10)	– (0)	7.5 (10)	15.9 (10)	15.9 (10)	– (0)
0.55	1.3 (10)	12.6 (10)	14.7 (10)	5.7 (1)	8.1 (10)	14.0 (10)	14.0 (10)	– (0)
0.60	1.4 (10)	11.7 (10)	13.8 (10)	5.3 (6)	8.4 (10)	14.1 (10)	14.1 (10)	– (0)
0.65	1.3 (10)	12.2 (10)	13.6 (10)	5.3 (5)	6.5 (10)	12.8 (10)	12.8 (10)	– (0)
0.70	1.6 (10)	12.5 (10)	13.4 (10)	4.9 (9)	4.3 (10)	13.3 (10)	13.3 (10)	– (0)
0.75	1.0 (10)	12.1 (10)	12.4 (10)	2.9 (10)	4.5 (10)	12.1 (10)	12.1 (10)	– (0)
0.80	1.0 (10)	11.3 (10)	12.2 (10)	1.0 (10)	5.7 (10)	11.0 (10)	11.0 (10)	– (0)
0.85	1.0 (10)	10.7 (10)	11.0 (10)	1.0 (10)	6.9 (10)	11.6 (10)	11.6 (10)	– (0)
0.90	1.0 (10)	10.5 (10)	10.3 (10)	1.0 (10)	7.2 (10)	11.6 (10)	11.6 (10)	– (0)
0.95	1.0 (10)	10.3 (10)	10.4 (10)	1.0 (10)	5.7 (10)	11.9 (10)	11.9 (10)	– (0)

rules with negated terms are generally higher than those for simple propositional rules, and convergence values for rules with negated terms and linguistic hedges are higher still. It is clear that the larger the hypothesis space, the greater the number of iterations that are required – the more pheromone updates that are necessary – so that ants may converge to an optimal or near-optimal solution.

Another point to note is that generally, for each form of the hypothesis language, SRL induced rulebases appear to converge later than IRL induced rulebases. It should be remembered, though, that the way these values are determined are different – the convergence iteration number given for an IRL induced rulebase is actually the average of the convergence iteration numbers for the individual classes within the dataset. The value given for an SRL induced rulebase is when for all classes, all ants for a particular class construct the same rule. A more equitable comparison then, to see whether one or the other strategy generally comes up with an optimal solution faster than the other, might be to use the convergence iteration number of the class that converges the latest, for IRL rulebases. Reanalysing IRL rulebase results in this way indicates that the rate of convergence is more equal – for some datasets SRL rulebases appear to converge faster, and for others IRL rulebases appear to have the upper hand.

System parallelisation

Another advantage offered by *FRANTIC* that should not be ignored is the obvious and numerous opportunities for parallelisation. Research in the design of faster ACO implementations has been increasing in recent years. The work is generally in the context of applying ACO to the more standard travelling salesman or quadratic assignment problems, though some is applicable to rule induction. Research in this area falls into two main categories: hardware-based approaches to improving computational speed, and software-based approaches.

Work in the first category may yield substantial run time savings, achieving close to linear complexity in some cases [133; 142]. However, this approach is often limited by memory, computational and I/O resources of the devices used, and therefore also tends to subtly – or not so subtly – change the nature of the standard ACO. For instance, in [142], a reconfigurable mesh [19] (a grid of processing elements whose connections with each other are reconfigurable at run time), is used to represent a pheromone matrix, and ants are ‘pipelined’ in close succession through this mesh (i.e. an ant need not complete its solution construction before the next ant starts building its own). However, in order to achieve an even faster implementation the authors make significant changes to the pheromone update strategy, and the decision making process employed in selecting solution components. Similarly in [165], a considerable decrease in computational time is achieved by using a field-programmable gate array (an integrated cir-

cuit that can be programmed in the field after manufacture), to represent not the pheromone matrix but the current global best *k* ants that are used *instead* of pheromones to guide solution construction.

Though hardware-based approaches are potentially extremely useful, a software approach is a more immediately viable first-step for future *FRANTIC* development, since it provides the advantage of allowing a parallel implementation that is conceptually faithful to the sequential implementation. An overview of ACO parallelisation strategies for the standard applications is provided in [160], [23] discusses a fine-grain implementation at the ant level, and various multiple ant colony implementations are discussed in [71; 98; 134].

The multi-colony work just cited makes the distinction between implementations that have homogeneous or heterogeneous colonies. In the former type each colony (which may be running on a separate processor), produces solutions for exactly the same problem, with regular interchange of solutions between colonies (i.e. similar to island-GAs). In the latter type each colony produces solutions that are biased towards meeting a specific criterion, so for instance, colony *A* may be constructing vehicle routing solutions where the main aim is minimisation of the number of vehicles, whilst colony *B* constructs vehicle routing solutions where the main aim is minimisation of the distance travelled [71]. In both these types of multi-colony implementation note that each ant in each colony constructs a *complete* solution, though solutions from different colonies may be biased in different ways. *FRANTIC* colonies, however, each create one *component* (a rule), of a solution (a rulebase), and so this work on multi-colony implementation has limited applicability to *FRANTIC*.

Concepts from ACO parallelisation work that are immediately applicable lie in the area of fine-grain parallelisation, and apply to both *FRANTIC* IRL and SRL modes of operation. These include the use of multiple processors for rule construction within an iteration, i.e. ants within the same iteration need not construct their solution one after each other, but simultaneously, depending on the availability of processors. Another possible application of multiple processors is during rule or rulebase evaluation where again, the work may be shared amongst available processors.

For simplified and full IRL, a clearly natural coarse-grain implementation consists of having one processor run a colony of ants, with each colony focusing on finding rules to describe a particular class. Each processor has its own copy of the dataset, and maintains its own construction graph and pheromone matrix. No communication between the colonies is required until *all* have produced their rules to add to the final rulebase, at which point a process collates these rules from each colony. For simplified IRL each processor would run merely one ACO to produce one rule to describe a class. For full IRL each processor would run as many ACOs as are dictated by the construction parameters and the `maxClassInstUncovered` parameter. A

Table 7.12: Comparison of *FRANTIC* rulebases induced using simplified (*simpIRL*) and full (*fullIRL*) iterative rule learning, and simplified simultaneous rule learning (*simpSRL*)

(a) Classification accuracy (%) and standard error of the mean (+/-)

	simpIRL		*fullIRL*		*simpSRL*	
	%	+/-	%	+/-	%	+/-
Wine	93.01	0.77	93.89	1.22	93.07	1.30
Iris	95.33	0.00	95.33	0.00	96.00	0.00
Glass	55.63	1.64	58.41	2.21	61.81	1.28
WT	90.42	0.62	93.07	0.00	85.25	0.00
Leuk	95.63	1.55	95.07	1.72	94.29	1.29

(b) Number of rules and number of terms in a rule

	simpIRL		*fullIRL*		*simpSRL*	
	Rules	Terms	Rules	Terms	Rules	Terms
Wine	3.00	5.61	6.00	7.58	3.00	5.48
Iris	3.00	2.00	3.40	2.03	3.00	1.67
Glass	6.00	6.58	12.00	7.67	6.00	6.55
WT	2.00	1.44	4.00	3.23	2.00	2.00
Leuk	2.00	10.74	2.82	3.65	2.00	12.62

similar approach to a parallel implementation of ACO-based crisp rule induction is described in [37]; the authors compare their results on several datasets with a modified *Ant-Miner* and *C4.5* with respect to accuracy and model complexity, but make no enlightening comments on any run time gains.

For SRL different colonies may be run on different classes. However, since evaluation is based on rulebases composed of rules describing different classes (which are being constructed on separate colonies), at the end of each iteration a central process needs to collate rules, form rulebases and finally evaluate them. Naturally, the evaluation of rulebases may also be performed by multiple available processors. System parallelisation and a detailed empirical and analytical investigation into scaling-up to large problems is one of the next major development steps for *FRANTIC*. Other development areas are discussed in the next chapter.

7.4 Summary

Table 7.12 is an amalgamation of Tables 7.2 and 7.7. It gives the highest average accuracy attained by each variant and mode of *FRANTIC*, with corresponding complexity statistics.

Additional rules in a rulebase to describe each class result in an increased robustness to the con-

`structionThreshold` and `representation` parameters, though the improvement in accuracy is not always so obvious. This is thought to be due at least partly to the learning strategy that does not take into account rules already in the rulebase when adding new ones. Another contributing factor may be the current way in which the `minInstPerRule` parameter is changed in between ACO runs, and which may be leading to the construction of later rules that over-fit the training data. A way of redefining the automatic updating of this parameter has been suggested, and in Section 9.2 suggestions are made for future work to explore the limits to which an IRL strategy may be taken (and in that way one may make further comparisons with an SRL approach).

A common approach to dealing with competing rules is to conduct a post-processing step to resolve rule conflicts and even eliminate redundant rules (i.e. rules that may be subsumed by more general ones). This work has investigated a different approach that seeks to resolve such issues during the rulebase induction process, by simultaneously constructing and evaluating rules that describe the different classes in a dataset. As seen from Table 7.12 simplified SRL can provide an improved accuracy and at no additional cost to rulebase size. It should be remembered that the statistics for SRL in this table are based on rulebases selected from the final iterations of *FRANTIC* runs, and as demonstrated in Section 7.8, this is a sub-optimal rulebase selection mechanism – it relies entirely, and unnecessarily so, on pheromone levels for convergence to optimal or near-optimal solutions. It is a simple matter to take advantage of any rulebase constructed in any iteration of an ACO, that might actually have a higher fitness than the rulebase the system converges to, by choosing the best rulebase of all iterations, and not merely the best of the final iteration (i.e. similar to the approach adopted for rule selection in IRL). Initial experiment results presented here suggest this is a very viable approach.

The SRL strategy also benefits from having one fewer parameter to tune than for simplified IRL (no `fitnessThreshold`), two fewer parameters than for full IRL (`removalThreshold` and `maxInstClassUncovered`), and improves the robustness to `constructionThreshold` settings. However, it does come at an increased computational expense (part of which at least, might have been expended on post-processing steps to resolve rule conflicts). Section 7.3 analyses the complexity of all *FRANTIC* variants and modes, and investigates ways in which run time costs may be reduced. As well as the obvious system parallelisation course that may be adopted, there are two immediately viable strategies that are likely to yield substantial results: reducing rule construction time by following a construction graph initialised on pre-processed term sets (valid for both IRL and SRL), and introducing a more efficient process for finding degrees of match between instances and rules (valid for both strategies though the impact is expected to be greater for SRL since a greater number of evaluations is carried out).

Other ideas suggested for reducing the cost of computation are likely to have a knock-on effect that in practice might also yield considerable time savings. With regards to the SRL strategy,

fewer ants required to produce reasonable rulebases means far fewer rulebases to evaluate at the end of each iteration. Of relevance to both strategies, the use of a convergence stopping criterion means fewer iterations that might be run, and consequently, fewer ants constructing rules, and fewer rules and rulebases to be evaluated.

This chapter has indicated several opportunities for *FRANTIC* development, whether it be to improve the accuracy of the induced rulebases (whilst maintaining the high level of comprehensibility), or to reduce computational costs. A major enhancement not mentioned here is that of providing *FRANTIC* with the capability of inducing more than one rule to describe a class, if necessary, while following an SRL strategy. Possibilities for this, and other future work of interest, are presented in the final chapter of this thesis. The next chapter analyses how *FRANTIC* fares when pitted against established learning algorithms such as *C4.5* and support vector machines.

Chapter 8

Comparisons with Other Algorithms

This chapter compares *FRANTIC*'s performance with that of several other well-established learning algorithms over real-world datasets with different properties.

The *FRANTIC* rulebases are selected from the simplified SRL induced rulebases discussed in Chapter 7, and are the ones leading to the highest classification accuracies. A few full IRL rulebases presented in the same chapter achieve a higher classification accuracy for the Wine and Water Treatment datasets (at the cost of an increase to rulebase size), and some simplifed IRL induced Leukaemia rulebases from Chapter 6 also achieve a higher classification accuracy. The accuracy increase is not significant, however, and a decision is taken to limit the selection of rulebases from those induced by only one strategy. The SRL strategy is selected, primarily because it is the one originally expected to produce the best results with regards to both accuracy and rulebase complexity, but also to enable a more equitable comparison with other algorithms whose parameters may not have been extensively tuned. It is therefore both prudent and expedient to make explicit the extent of parameter tuning that has been carried out for each of the algorithms presented in this chapter.

With regards to *FRANTIC*, the four variations on the knowledge represention are each tried with `constructionThreshold` set to different settings (in the range $[0.5, 0.95]$ with a step value of 0.05), for one setting of `minInstPerRule`. For one variation of the knowledge representation (rules that contain negated terms), two other `minInstPerRule` settings are tried, with again different settings for `constructionThreshold`. The other parameters (including number of iterations and number of ants) are set at their 'default' values throughout. Details of how the parameter settings are determined are explained in Section 5.5, while the specific settings that lead to the rulebases whose results are discussed in this chapter are presented in Table B.4 on page 221.

It is worth highlighting that these *FRANTIC* settings have been determined for inducing rule-

bases using the simplified *IRL* strategy, and not the SRL one. It is reasonable to assume that different settings are necessary for a different strategy, and one example to support this assumption is that the better SRL induced rulebases for the Glass dataset are the ones with `minInstPerRule=70%`, while the better simplified IRL induced rulebases are ones where `minInstPerRule=50%`; it is quite likely that more optimal settings exist for the SRL approach.

The algorithms against which *FRANTIC* is compared are introduced in Section 5.1 on page 72 and described in more detail (including the determination of parameter settings) in Section B.2 on page 215. The next few paragraphs briefly describe the parameter tuning. Most algorithms are run several times under different conditions, and the best results with regards to classification accuracy are the ones reported in this chapter. If more than one run of an algorithm achieves the same highest accuracy, then the best results with regards to model complexity (if applicable) are used. Parameters not explicitly mentioned are set at the default values provided by the implementation of an algorithm, and the conditions tested that lead to the results discussed in this chapter are given in Table E.2 on page 267.

C4.5 is run with the default value of two for the minimum number of instances to match a leaf, and non-binary splits on attributes are allowed. The algorithm is run without pruning, with *C4.5* pruning, and with reduced-error pruning. Reduced-error pruning involves randomly splitting the training data and keeping one fold for evaluating pruning steps on the tree grown by the other folds – the value for the number of folds is the default of three. Reduced-error pruning offers the advantage of better estimating pruning effects through use of a separate subset of the dataset, but fewer training instances are used to build the tree.

Naïve Bayes (*NB*) generally handles numeric attributes by assuming that the values follow a normal distribution. The *Weka* implementation of *NB* offers an alternative which is to use a kernel density estimator when no assumptions may be made about attributes. *NB* was run on each dataset once using a normal distribution assumption, and once using a kernel density estimator. The particular *Weka* support vector machine implementation utilised is *SMO* (Sequential Minimal Optimization), a fast method for training SVMs. No normalisation of the data is carried out, but two different kernels are used to determine the maximum margin hyperplane for each dataset – a polynomial kernel and a radial basis function.

Four algorithms generate fuzzy rulebases or fuzzy decision trees. *QSBA* is a deterministic algorithm for inducing rules that has no parameters to be set. *FSBA* has two parameters to set, and these were tuned by varying their values in the range $[0.5, 1.0]$ using a step value of 0.05. The other two fuzzy induction algorithms – *NECLASS* and *FID3.4* – each have several parameters that specify how the fuzzy inference process is to be carried out. These parameters are deliberately set so as to make the inferencing process of all fuzzy algorithms equivalent, and this is described in Section 5.4.2 on page 81.

The *NECLASS* system is further allowed to determine the optimum number of initial rules (as this option is suggested to optimise accuracy), fuzzy sets are tuned, and then all available pruning steps are applied (each followed by further fuzzy set optimization). The rule learning procedure parameter is set at the recommended 'best per class' (as it guarantees each class is covered by approximately the same number of rules, i.e. independently of the class distribution).

As well as providing the user with the facility to specify how inference is to be carried out, the *FID3.4* implementation also provides several parameters that control the tree-building process, and for pruning trees in order to reduce their size and improve generalisation capability on test datasets. The parameters that define the fuzzy operators used in determining the best attribute on which to split during tree building are set to 'best' – i.e. at each node to be split the fuzzy operator used at that node is the one that results in the highest value for the splitting criterion. Different levels of pruning are also tried for each dataset – no pruning, and then gradually increasing the stringency of the pruning mechanism and setting the relevant parameter to values of 0, 0.5 and 1 (possible range for this parameter being [0,1]).

Care has been taken to replicate the training environment between the algorithms (as described in Sections 5.3 and 5.4). However, the parameter tuning for some of the algorithms has not necessarily been as extensive as that for others, including *FRANTIC*, *FSBA*, and *QSBA* (which has no parameters to be set), and this should be borne in mind when considering the following results. Section 8.1 compares the algorithms' performance on any one dataset with respect to the accuracy achieved by the induced models. For the imbalanced Water Treatment and Leukaemia datasets it explores a model's performance with respect to accurately describing and predicting the minority class of these datasets. Section 8.2 presents results relating to the complexity of the induced models. These two sections also provide a summary of an individual algorithm's performance over *all* datasets, and these summaries are collated and discussed in the final section, where both major criteria – accuracy and comprehensibility – are taken into account.

8.1 Model Accuracy

Figure 8.1 on page 184 compares the performance of the models induced by all the algorithms on all the datasets, with regards to the classification accuracy. *NEFC* denotes results of models induced by *NEFCLASS*, and *FRAN* those of models induced by *FRANTIC*; all other system names are as introduced in Section 5.1. The error bars on each graph denote the standard deviation obtained from the ten accuracies produced by models from a ten-fold cross-validation run (or ten ten-fold cross-validations in the case of *FRANTIC* results); this gives an indication

of the robustness of an algorithm to different training sets. The results that generated the subfigures in Fig. 8.1, including estimates for the standard error of the mean, are presented in Table E.1 on page 267. In order to facilitate a comparison, each subfigure illustrates the performance of all algorithms on one of the datasets.

The error bars on each graph denote the standard deviation obtained from the ten accuracies produced by models from a ten-fold cross-validation run (or ten ten-fold cross-validations in the case of *FRANTIC* results). The results that generated the subfigures in Fig. 8.1 are presented in Table E.1 on page 267. In order to facilitate a comparison, each subfigure illustrates the performance of all algorithms on one of the datasets.

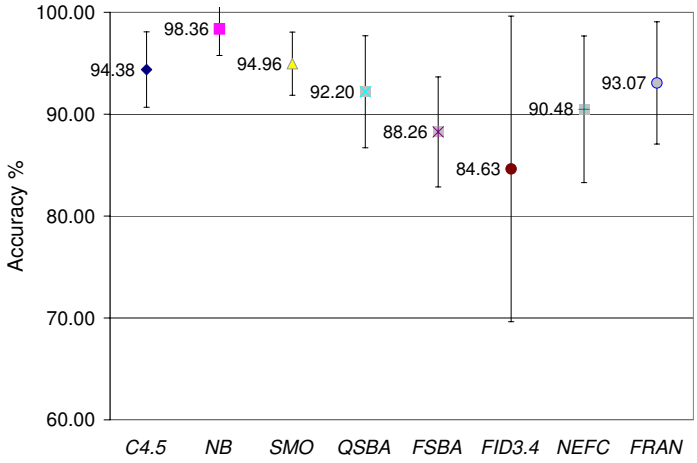
All classes of the Wine dataset are linearly separable, while one class of the Iris dataset is not linearly separable from the other two. All algorithms achieve a good to high performance on these two datasets (Figs. 8.1(a) and 8.1(b)), though standard deviations for accuracies on the Wine dataset are on the whole lower, which perhaps may be attributed to the separability of the classes. With regards to the six-class Glass dataset, all algorithms achieve a much lower classification accuracy than on the other datasets (Fig. 8.1(c)).

The performance of the algorithms on the Water Treatment dataset is very mixed (Fig. 8.1(d)) – four of the algorithms achieve a low to reasonable accuracy with a noticeable standard deviation, while the remaining four algorithms achieve a very high accuracy with negligible standard deviation. This is an extremely imbalanced dataset that poses problems for most learning algorithms, and the results depicted in Fig. 8.1(d) are investigated in detail in the following subsection, along with the results of the Leukaemia dataset which is another highly imbalanced dataset. The Leukaemia dataset has another notable feature in that it has an almost equal proportion of attributes to training instances. Yet, all models achieve a reasonably high accuracy, though it should be observed that in most cases the standard deviation is very high (Fig. 8.1(e)).

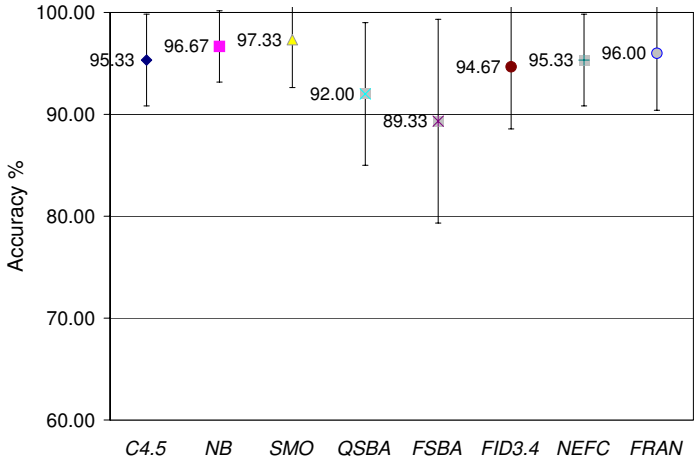
8.1.1 Performance on Imbalanced Datasets

In order to determine how effective an algorithm is in inducing models that can describe and accurately predict a minority class in a dataset, the additional metrics of True Positive Rate (*TP_Rate*) and False Positive Rate (*FP_Rate*) are used. In this section, only the two-class Water Treatment and Leukaemia datasets are considered, partly because it is easier to analyse and visualise results for a two-class problem, and partly because these are the main datasets with an uneven class distribution – the Iris dataset has an exactly equal class distribution, the Wine dataset has an approximately equal distribution, and the Glass dataset has several classes, all of which may be approximately equal to some classes but not to others.

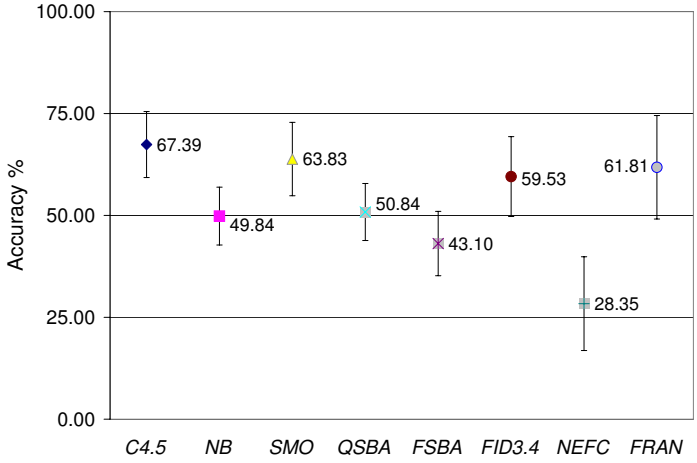
The positive (minority) class of the Leukaemia dataset is the one describing acute myeloid leu-



(a) Wine

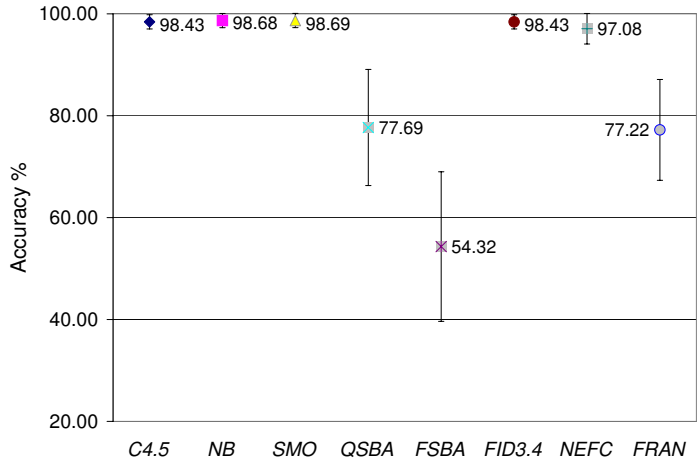


(b) Iris

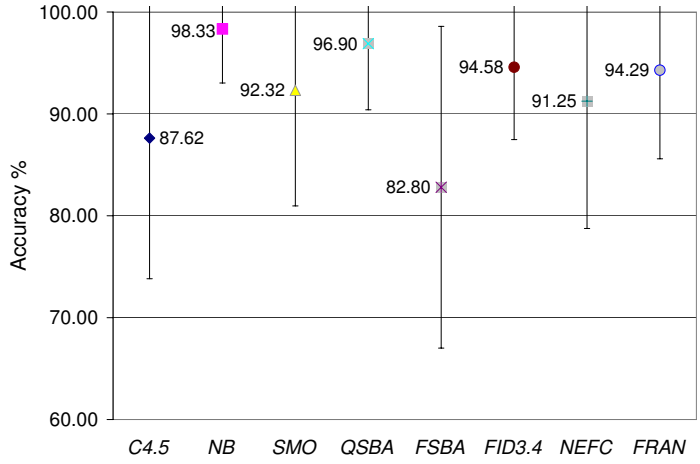


(c) Glass

Figure 8.1: Classification accuracy of models produced by all algorithms on datasets. Error bars denote standard deviation of the accuracies of models produced by a 10-fold x-validation.



(d) Water Treatment



(e) Leukaemia

Figure 8.1: Classification accuracy of models produced by all algorithms on datasets. Error bars denote standard deviation of the accuracies of models produced by a 10-fold x-validation (cont.)

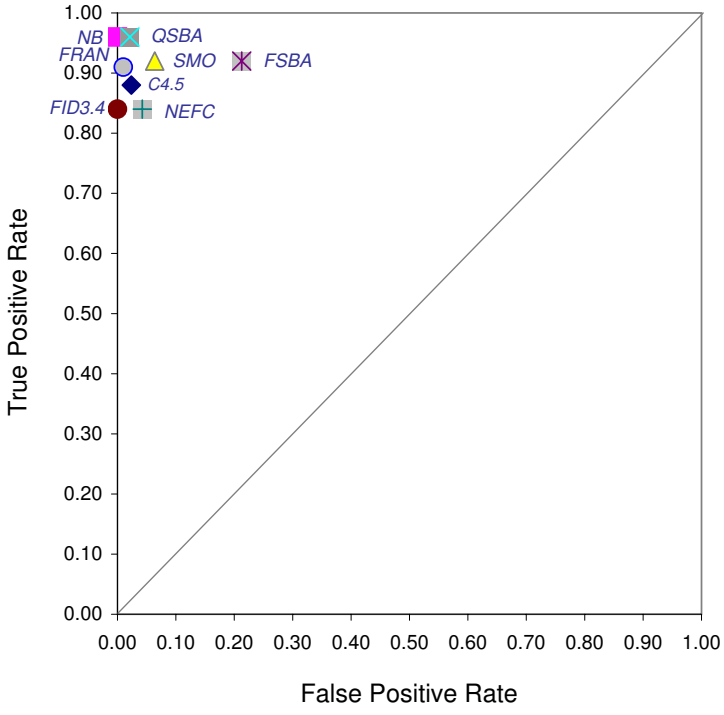
Table 8.1: Algorithm performance with respect to describing the minority class in a dataset

(a) Leukaemia

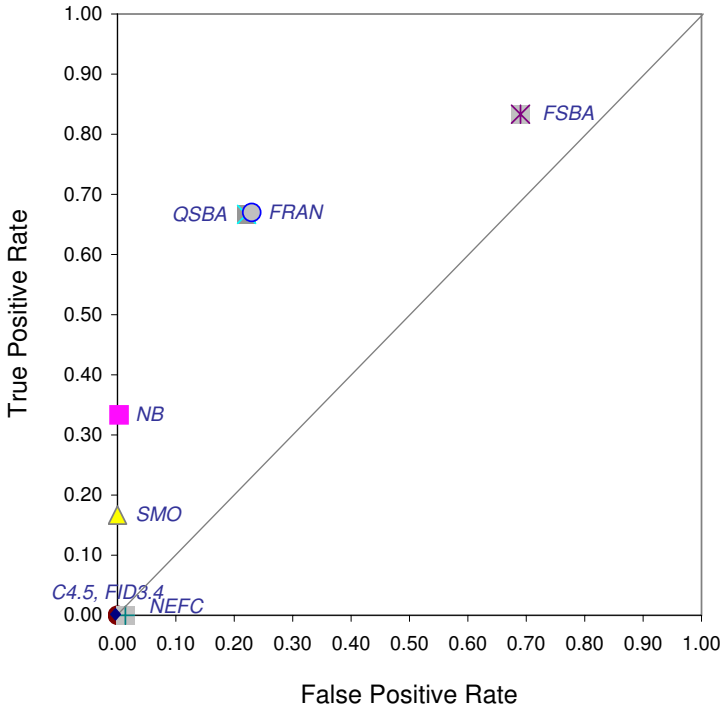
	TP Rate	FP Rate	Accuracy
C4.5	0.88	0.02	87.62%
NB	0.96	0.00	98.33%
SMO	0.92	0.06	92.32%
QSBA	0.96	0.02	96.90%
FSBA	0.92	0.21	82.80%
FID3.4	0.84	0.00	94.58%
NEFC	0.84	0.04	91.25%
FRAN	0.91	0.03	94.29%

(b) Water Treatment

	TP Rate	FP Rate	Accuracy
C4.5	0.00	0.00	98.43%
NB	0.33	0.00	98.68%
SMO	0.17	0.00	98.68%
QSBA	0.67	0.22	77.69%
FSBA	0.83	0.69	31.79%
FID3.4	0.00	0.00	98.43%
NEFC	0.00	0.01	97.08%
FRAN	0.67	0.23	77.22%



(a) Leukaemia



(b) Water Treatment

Figure 8.2: CAlgorithm performance with respect to describing the minority class

kaemia (AML). It accounts for 35% of the total instances, so that the majority class has almost twice as many instances. The second and third columns of Table 8.1(a) on page 185 show the *TP_Rate* and *FP_Rate* achieved by all algorithms on the Leukaemia dataset. The results obtained from running all the algorithms under different parameter settings were reviewed and those that showed the most favourable results with respect to models that best describe the positive class are used (these are the same results used in Fig. 8.1 in the previous subsection).

The highest accuracy results obtained by a model induced by *FSBA* is 93% ($\alpha = 1, \beta = 0.5 - 1$), but at a cost of being unable to accurately describe the minority class in the Leukaemia dataset. The models with best true positive and false positive rates are produced with $\alpha = 0.9, \beta = 0.5 - 1$, which results in an improved ability of the induced model to accurately predict positive class instances (high *TP_Rate*), but at the cost of misclassifying several of the majority class instances (high *FP_Rate*) – the overall classification accuracy is down to 83%.

The same information as in Table 8.1(a) on page 185 is presented in the form of a ROC graph, Fig. 8.2(a), where the *FP_Rate* is plotted on the x-axis and the *TP_Rate* on the y-axis – the higher the *TP_Rate*, and the lower the *FP_Rate*, the better is the performance of a classifier, so that classifiers positioned closer to the top-left hand corner of the ROC graph perform better. The diagonal line running through the origin indicates performance no better than random. Most algorithms achieve a reasonable combination of high *TP_Rate*, low *FP_Rate* and high overall classification accuracy on the Leukaemia dataset, with *NB* achieving the best results, followed closely by *QSBA* and then *FRANTIC*.

The highly imbalanced Water Treatment dataset – only 2% of total instances describe the minority 'Faulty' class – presents a different picture altogether. Most algorithms struggle to induce models that can accurately describe *both* classes (Table 8.1(b) and Fig. 8.2(b)). As for the Leukaemia dataset, all algorithm results for the Water Treatment dataset were reviewed and the best with respect to describing the minority class are used[1]. Table 8.1(b) shows that a model induced by *FSBA* achieves the best performance with respect to identifying instances that belong to the minority class, but it also misclassifies many instances of the majority class and therefore the overall classification accuracy is poor. A *FRANTIC* model obtains the second-highest *TP_Rate* (jointly with *QSBA*) and a more reasonable *FP_Rate* that results in overall classification accuracy of 77%.

C4.5, *NB*, *SMO*, *FID3.4* and *NEFCLASS* all achieve a classification accuracy of 97+%. The decision trees label *all* instances with the majority class, and since approximately 98% of instances belong to the majority class, then it is no surprise they obtain 98% accuracy. Pruned

[1]Better results with respect to overall accuracy are possible for both *FSBA* and *FRANTIC* – *FSBA* can achieve a high of 66% accuracy on this dataset ($\alpha = 0.8, \beta = 0.5 - 1$), but at a cost to describing the minority class, while *FRANTIC* can achieve a high of 93% accuracy, but at a lower *TP_Rate* than that given in Table 8.1(b).

C4.5 models, and unpruned or pruned *FID3.4* models for this dataset are actually trees with just one leaf, i.e. these models make no attempt to describe the minority class. *NEFCLASS* is unable to correctly classify any of the positive instances, and occasionally misclassifies a negative one. *NB* and *SMO* never misclassify negative instances, but are unable to correctly classify more than a few positive ones. Only *QSBA* and *FRANTIC*, with 78% and 77% overall classification accuracy respectively, manage to reasonably describe *both* classes in this highly imbalanced dataset.

Of note is the observation that *FRANTIC*'s performance on the Water Treatment dataset, and *QSBA*'s, is achieved without the use of common data pre-processing methods to even out the proportion of classes in an imbalanced dataset, such as undersampling the majority class (e.g. [122]), over-sampling the minority class (e.g. [124]), or a combination of both (e.g. [35]). Their success may be partly based on the fact that they are 'recognition-based' learners, as opposed to discriminant ones, where they make little (in *FRANTIC*'s case[2]), or no use (in *QSBA*'s[3]), of instances from *other* classes when constructing rules for a particular class (e.g. [108]).

The literature on induction from imbalanced datasets suggests that the difficulty in inducing accurate information need not be directly caused by the class imbalance. Reference [156] suggests that the degree of overlap between the classes may contribute to the problem, and [109] suggests that class imbalance becomes a problem only when the size of the minority class is very small with respect to its concept complexity. This is the case in the Water Treatment dataset, for instance – the minority class is made up of several other very small classes (with only one or a few instances in each), describing different problems that may befall the plant, and it is this that may have posed a problem for many of the learning algorithms used in this work.

Reference [109] states that most induction algorithms favour generality over specialisation and that though this is good for common cases (such as those describing good operation of a plant), it is not appropriate for cases that occur rarely (such as when a plant breaks down). Such a bias may even cause rare cases to be ignored completely. Another part of *FRANTIC*'s strength may therefore lie in its facility to control the balance between generalisation and specialisation, as implemented by the parameters `minInstPerRule` and `constructionThreshold`.

8.1.2 Ranking of Algorithms

Figure 8.3 on the following page provides an overview of the algorithms' performance over *all* datasets. For most datasets, the algorithm that induces the model achieving the highest classi-

[2] *FRANTIC* uses only target class instances when creating a rule, but *all* instances are used during rule evaluation.

[3] *FSBA*, like *QSBA*, creates a rule based only on instances of the target class, and in fact achieves the highest *TP_Rate* for the positive class. However, it also achieves a high *FP_Rate* resulting in a very low overall accuracy.

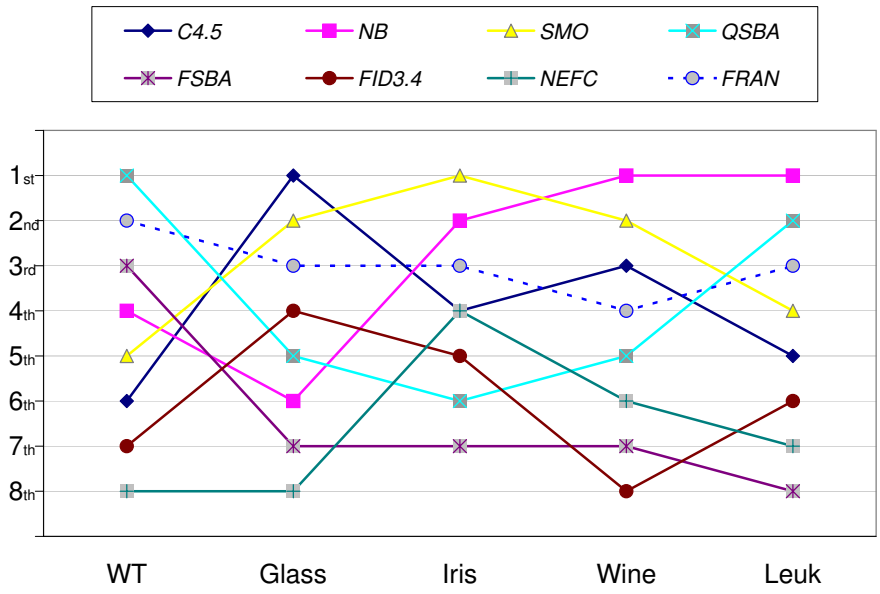


Figure 8.3: Comparison of algorithm performance over all datasets – classification accuracy

fication accuracy is ranked first, the one achieving second-highest accuracy is ranked second, and so on. If more than one algorithm achieves the same accuracy, they are ranked the same. For the Water Treatment and Leukaemia datasets, however, the Euclidean distance of an algorithm from the top left-hand corner of its ROC graph (Fig 8.2) is used for ranking. Ranking an algorithm by classification accuracy is practically meaningless for the highly-imbalanced Water Treatment dataset. For the Leukaemia dataset, most algorithms achieve a similar performance in describing the positive class and in overall accuracy, so that a ranking based on accuracy produces a similar line-up to that produced by using distance, with the same three algorithms as in Fig. 8.3 ranked the highest, and the same one ranked the lowest.

An overall indication of an algorithm's performance may be obtained by summing the rankings it achieves for all datasets – the lower the total score, the higher the rank. This results in 14 points for *NB* and *SMO* (i.e. overall rank of joint-first), 15 for *FRANTIC* (i.e. overall rank of second), 19 for *C4.5* and *QSBA*, and 28, 30 and 32 for *NEFCLASS*, *FID3.4*, and *FSBA* respectively. It should be noted that the accuracies achieved by these algorithms, and the *TP_Rate*s for the Leukaemia dataset, often differ by only small amounts, and that these rankings are an *ordinal* measure of overall performance and in no way suggest, for instance, that *NB* and *SMO* with a value of 14, are 'twice as good' as *NEFCLASS* with a value of 28.

It is clear from Fig. 8.3 that no one algorithm dominates all others over all datasets, or that one is consistently worse than the others. *NB* and *SMO* rank first or second for most datasets but then fifth or sixth for one, *C4.5*'s rankings range from first to sixth, *FID3.4*'s from fourth to eighth, and so on. *FRANTIC* does not achieve the highest rank on any dataset (ranking second,

third or fourth), yet it is also the only algorithm that does not perform particularly badly on *any* dataset. These are therefore highly encouraging results for *FRANTIC*, as they suggest a robustness to inherent characteristics of a dataset.

8.2 Model Comprehensibility

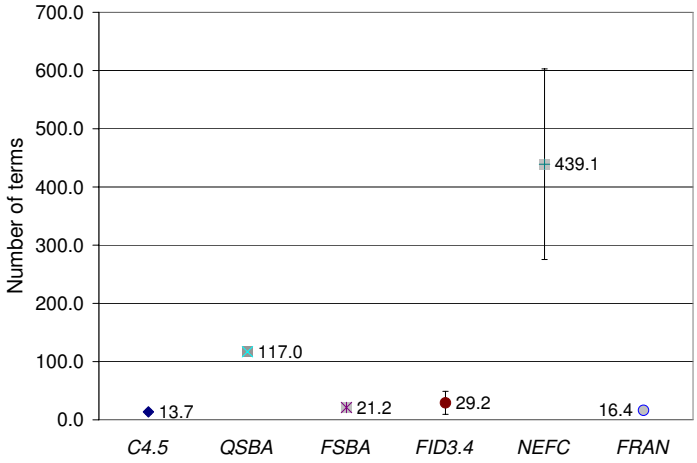
This section analyses the complexity of the models whose accuracy results are presented in the preceding section. The motivation is to compare the comprehensibility of the induced models, with smaller models being considered more human-comprehensible than larger models.

As described in Section B.2, *NB* determines a probability for each class that an instance may be assigned, and *SMO* forms a linear discriminant function to describe a boundary between each two classes. The models produced by these two algorithms are not open to an explicit and direct interpretation and validation by humans, especially when compared with rulebases and decision trees. They are therefore left out of the comparisons made between the models produced by the other algorithms.

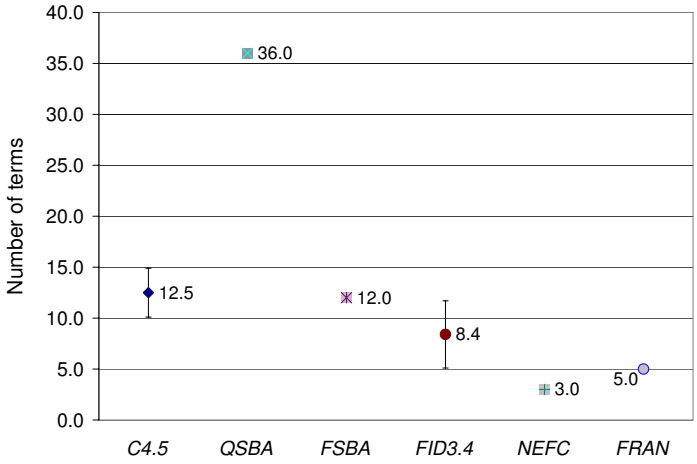
Figure 8.4 on the next page compares the performance of the remaining algorithms on all the datasets, with respect to the complexity of the models induced. As a global measure of model complexity, the number of terms in a decision tree or rulebase is used. The error bars on each graph denote the standard deviation obtained from the ten models produced by a ten-fold cross-validation run, and therefore gives an indication of the impact a different training set may have on model complexity. In order to facilitate a comparison, each subfigure illustrates the performance of all algorithms on one dataset; note that the y-axis scale is different from one dataset to another.

There is no standard deviation for the number of terms of a model produced by *QSBA*, since this algorithm creates rules that contain *all* terms in the rule antecedent (though each is weighted to indicate its relative importance in describing a particular class). All other algorithms may produce models in a ten-fold cross-validation run that differ in size. Their standard deviation is not always visible in the subfigures of Fig. 8.4, either because the resulting standard deviation is zero, or because the deviation of one of the algorithms is large and the scale of the graph does not allow the smaller deviations to be visible (e.g. Figs. 8.4(e) and 8.4(d)). However, the number of terms and standard deviations (and an estimate of the standard error of the mean) are also presented in Table E.3 on page 268.

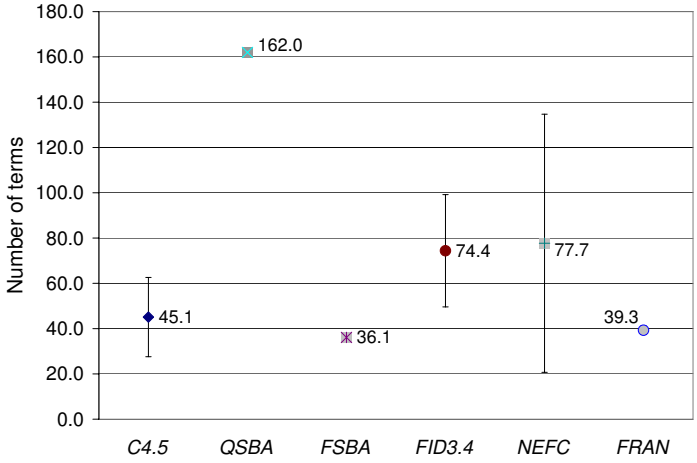
NEFCLASS (*NEFC* in graphs), and the decision tree inductors *C4.5* and *FID3.4* tend to produce models in a ten-fold cross-validation run with the greatest variation in the number of terms – in Fig. 8.4(e) *NEFCLASS* has the highest standard deviation by far, in Fig. 8.4(b) it is *FID3.4* and



(a) Wine

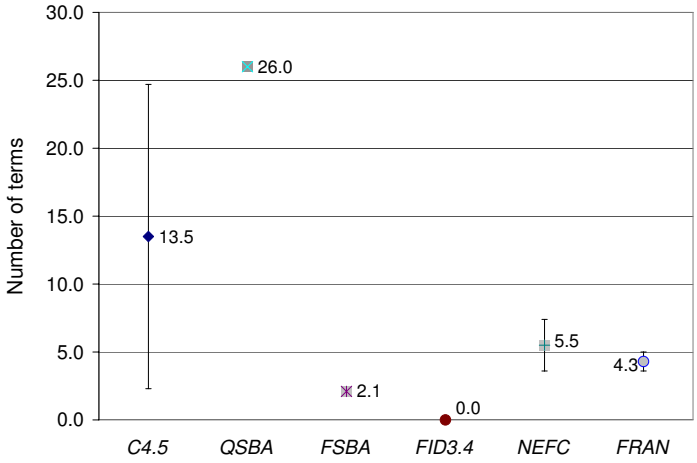


(b) Iris

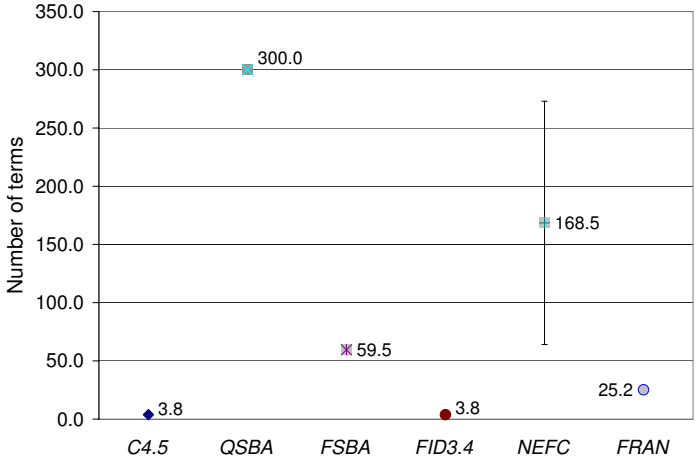


(c) Glass

Figure 8.4: Average number of terms in a rulebase or decision tree. Error bars denote standard deviation over the models produced by a 10-fold x-validation



(d) Water Treatment



(e) Leukaemia

Figure 8.4: Average number of terms in a rulebase or decision tree. Error bars denote standard deviation over the models produced by a 10-fold x-validation (cont.)

C4.5, in Fig. 8.4(c) it is all three algorithms, in Fig. 8.4(d) *C4.5* and *FID3.4*, and in Fig. 8.4(a) it is *NEFCLASS*.

In all cases – ignoring *QSBA* which is restricted by including all terms in its rules – it is also one of these three algorithms that produces the models with the greatest number of terms in a rulebase or decision tree. These algorithms have options that enable pruning, and all were run utilising this facility in order to obtain the most accurate and comprehensible models. It is acknowledged that an extensive search of the pruning parameter settings was not carried out, however, since several different options were allowed, the results obtained are considered indicative of an algorithm's general performance on these datasets.

If runs of an algorithm under different parameter settings (i.e. not just pruning parameters) produce models with the same highest accuracy but different complexity, the statistics for the smaller model are the ones used in Fig. 8.4 and Table E.3. The exception is the result for *C4.5* for the Water Treatment dataset (Fig. 8.4(d)). Running *C4.5* without pruning, with subtree replacement and raising pruning, and with reduced-error pruning all produced the same accuracy results. However, the runs that utilise pruning produce trees with just one leaf – all instances are automatically classified as one particular class of the dataset. Such a tree is not considered as providing particularly useful knowledge, since no information is available to describe the second class. The complexity statistics used for this algorithm on this dataset is therefore for unpruned models (achieving the same accuracy as the pruned models).

For this same dataset – Water Treatment (Fig. 8.4(d)) – *FID3.4* is also run without pruning and with different settings to a pruning parameter. However, *all* runs produce trees with just one leaf, classifying all instances as the majority class. When ranking an algorithm's performance on a particular dataset based on the average number of terms in a model, *FID3.4* is therefore ranked last on the Water Treatment dataset.

8.2.1 Ranking of Algorithms

Figure 8.5 on the next page provides an overview of the algorithms' performance over *all* datasets with respect to model complexity. For each dataset, the algorithm that induces the smallest model with regards to the average number of terms is ranked first, the one achieving second-smallest size is ranked second, and so on. If more than one algorithm produces models with the same average complexity, they are ranked the same.

Similar to summarising an algorithm's performance with regards to classification accuracy, an overall indication of its performance with regards to complexity is obtained by summing the rankings it achieves for all datasets. *FRANTIC* (with a sum of 24) attains an overall rank of first, followed by *FSBA* (23), *C4.5* (21), *FID3.4* (18), *NEFCLASS* (16), and *QSBA* (8). It should be

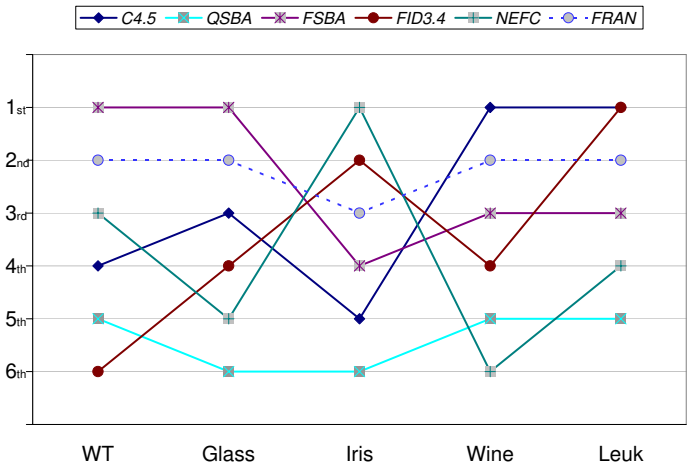


Figure 8.5: Comparison of algorithm performance over all datasets – average number of terms in a rulebase or decision tree

remembered that *SMO* and *NB* are excluded from this comparison since they would be ranked last on each dataset.

Section E.2 contains a supplementary analysis of model complexity by considering the average number of leaves/rules in a decision tree/rulebase, and the average number of nodes/terms per tree path/rule. These may be useful alternatives when comparing two algorithms that obtain similar measures with regards to the average number of terms in a rulebase, since it is arguable whether more but shorter rules, are more easily comprehensible than fewer but longer rules. Section E.3 shows examples of the models induced by all the algorithms.

8.3 Summary

This chapter has compared the performance of *FRANTIC* induced rulebases with that of models induced by several well-established algorithms over five real-world datasets with different issues. Algorithms included a support vector machine, crisp and fuzzy decision tree learners, and fuzzy rule induction algorithms. Dataset characteristics included an extreme imbalance in the distribution of instances between the classes, class inseparability, and an approximately equal proportion of attributes and instances in a dataset.

Where an algorithm had parameters to be set, several values were tried for the most essential features of the algorithm (following the recommendations and suggestion in the documentation with the implementations). It is acknowledged that this is not an extensive exploration and exploitation of the capabilities of some of these algorithms, and that in several cases the amount of tuning carried out for a particular parameter is far less than that for some *FRANTIC*

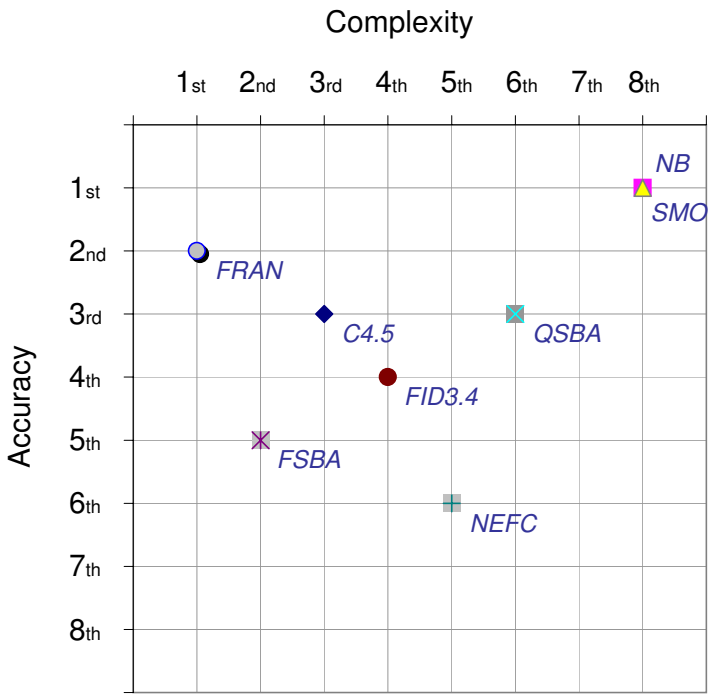


Figure 8.6: Ranking of algorithm performance over all datasets – classification accuracy and model comprehensibility

parameters. For several algorithms (including *FRANTIC*) there are parameters that have not been tuned at all, i.e. where the recommended default values are used. One algorithm (*QSBA*) has no parameters to set, and another (*FSBA*) has had both its parameters' settings varied extensively. These issues should be taken into consideration when interpreting the comparisons made.

Figure 8.6 illustrates the ranking that models induced by each algorithm achieve over all datasets – the axis on the left indicates performance with respect to classification accuracy, and the top axis indicates performance with respect to model comprehensibility. The rankings for accuracy are obtained from Section 8.1.1 and those for comprehensibility from Section 8.2.

Algorithms placed towards the top left-hand corner of the graph indicate better performance with regards to both criteria. *FRANTIC* presents the most even mix of accuracy and comprehensibility, followed by *C4.5*. *NB* and *SMO* produce models that are more accurate, but at a cost to comprehensibility. Almost all algorithms perform badly on at least one dataset (ranking 6th or lower), with respect to accuracy, and/or comprehensibility. *FRANTIC* achieves its high overall placement in Fig. 8.6 not because it performs spectacularly on a few datasets, but because it appears to be robust to the different problems posed by the different datasets and performs *consistently*, achieving a reasonable to very good performance on all datasets.

FRANTIC's robustness is believed to be a consequence of the rule construction mechanism with inbuilt parameters that allow a balance between specialisation and generalisation of rules (thereby preventing over-fitting to the training data), the way `minInstPerRule` has been implemented that allows it to be more specific to different classes, and the simultaneous evaluation of fuzzy rules that encourages a rulebase composed of complementary rather than competing rules. Furthermore, as discussed in Section 7.8, it is considered highly probably that SRL results may be improved by redesigning the rulebase selection mechanism. Other ways for potentially improving this approach is by allowing the system to construct more than one rule for a class, and this is discussed in the following chapter.

Chapter 9

Conclusions

This research has investigated the automated induction of human-comprehensible knowledge from empirical data, which may then be used to automate decision-making or to identify emerging trends in an organisation's business. The first use leads to a wide range of classification related applications, such as medical diagnosis and industrial plant control. The second may lead an organisation to develop new products and services, or enhance existing ones. Increasingly, induced knowledge is also used to help develop new domain theory in data-intensive areas such as the earth sciences and bioinformatics.

A common perception in the current literature on fuzzy rulebase induction is that the comprehensibility of the induced knowledge comes at a cost to its accuracy in describing the data from which it is extracted, and vice versa. This research proceeds from a basic requirement that induced knowledge is in a form in which humans can validate, and then focuses on increasing the accuracy of induced knowledge whilst maintaining its high comprehensibility.

The literature also suggests that a common approach for inducing fuzzy rulebases, that of iterative rule learning, may not be an optimal one with regards to accuracy. However, little or no evidence has been put forward to support or disprove this claim. This work therefore makes a direct and explicit comparison between what is perhaps the most common strategy for rulebase induction, and one that is introduced here specifically for the induction of fuzzy rulebases. The other major avenue explored for increasing rulebase accuracy is that of enriching the hypothesis language for describing the underlying patterns in datasets.

This chapter now concludes the research presented, with the next section highlighting key insights and contributions, and the following section providing a clear direction for future work.

9.1 Outcomes and Contributions

FRANTIC, the investigative tool developed during this work, is as much a teaching algorithm as a learning one. Analyses of its results not only show that the comprehensibility of induced rulebases can be maintained while improving their accuracy, but provide key insights and highlight fundamental concepts often neglected or ignored (e.g. that different classes in the same dataset may need to be treated differently), and common misconceptions (e.g. a richer hypothesis language necessarily leads to a more accurate rulebase).

The next subsection highlights *FRANTIC* features that enable it to induce rulebases that are both highly comprehensible and competitively accurate. The following subsection outlines the key insights obtained from analysing *FRANTIC* results, and which may also be used to direct future work.

9.1.1 System Features and Strengths

A major *FRANTIC* benefit is its consistent performance over datasets with different inherent characteristics, such as class separability and distribution. This consistency is due, at least in part, to the SRL strategy adopted. This strategy bases the selection of the 'best' rule for pheromone updates between ACO iterations on how rules describing different classes interact when classifying the training instances.

Despite the current suboptimal method used in SRL for selecting the final rulebase (best of final iteration vs. best of all iterations), very promising results are available. On all datasets tested, SRL-induced rulebases achieve an accuracy comparable or superior to that achieved by simplified and full IRL-induced rulebases, and with regards to rulebase comprehensibility, SRL rulebases are smaller than full IRL rulebases and therefore more comprehensible. An additional benefit of this approach that should not be overlooked is the increased robustness to changes in values of the `constructionThreshold` parameter, which, together with the `minInstPerRule` parameter, is key in providing the facility to control over-fitting to the training data. Furthermore, SRL uses one fewer parameter than simplified IRL and three fewer parameters than full IRL.

However, the constructionist nature of the rule discovery mechanism also affords several advantages, which render even the results of IRL-induced rulebases comparable with those of the established learning algorithms used in Chapter 8. The rule construction mechanism allows:

- Flexibility to create rules with increasing richness in the hypothesis language used;
- Easy insertion of domain knowledge, when available, into the rule construction process;

- Facility to control over-fitting to the training set through use of the rule construction parameters;
- Induction of accurate rules to describe small classes in highly imbalanced datasets.

It should be noted that the above points are also applicable and potentially very useful to the induction of crisp rules using ACO or any other constructionist technique for rule discovery. A heuristic is generally used in ACO-based crisp rule induction, as is a less sophisticated version of the `minInstPerRule` parameter (*Ant-Miner* sets one absolute value that is applied to all classes in the dataset). However, there is certainly scope for taking advantage of a richer hypothesis language, and for increasing the flexibility of the `minInstPerRule` equivalent, in order to render rule induction more relevant and specific to individual classes. The role of the construction parameters is also instrumental in controlling the generalisation and specialisation of induced rules.

Though not directly affecting the quality of the induced rulebases, the semi-automatic nature of key system parameters provide an additional benefit, by removing the need for user intervention before and/or during the induction of rulebases.

9.1.2 Key Rule and Rulebase Induction Findings

Experiment analyses provide important insights related to rule and rulebase induction (several of which are also relevant to crisp induction), and these are discussed in detail in Chapters 6–8. The ones highlighted here are considered to be particularly illuminating with regards to future work on fuzzy rulebase induction. Some are system or approach specific (i.e. may indicate ways to further enhance *FRANTIC* in particular), and others are more general (and may therefore be applied in the design of fuzzy rulebase systems in general). How these findings may be used to extend *FRANTIC* is discussed in Section 9.2.

The impact of the expressiveness of the hypothesis language can not be underestimated. Results indicate that the richer the language, the more likely *FRANTIC* is to find accurate rules to describe the different classes in a dataset, over a range of `constructionThreshold` values. More specifically, there is generally less variance between the accuracies of rulebases induced using different values for this parameter, if the language used is more rather than less expressive (Fig 6.6).

However, these same results also indicate that on some datasets, and for some `constructionThreshold` values, rulebases induced using simpler forms of the hypothesis language attain the highest accuracies. On one dataset (Wine), the highest accuracy of all the rulebases induced using different forms of the hypothesis language, and over all `constructionThreshold` values,

is achieved by a rulebase comprising simple propositional rules. Furthermore, detailed analysis of ten-fold cross-validation results reveal that different folds are described by some forms of the hypothesis language better than by others (Table 6.4), and that the better descriptors may be the simplest forms of the language. The more expressive forms of the language provide a consistency over `constructionThreshold` values *and* the different folds of a cross-validation run, but possibly at a cost of inducing a rulebase that may yet achieve a greater accuracy. These results clearly indicate that the impact of the hypothesis language is itself affected by the instances in a training dataset, and that it is not always necessary to use the richer (and often the more computationally expensive) form of a language.

Another interesting finding is the impact of the method for selecting the rule to be added to the final rulebase in IRL (at the end of each ACO run), and of selecting the final rulebase in SRL (after all ACOs have completed their final iteration). For forms of the hypothesis language other than rules with internal attribute-value disjunction, convergence generally occurs. However, there is no guarantee that pheromone levels converge to the rule or rulebase that achieves the highest fitness value (for IRL), or the highest classification accuracy on the training set (for SRL).

In IRL this is countered by the selection method that adds to the final rulebase the best rule of all iterations of an ACO run. In SRL, the best rulebase of the final iteration is selected, and this may lead to unnecessarily suboptimal rulebases, as there is no guarantee that this rulebase is the best one that has been created over all iterations. Artificial pheromones as a mechanism for learning how to construct better and better rules from iteration to iteration do work. However, it is important to appreciate the circumstances in which they may work, to what extent, and how limitations may be removed or alleviated. The different rule and rulebase selection methods for IRL and SRL highlight this.

Though not necessarily obvious from the results presented in Chapters 6–8, a (often under-utilised) premise on which *FRANTIC* is partially developed is that multi-class learning needs to be sensitive to the different requirements of the different classes. As stated in Section 4.3.1, early *FRANTIC* experiments indicated that setting the value of `minInstPerRule` to a *proportion* of class instances, rather than an absolute value applicable to all classes, leads to more accurate rulebases. The results discussed in Section 8.1.1 with respect to performance on the highly imbalanced Water Treatment dataset presents further evidence to support this conjecture that different classes may need to be treated differently.

For instance, the *FRANTIC* results in Table 8.1(b) are obtained with `minInstPerRule=70%`. When an exact number for the proportion of instances cannot be determined for a class, the figure is rounded. In the case of the Water Treatment dataset, this leads to 233 instances for the majority class (rounded from 233.1), which is very close to the stipulated 70% for the majority

class, and to 4 instances (rounded) for the minority class, which is actually 80% of the number of class instances. In effect, a *different* proportion of instances has been set for the different classes, and in this case has produced the best *FRANTIC* results. This suggests that *FRANTIC* may benefit from being rendered even more sensitive to the different requirements of different classes.

The following section details how these findings (and others discussed elsewhere in this document) may direct future work.

9.2 Future Work

Some possible future work has already been indicated. This includes: variations on the different elements of an ACO algorithm, such as the heuristic, pheromone updating strategy, and controlling the relative influence between the heuristic and pheromone levels (Chapter 3); the possibility of reducing or merging some or all of the fuzzy threshold parameters (Section 4.3.4); and several ways for reducing the computational expense of the rule construction and evaluation processes, including the use of a dynamic termination criterion for an ACO run and system parallelisation (Section 7.3).

This section describes work of a more strategic nature, aimed at testing new hypotheses arising from analysing experiment results in earlier chapters, and continuing a coherent and systematic investigation into the best approaches for fuzzy rule induction. Some of the work described is applicable to both of the rulebase induction strategies investigated in this research, and some is specific to one or the other.

Other suggested new work investigates the boundaries – the similarities and differences, the respective advantages and limitations – between the two strategies yet more closely. Though a premise of this research is that simultaneous evaluation and induction of a fuzzy rulebase leads to complementary rather than competing rules, results indicate that the IRL approach may be further improved (Section 7.1.2). This should not be ignored, as any knowledge one may gain about the advantages and/or limitations of one or the other strategy can only lead to an improved overall knowledge that may be applied more effectively to induction problems. The literature is replete with the results of small changes made to individual algorithms. Studies that aim to make comparisons between strategies and/or to consolidate knowledge and experience are less common. The research presented here is such an attempt to gain a higher-level understanding of what may or may not in general be an effective method for fuzzy rule induction, and the suggested future work for further exploration of the strategies is motivated by the same aim.

9.2.1 Strategy-Independent Work

Previous discussions have highlighted that the power of the hypothesis language is itself impacted by the training instances, and that there are occasions when a simpler form of the language may lead to a more accurate rulebase than if a richer form were used.

This therefore points the way to an important area of further *FRANTIC* development. Currently, the user decides which type of rules are constructed by ants within an ACO, and all ants construct the same type of rule. However, it is a conceptually simple modification to allow different subgroups of ants within an iteration to follow different construction graphs and adhere to different solution validation methods, and therefore construct rules using different (some less and some more rich) forms of a hypothesis language. One could allow the system to operate like this for all iterations, or allow it to determine after a number of iterations which types of ants are creating the better rules, and then instruct all other ants to create the same type of rule. This latter more flexible option may have the advantage of preventing unnecessary computation.

It should be noted that this modification will allow different classes to be described by different forms of the hypothesis language, if necessary. This change therefore supports the principle discussed in the previous section, that of enabling the system to be more sensitive to specific class needs.

Another modification in the same vein is that of setting *different* proportions of class instances for the `minInstPerRule` parameter, for *different* classes. As discussed in the previous section, this was inadvertently done in some experiments due to rounding of numbers, and the best results on the highly imbalanced Water Treatment dataset were obtained with effectively different values for the two classes. For full IRL mode, the `maxClassInstUncovered` parameter that determines how many class instances may be left uncovered before finding rules to describe the next class, should also be modified so that different proportions may be set for each class.

This modification effectively entails setting more than one parameter value, since each class may require a different value. However, it is possible that further system autonomy may be developed, for instance, by running ants within an iteration that follow different parameter settings. Again, after a number of iterations the system may be enabled to decide which values lead to the better rules.

9.2.2 Can Iterative Rule Learning be Improved?

As discussed in Section 7.1.2, over-fitting may be occurring during full IRL runs. This is due to the current way of re-setting the `minInstPerRule` parameter in between ACO runs for finding

rules to describe a specific class. This parameter is reset to ensure that the original *number* of instances that may remain uncovered is maintained. However, the current way of doing this also changes the original `minInstPerRule` *proportion* of class instances, and the new one may be considerably lower or greater than the original. Section 7.1.2 outlined how this may be prevented, since over-fitting to the training data may be occurring when a revised `minInstPerRule` proportion is considerably different from the original one.

Another suggested development arises from considering the work of Galea, Shen and Singh [69]. As described in Section 3.3.2, this work uses ACO as the rule discovery mechanism, and constructs simple propositional fuzzy IF-THEN rules using a class-independent IRL strategy. The authors compare the common method for adjusting the training set in between ACO runs (that of removing instances that are covered by the new best rule), with one that instead weights instances covered by the best rule, so that they are given less consideration in later ACOs. The conclusions are promising, with improved accuracy, improved robustness to a key paramater, though larger rulebases.

Other ways for adjusting the training set between iterations may also be considered. One simple modification to the removal of covered instances between iterations, may be to change the rule selection process slightly. Instead of evaluating each rule individually and separately on the training set, it may be possible to first combine each rule to be evaluated with existing rules in the final rulebase, and then use this new rulebase to (partially) classify the training set. This means that any new rule added to the final rulebase has been selected, based on how well it interacts with rules already present in it. It would be highly informative to explore the impact of these and similar strategies on *FRANTIC*'s class-dependent full IRL strategy.

9.2.3 Extending Simultaneous Rule Learning

The most obvious extension to the current SRL strategy is to enable it to induce more than one rule to describe each class in a dataset, if necessary. A conceptually simplistic approach would be to pre-determine the number of rules that may be required to describe each class, and then to initialise the appropriate number of ACO algorithms. 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 subclasses within a class would then indicate the number of ACOs to be initiated for that class.

A more sophisticated approach may be possible by again taking inspiration from social insects. Variable response threshold models have been developed by biologists to help explain specialisation in task performance by social insects that are initially identical. This model was first-developed for a multi-task environment in [181], which investigated division of labour in

insect societies. In a variable response threshold model every individual has a response threshold for every task, and an individual is more likely to perform a specific task when the level of the task-associated stimulus exceeds its corresponding threshold. The thresholds of individuals are allowed to vary in time, and a simple reinforcement process allows intially identical individuals to become specialists in a particular task – an individual's threshold for a particular task is decreased if it performs that task (i.e. making it more likely to perform that task again), and increased if it does not perform the task (i.e. making it less likely to perform the task).

In the context of simultaneously inducing the fuzzy rules for a rulebase, a population of initially identical artificial ants (or a number of initially identical colonies) could change, so that different subgroups of ants (or different colonies) work to produce descriptions for different classes simultaneously.

The reinforcement process in a variable response threshold model, however, allows the adjustment, in response to external or internal changes of the colony, of the number of workers performing specific tasks. Consequently, this model has served as a basis for dynamically regulating the allocation of tasks in artificial systems, including dynamic scheduling in the context of painting of trucks [25], and retrieval of mail by express couriers [18]. This task re-allocation feature may therefore be useful for providing overall control of the simultaneous induction of the entire fuzzy rulebase, where it might not be known in advance how many rules are required to describe each class.

With a little stretch of the imagination, the concept of super-colonies (from which SRL gains inspiration), may be extended further to deal with distributed and heterogeneous data. The main requirements are perceived to be the physical and technical resources that allow artificial colonies to communicate over a distributed geographical area, and to handle large amounts of data. Several ideas for enabling *FRANTIC* to scale up to large data volumes have already been discussed in Section 7.3.

A final piece of work is relevant to both IRL and SRL. This research uses ACO to construct individual fuzzy rules in two different strategies. However, IRL for fuzzy rule induction has often been implemented using GAs, GP or ES, for instance, as the rule discovery mechanism. Future studies could re-explore and re-compare the two strategies using different rule discovery mechanisms, whether they be constructionist, stochastic, deterministic, or otherwise. Discovering whether the two strategies exhibit strengths and limitations similar to those reported in this research, independent of the rule discovery mechanism, would help consolidate more of the existing knowledge and experience in the field of fuzzy rule induction.

Appendix A

Fuzzy Rules and Rule-Based Systems

This appendix provides a self-contained introduction to fuzzy rules and rule-based systems, in so far as it is necessary to understand the work presented in this thesis. For a more comprehensive exposition the reader is directed to [152] for fuzzy set theory and logic in general, and to [105] for classification and modeling with linguistic fuzzy rules in particular.

There are several different approaches for reasoning with imperfect or imprecise knowledge [149], including fuzzy rule-based systems (FRBSs) that are based on fuzzy set theory and fuzzy logic [195]. FRBSs capture and reason with imprecise or inexact knowledge (in fuzzy logic everything is a measure of degree [199]), 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. [90; 151]).

A simplified view of an FRBS is depicted in Fig. A.1 on the next 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 [22] – the interviews are generally long, inefficient and frustrating for both the domain experts and knowledge engineers, especially

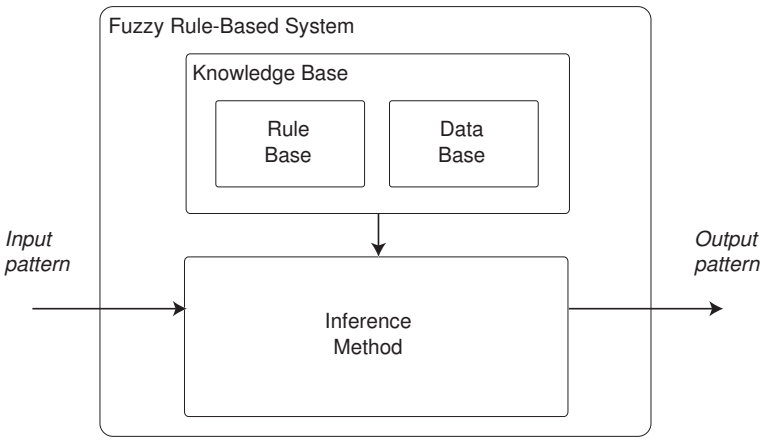


Figure A.1: A fuzzy rule-based system (FRBS)

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 sections 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.

A.1 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:

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

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 [151].

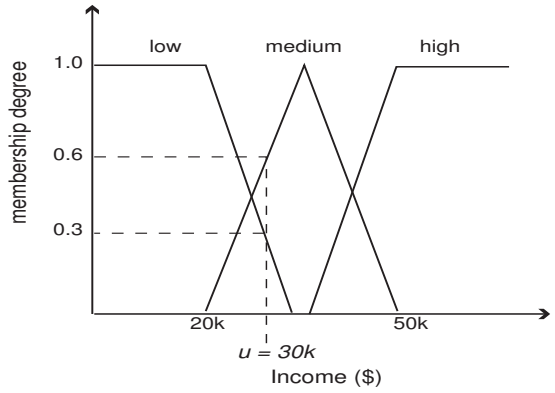


Figure A.2: A linguistic fuzzy variable

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:

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

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

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

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 [116] in more detail.

A.2 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 [196; 197; 198]. The domain values are called linguistic terms or labels, and each has associated with it a defining membership function.

Figure A.2 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 of 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. A.2 on the previous page 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 $\mu_{high\_income}(\$30k) = 0.0$.

There are different types of fuzzy IF-THEN rules, but the rules induced here are linguistic Mamdani-type rules [129], e.g.[1]:

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

Linguistic fuzzy rules, therefore, are a particularly explicit and human-comprehensible form for representing domain knowledge.

A.3 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 [41; 101] 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 is chosen because it is the one utilised by other algorithms against which *FRANTIC* is compared, and this issue is discussed in Section 5.4.2 on page 81.

The inference process used is a single winner based-method [101] 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. A.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.

[1] The underlying dataset which this rule partly describes is a fuzzified version of the artificial Saturday Morning Problem dataset introduced by Quinlan [157], where each instance is classified as to which sport – *Volleyball*, *Swimming*, or *Weightlifting* – should be played on a Saturday morning depending on the weather conditions

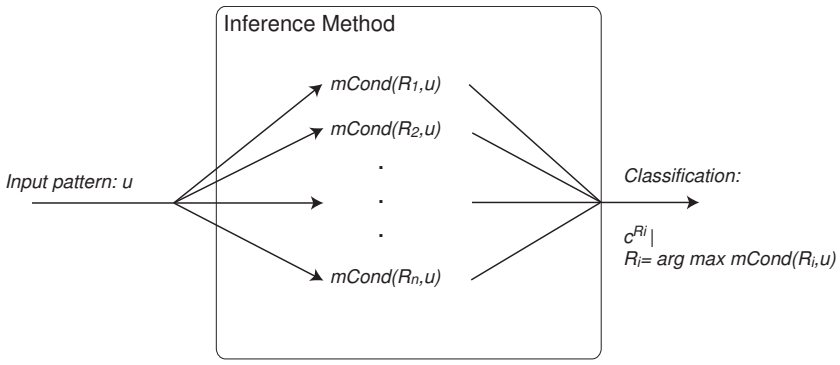


Figure A.3: Classification by an FRBS – single winner method

A.4 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 separated by a semi-colon, 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). Table A.1 on the next page presents the attributes and terms for this dataset. Terms that are present rule R_1 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

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

In the 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:

$$mAtt(R_1^k,u^k) = \begin{cases} 1 \ : \ all\ values\ of\ R_1^k\ are\ 0 \\ max_j(min(\mu_j(R_1^k),\mu_j(u^k))) \ : \ otherwise \end{cases} \tag{A.6}$$

Table A.1: The Saturday morning problem dataset

Attribute	Terms
OUTLOOK	*Sunny*, *Cloudy*, *Rain*
TEMPERATURE	*Hot*, *Cool*, *Mild*
HUMIDITY	*Humid*, *Normal*
WIND	*Windy*, *Not-windy*
ACTIVITY	*Volleyball*, *Swimming*, *Weightlifting*

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 is 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 are 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 are $mCond(R_2, u) = 0.2$ and $mCond(R_3, u) = 0.4$, then u is assigned the same class as that of R_1 – *Weightlifting*. Since the actual class of u is *Weightlifting*, then during training or testing this is counted as a correct classification.

When determining the accuracy of the induced rulebase, if more than one rule describing different classes obtains the highest degree of condition match with u, then this is considered a misclassification.

Appendix B

Experiment Setup Details

This Appendix provides supplementary information to Chapter 5 *Experiment Preliminaries*. It details the datasets and algorithms used (including parameter settings) in exploring FRANTIC's performance and capabilities, and the system parameter settings leadting to the results analysed in Chapters 6 to 8.

B.1 The Datasets

For ease of reference, a simplified form of Table 5.1 is reproduced below.

Table B.1: Summary of dataset properties

Dataset	Instances	Attributes	Classes	Distribution
Wine	178	13	3	33% : 40% : 27%
Iris	150	4	3	33.3% : 33.3% : 33.3%
Glass	214	9	6	33% : 35% : 8% : 6% : 4% : 14%
WT	377	5	2	98% : 2%
Leuk	72	50	2	65% : 35%

The Wine Recognition Dataset

The wine dataset records the measurements of 13 different chemicals in Italian wines from the same region that have been produced using three different cultivars, i.e. different varieties of grape under cultivation. The classification task is to identify the cultivar given the chemical analysis, with the distribution of instances between the cultivars being approximately equal.

This dataset presents a well-posed problem with separable classes and is generally used for initial testing of new learning algorithms. It has therefore been included as a 'control' set, as it is expected that all the algorithms should perform at least moderately well on it.

The Iris Plants Database

This dataset contains different measurements of two different parts of the iris plant – the width and length of the sepals and petals. The classification task is to determine the iris plant species – out of three possibilities – based on the plant measurements.

Originally introduced by Fisher in 1936 for taxonomy problems [56], it is still one the most extensively used datasets for classification testing. It has an exactly equal distribution of the instances between the three classes, but only one class is linearly separable from the other two.

The Glass Identification Database

The glass database contains measurements of the refractive index and the levels of various minerals and metals in different pieces of glass. The motivation is the identification of the particular type of glass left at a crime scene, which may be later used as evidence for prosecution.

The instances belong to one of six types of glass, with a distribution that is relatively uneven between some of the classes. This dataset is chosen mainly as it has a larger number of classes than the other datasets, and highlights an important area of future work for *FRANTIC*.

The Water Treatment Plant Database

The Water Treatment (WT) dataset contains the daily observations of 38 sensors monitoring the operation of an urban waste water treatment plant, with the objective being to predict faults in the process. Observations were taken over 527 days and are real-valued. There are 13 possible classifications for each daily set of observations, with many assigned to only a few records in the database, see Table B.2. These classifications have therefore been collapsed to two: *Normal* and *Faulty*, with the categories indicating acceptable performance in Table B.2 comprising the *Normal* class, and the remaining ones recording operational plant faults forming the *Faulty* class.

When faults are reported these are generally fixed very quickly and so the database contains a disproportionate number of records indicating correct operation of the plant, versus faulty operation. Records that have no assigned classification, and others with missing values have

Table B.2: Water Treatment Plant – number of observations for each classification

Class	Description	No. Cases
1	Normal situation	275
2	Secondary settler, type 1 problems	1
3	Secondary settler, type 2 problems	1
4	Secondary settler, type 3 problems	4
5	Good performance	116
6	Solids overload, type 1 problems	3
7	Secondary settler, type 4 problems	1
8	Storm, type 1 problems	1
9	Normal situation, low influent	65
10	Storm, type 2 problems	1
11	Normal situation	53
12	Storm, type 3 problems	1
13	Solids overload, type 2 problems	1

been removed. This leaves 377 records, with 98% classed as *Normal* and only 2% classed as *Faulty*.

This dataset has been used by other researchers in a feature-reduced state [169; 170], which resulted in more accurate classifiers. In this thesis a feature subset selection process [170] has also been applied to the dataset resulting in a final subset of 5 attributes from the original 38.

The Leukaemia Cancer Classification Dataset

The Leukaemia (Leuk) dataset contains 72 different samples of simultaneous gene expression levels obtained from DNA microarrays, with each sample being classed as one of two types of acute leukaemia – acute lymphoblastic leukaemia (ALL) or acute myeloid leukaemia (AML). This dataset is commonly used to test the application of machine learning techniques in the increasingly important bioinformatics area.

The first published use of this dataset in a classification context is [75], where the authors' first aim was to reduce the original number of 6817 genes to a more manageable one. The authors identified the 25 most highly correlated genes with each of the leukaemia types, see Fig. B.1[1]. Each column corresponds to a different sample in the training set, while each row corresponds to a different gene. The expression levels for each gene have been normalised across the sam-

[1]Reproduced from [75] with kind permission of T. R. Golub.

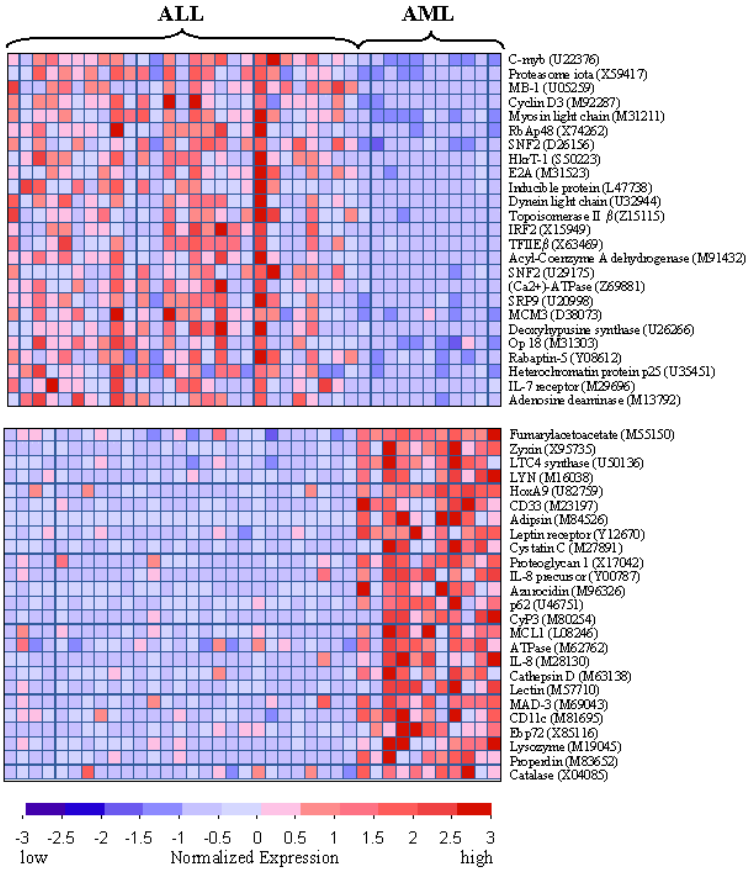


Figure B.1: Leukaemia dataset – the 50 genes most distinguishing the two classes ALL and AML. The top panel shows genes more highly expressed in ALL, while the bottom one shows those more highly expressed in AML.

ples so that the mean is 0 and the standard deviation 1. Expression levels greater than the mean are shaded in red, those below the mean in blue, while the scale indicates standard deviations above or below the mean. Using 38 samples for training and 34 for testing, these 50 genes were used to produce reasonable classifiers.

This thesis uses the same 50 genes identified by the original authors, but ten-fold cross-validation as the evaluation strategy. The proportion of number of attributes (50), to number of instances (approximately 65 in each of the 10 training folds), is considerably higher than in most data mining situations, but quite common in bioinformatics applications and believed to cause problems in the induction of accurate predictive models.

Though detailed results are not provided here, *FRANTIC* was also run with the same training and testing split of the dataset as used in [75], resulting in the number of attributes *exceeding* the number of training samples. Without extensive parameter tuning – i.e. initialising several different experiments with variations only in the `minInstPerRule` and `constructionThreshold` parameters – *FRANTIC* achieves a classification accuracy of 91.20%, which is greater than the original work's accuracy of 85.29%. All other results of experiments undertaken as part of the research for this thesis are obtained using ten-fold cross-validation.

B.2 The Other Algorithms

For ease of reference a simplified form of Table 5.2 is reproduced below. The following paragraphs describe how the settings for these other algorithms are determined. Any parameters for an algorithm that have not been explicitly mentioned are used at the default values provided by the implementation of that algorithm. Table E.2 on page 267 lists the settings that generate the results used to compare the performance of these algorithms with that of *FRANTIC*'s.

Table B.3: Summary of algorithms against which *FRANTIC* is compared

Algorithm	Fuzzy/Crisp	Model Type
C4.5	Crisp	Decision tree
NB	Crisp	Naïve bayes
SMO	Crisp	Support vector machine
QSBA	Fuzzy	Rule-based system
FSBA	Fuzzy	Rule-based system
NEFCLASS	Fuzzy	Rule based system
FID3.4	Fuzzy	Decision tree

C4.5

C4.5 [158] is a well-established learning algorithm that induces decision trees from training data. Decision tree induction follows a 'divide and conquer' approach whereby the training instances are partitioned into disjoint subsets and the algorithm is applied recursively to each subset. The splitting criterion for the training data is the attribute that shows the highest gain ratio – this is an information-based measure that takes into account the expected reduction in entropy of partitioning the training set samples based on the attribute, while attempting to minimise the number of partitions.

Each internal node of an induced tree therefore specifies a test on an attribute of the dataset. Each outgoing branch of the node corresponds to a possible result of the test, and leaf nodes represent the class to be assigned to an instance. To classify an instance a path is traced from the root node of the tree to a leaf – each internal node reached specifies the outgoing branch to be taken, based on the value of the relevant attribute in the instance. The instance is assigned the class label of the leaf node reached at the end of the path.

C4.5 is run several times under different conditions. In all cases, the minimum number of instances to match a leaf is the default value 2, and non-binary splits on attributes are allowed. The algorithm is run without pruning, with *C4.5* pruning, and with reduced-error pruning. Reduced-error pruning involves randomly splitting the training data and keeping one fold for evaluating pruning steps on the tree grown by the other folds – the value for the number of folds is the default of 3. Reduced-error pruning offers the advantage of better estimating pruning effects through use of a separate subset of the dataset, but fewer training instances are used to build the tree.

For each dataset, the best result with regards to classification accuracy from the different runs of *C4.5* is the one reported in Chapter 8. If more than one *C4.5* run produces the same highest accuracy, then the best results with regards to model complexity are used (i.e. the smaller the decision tree the better), with the exception of pruned trees that only have one node and therefore provide no explanatory power since *all* instances are labelled with the same class.

Naïve Bayes

Naïve Bayes (*NB*) [87] is a statistical modelling approach based on Bayes' rule of conditional probability. It does not produce an explicit model such as a rulebase or decision tree, but determines probabilities for each class label when considering an instance to be classified – the instance is labelled with the class that has the highest probability.

These probabilities are based on class frequencies from the dataset (i.e. the proportion of in-

stances in a dataset that has a particular class), and attribute-value frequencies for each class (the proportion of instances with a particular class that has a specific value for a specific attribute). The technique works on the assumption that each attribute contributes equally to the decision making and is independent of all other attributes. Though this assumption may not always hold in datasets, *NB* often works well in practice, and is included in this research as a simple representative of probabilistic modelling approaches.

NB generally handles numeric attributes by assuming that the values follow a normal distribution. The *Weka* implementation of *NB* offers an alternative which is to use a kernel density estimator when no assumptions may be made about attributes. *NB* was run on each dataset once using a normal distribution assumption, and once using a kernel density estimator. For each dataset, the best result with regards to classification accuracy from the two runs of *NB* is the one reported in Chapter 8.

SMO

A support vector machine (SVM) [183] is a combination of a linear model and an instance-based method as it selects a number of instances (called support vectors) that lie on the boundary between two classes, and forms a linear discriminant function (called a maximum margin hyperplane) that separates the vectors as much as possible. If a dataset has more than two classes, then pairwise classification is used – i.e. each two classes are considered in turn – to form such a classifier. For a dataset with k classes, $k(k-1)/2$ classifiers are therefore formed.

SVMs are known to achieve a high classification accuracy as they do not overfit to training data – the discriminant function is made up of a *subset* of the original training instances, and only an instance to be classified that should be a support vector itself, is likely to be misclassified, i.e. typical class instances will fall clearly on one or other side of the maximum margin hyperplane.

The particular *Weka* SVM implementation utilised in this research is *SMO* (Sequential Minimal Optimization) [155], a fast method for training SVMs. No normalisation of the data is carried out, but different kernels (similarity functions) are used to determine the maximum margin hyperplane – a polynomial kernel and radial basis function (RFB) one. For each dataset, the best result with regards to classification accuracy of the two runs using different kernel functions for *SVM*, is the one reported in Chapter 8.

QSBA

QSBA [161] stands for *Q*uantified *S*ubsethood-*B*ased *A*lgorithm, and is a deterministic algorithm with no parameters to set. It produces only one fuzzy IF-THEN rule to describe each

class in a dataset, and uses subsethood values (Section 4.1.2) to determine these rules – a rule contains each possible term (i.e. linguistic value) in its antecedent, and the associated subsethood value is used to determine a quantifier in the range [0,1] for a term.

The interpretation in the induced fuzzy rules of linguistic values of the same attribute is of internal disjunction, e.g. IF Q-E is $(0.31Low$ OR $1.0Normal$ OR $0.44High) \cdots$

FSBA

For ease of reference, *FSBA* is the name given in this thesis to the fuzzy subsethood-based algorithm developed in [36]. *FSBA* uses subsethood values to select conditions to formulate one fuzzy IF-THEN 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, for e.g. $IF\ Membership(NORMAL) < \beta\ THEN\ OUTCOME\ is\ FAULTY$.

Both α and β are tuned fairly extensively by varying their values in the range $[0.5, 1.0]$ using a step value of 0.05. For each dataset, the best result with regards to classification accuracy from the different runs of *FSBA* is the one reported in Chapter 8. If more than one *FSBA* run produces the same highest accuracy, then the best results with regards to model complexity are used (i.e. the smaller the rules and rulebases, the better).

NEFCLASS

NEFCLASS – *NE*uro *F*uzzy *CLASS*ification – is a neuro-fuzzy system for the induction of fuzzy linguistic rules and associated membership functions. The term "neuro-fuzzy" is defined by the authors as the "employment of heuristic learning strategies derived from the domain of neural network theory to support the development of a fuzzy system" [139]. Using the number and type of fuzzy sets specified for each attribute by the user, the system first creates an initial rulebase from the training data, and then optimises the fuzzy sets using a backpropagation-like algorithm.

The authors emphasise that *NEFCLASS* is an interactive and iterative tool (i.e. the user may have to go through several steps of refining parameter settings), requiring user input with regards to decisions on fuzzy sets, operations on the fuzzy sets, and controlling the size of rules

and rulebases [140]. In running *NEFCLASS* to produce the results in Chapter 8 a simplified process is followed – the system is allowed to determine the optimum number of initial rules (as this option is suggested to optimise accuracy), fuzzy sets are tuned, and then all available pruning steps are applied (each followed by further fuzzy set optimization).

The rule learning procedure parameter is set at the recommended 'best per class' (as it guarantees each class is covered by approximately the same number of rules, i.e. independently of the class distribution). Details regarding the system's inbuilt classification process for testing the induced rulebases are discussed in Section 5.4.2 on page 81, in the context of trying to replicate the testing environment between the different algorithms.

FID3.4

FID3.4 [107] produces fuzzy decision trees from training data, where each node in the tree is a fuzzy attribute (e.g. `Age`), and edges leading from a node relate to fuzzy terms such as `Age is old`, instead of a crisp condition such as `Age`$\geq$`80`. The procedure for using a fuzzy tree is therefore different from a crisp decision tree, as all instances to be classified will match all leaves, but to different degrees.

The implementation provides several parameters to control the tree-building process, a facility to prune a tree in order to reduce its size and improve its generalisation capability on test datasets, and different ways in which the inferencing process may be carried out on a test set. Details regarding the fuzzy sets of the datasets tested on *FID3.4* are discussed in Section 5.4.1 on page 79.

In building the decision trees that produce the results in Chapter 8, the parameters that define the fuzzy operators used in determining the best attribute on which to split are set to 'best' (i.e. at each node to be split, the fuzzy operator used at that node – out of several available, including *min* and *prod* – is the one that results in the highest value for the splitting criterion). *FID3.4* is then run four times with different levels of pruning – no pruning, and then gradually increasing the stringency of the pruning mechanism and setting the relevant parameter to values of 0, 0.5 and 1 (possible range for this parameter being [0,1]).

Details regarding customising the system's classification process for testing the induced fuzzy decision trees are discussed in Section 5.4.2 on page 81.

For each dataset, the best result with regards to classification accuracy from the different runs of *FID3.4* is the one reported in Chapter 8. If more than one *FID3.4* run produces the same highest accuracy, then the best results with regards to model complexity are used (i.e. the smaller the decision tree the better).

B.3 *FRANTIC* Parameter Settings

FRANTIC parameter settings are varied to investigate their impact on the rulebases produced by the system, and these results are discussed in Chapters 6 and 7. Table B.5 on page 222 shows *FRANTIC* settings used for the different rule learning strategy modes. Each subtable table has been reproduced in the main text where appropriate; they are shown together in this appendix for ease of comparison. Note that for the parameter `representation`, 'Negation' denotes rules constructed with negated terms in the antecedent, 'Simple' denotes simple propositional rules, 'Disjunction' rules with internal disjunction between attribute values, and 'Hedges' rules with both negated terms and linguistic hedges.

In these experiments the `constructionThreshold` parameter is always set to the values in the range $[0.50, 0.95]$, with a step value of 0.05, and generally the form of the hypothesis language used is rules that contain negated terms, the number of ants is 10, and the `minInstPerRule` setting for each dataset is as indicated in the table. When these parameter settings are varied in some experiments, the additional values used are indicated in the brackets.

The value for `minInstPerRule` is dataset dependent and how this is obtained for each dataset by running preliminary tests is explained in Section 5.5 – the `minInstPerRule` value chosen from these preliminary tests is the one that appears to lead to rulebases with greater generalisation power. The two additional values in brackets used in some of the experiments are the values closest to the initial `minInstPerRule` setting that appear to lead to rulebases with similar classification accuracy. Note, that though the value of `minInstPerRule` is generally dependent on the dataset, it appears to be less so when running in full IRL mode (at least for the datasets used here).

Table B.4 shows *FRANTIC* parameters settings that lead to the rulebases used for comparing with the models produced by the other algorithms utilised in Chapter 8. These *FRANTIC* rulebases achieve the highest accuracy from the simplified SRL rulebases produced and analysed in Chapter 7. A few full IRL rulebases induced achieve a higher classification accuracy for some datasets, but at the cost of an increase to rulebase size. Since the accuracy increase is not significant, the decision taken is to use the simpler rulebases in the comparison with other learning algorithms.

Table B.4: *FRANTIC-simpSRL* parameter settings used for comparing its performance with other learning algorithms

	Wine	Iris	Glass	WT	Leuk
`representation`	Hedges	Simple	Negation	Negation	Negation
`numIterations`	30	30	30	30	70
`numAnts`	10	10	6	10	10
`minInstPerRule`	60%	60%	70%	70%	80%
`constructionThreshold`	0.85	0.85	0.60	0.50	0.90

Table B.5: *FRANTIC* parameter settings for exploring the system's performance

(a) Simplified iterative rule learning mode

	Wine	Iris	Glass	WT	Leuk
representation	— Negation (Simple, Disjunction, Hedges) —				
constructionThreshold	— 0.50, 0.55, ..., 0.90, 0.95 —				
minInstPerRule	(50%) 60% (70%)	(40%,50%) 60%	50% (60%,70%)	70% (80%,90%)	(70%,80%) 90%
numAnts	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10
numIterations	50	100	50	100	70
fitnessThreshold	0.5	0.5	0.5	0.5	0.5

(b) Full iterative rule learning mode

	Wine	Iris	Glass	WT	Leuk
representation	— Negation (Simple, Disjunction, Hedges) —				
constructionThreshold	— 0.50, 0.55, ..., 0.90, 0.95 —				
minInstPerRule	(30%,40%) 50%	(40%) 50% (60%)	(30%,40%) 50%	50% (60%,70%)	40% (50%,60%)
numAnts	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10	(2,6) 10
numIterations	50	100	50	100	70
fitnessThreshold	0.5	0.5	0.5	0.5	0.5
removalThreshold	0.5	0.5	0.5	0.5	0.5
maxClassInstUncovered	10%	10%	10%	10%	10%

(c) Simplified simultaneous rule learning mode

	Wine	Iris	Glass	WT	Leuk
representation	— Negation (Simple, Disjunction, Hedges) —				
constructionThreshold	— 0.50, 0.55, ..., 0.90, 0.95 —				
minInstPerRule	(50%) 60% (70%)	(40%,50%) 60%	50% (60%,70%)	70% (80%,90%)	(70%,80%) 90%
numAnts	(2,6) 10	(2,6) 10	(2,6) 10	(2,4) 6	(2,6) 10
numIterations	30	30	30	30	70

Appendix C

Supplementary Results to Chapter 6

This appendix provides supplementary results to the analyses and discussions in Chapter 6. Some tables in this appendix present results that generate figures in the main text, and other tables and figures are for datasets not discussed in detail in the main text, due to similarities with other datasets that have been analysed there. All the results in this appendix pertain to *FRANTIC* rulebases induced using a simplified IRL approach.

Tables generally present the main statistic – whether it be average accuracy or average terms in a rulebase/terms per rule/rules per rulebase – and associated standard error of the mean (SEM) and standard deviation (STD). Since 10 ten-fold cross-validations are carried out, the STD is the average of the 10 standard deviations obtained from each of the 10 cross-validations (this gives an indication of *FRANTIC*'s robustness to the different training sets used in one ten-fold cross-validation), and the SEM is the standard deviation of the 10 average accuracies from each of the 10 cross-validations (this gives an estimate of the reliability of the main statistic presented in a table).

C.1 The Heuristic

Table C.1 on the next page presents the average convergence iteration number for each dataset, for rulebases induced by *FRANTIC* running in simplified IRL form, with and without the use of heuristic information during rule construction. The standard error of the mean (SEM) is also provided. A graphical representation of the data in this table is presented in Fig. 6.1 on page 90. The concept of a convergence iteration number is described on page 86.

Figures C.1–C.4 depict the individual class convergence values for the Iris, Glass, Water Treatment and Leukaemia datasets respectively. The graphs depict similar properties to that of the Wine dataset which is discussed in Section 6.1 on page 85.

Table C.1: Average convergence (AVE) and standard error of the mean (SEM) for *FRANTIC-simpIRL* rulebases induced with (+Heuristic) and without (−Heuristic) use of heuristic information during rule construction

(a) Wine

construction Threshold	+Heuristic AVE	+Heuristic SEM	-Heuristic AVE	-Heuristic SEM
0.50	20.47	0.96	20.72	1.35
0.55	18.86	0.84	19.25	0.88
0.60	16.79	0.60	17.29	0.65
0.65	15.80	0.81	16.15	0.52
0.70	15.78	0.66	16.09	0.59
0.75	14.75	0.57	15.38	0.69
0.80	13.92	0.66	14.44	0.75
0.85	13.51	0.81	13.91	0.65
0.90	12.57	0.45	13.04	0.62
0.95	12.17	0.45	12.71	0.49

(b) Iris

construction Threshold	+Heuristic AVE	+Heuristic SEM	-Heuristic AVE	-Heuristic SEM
0.50	11.46	0.52	11.80	0.39
0.55	10.71	0.47	11.98	0.38
0.60	10.89	0.49	11.12	0.54
0.65	10.88	0.53	11.07	0.50
0.70	10.08	0.54	10.82	0.20
0.75	10.31	0.56	10.76	0.50
0.80	9.89	0.57	9.91	0.32
0.85	7.66	0.37	7.46	0.20
0.90	6.57	0.40	6.82	0.40
0.95	6.51	0.31	6.63	0.39

(c) Glass

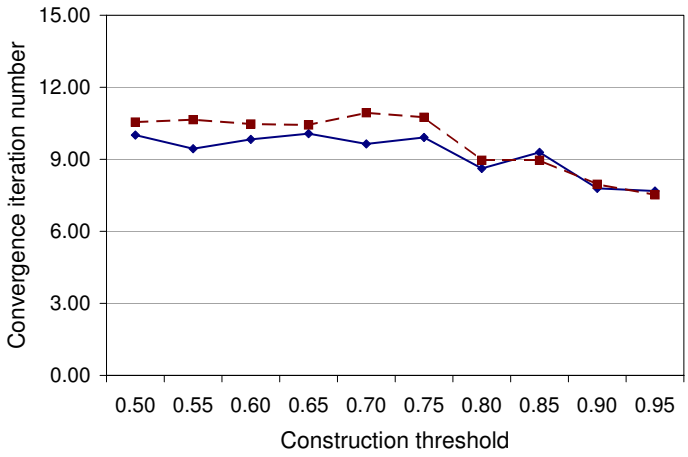
construction Threshold	+Heuristic AVE	+Heuristic SEM	-Heuristic AVE	-Heuristic SEM
0.50	21.65	0.83	21.70	0.64
0.55	20.31	0.56	20.81	0.47
0.60	17.71	0.63	17.66	0.79
0.65	16.11	0.45	16.41	0.43
0.70	15.58	0.46	15.94	0.41
0.75	15.15	0.39	15.10	0.30
0.80	14.42	0.42	14.76	0.52
0.85	14.70	0.42	15.03	0.38
0.90	15.10	0.53	15.22	0.31
0.95	14.54	0.39	14.93	0.36

(d) Water Treatment

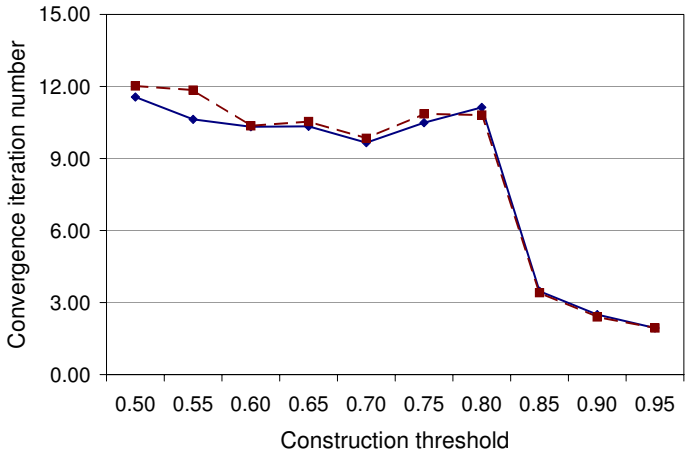
construction Threshold	+Heuristic AVE	+Heuristic SEM	-Heuristic AVE	-Heuristic SEM
0.50	13.41	0.73	13.22	0.92
0.55	12.44	0.76	12.70	0.57
0.60	13.10	0.41	13.54	0.39
0.65	13.11	0.69	13.59	0.93
0.70	10.78	0.77	10.71	0.52
0.75	11.35	1.00	11.28	0.64
0.80	7.11	0.53	7.27	0.52
0.85	4.79	0.38	4.99	0.50
0.90	4.72	0.39	4.78	0.36
0.95	4.64	0.41	4.90	0.46

(e) Leukaemia

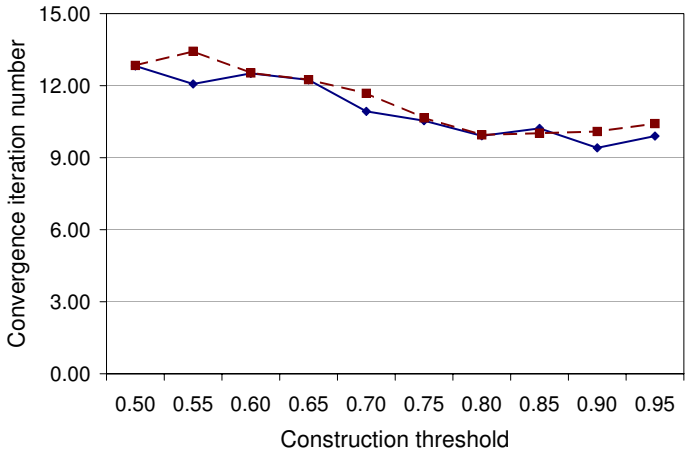
construction Threshold	+Heuristic AVE	+Heuristic SEM	-Heuristic AVE	-Heuristic SEM
0.50	4.98	0.35	4.99	0.28
0.55	12.60	0.50	13.02	0.90
0.60	11.66	0.62	12.24	0.71
0.65	12.25	0.78	12.00	0.39
0.70	12.46	0.57	12.04	0.83
0.75	12.14	0.90	12.45	0.45
0.80	11.34	0.58	11.20	0.61
0.85	10.73	0.52	11.32	0.89
0.90	10.49	0.60	10.42	0.53



(a) *Iris-setosa* class

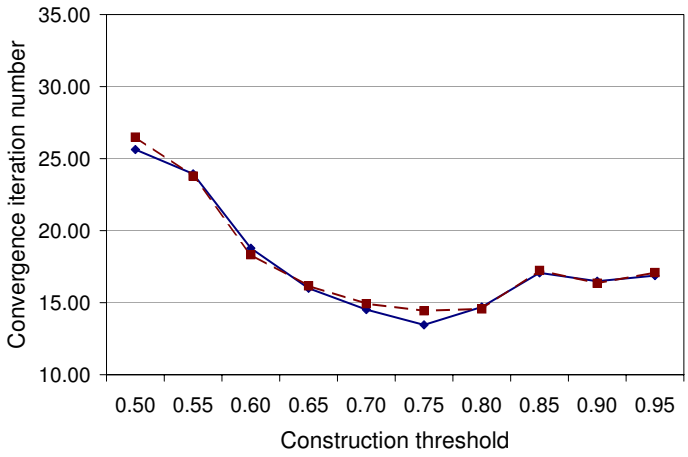


(b) *Iris-versicolor* class

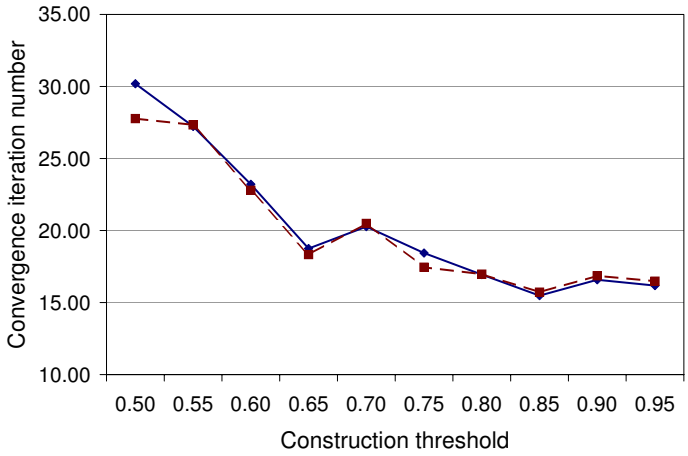


(c) *Iris-verginica* class

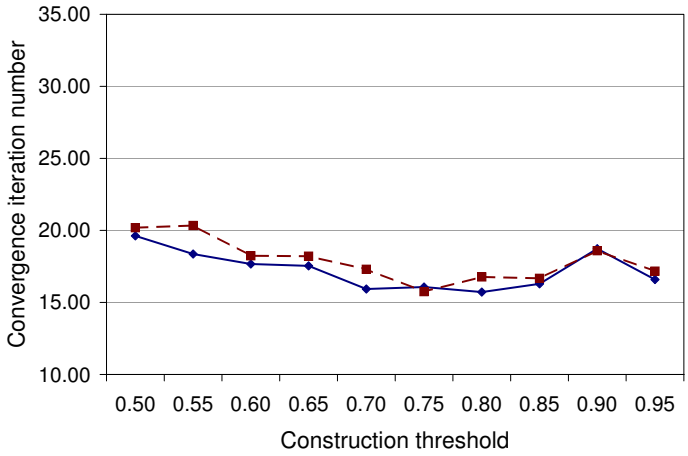
Figure C.1: Iris dataset – individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information



(a) *Building windows* class

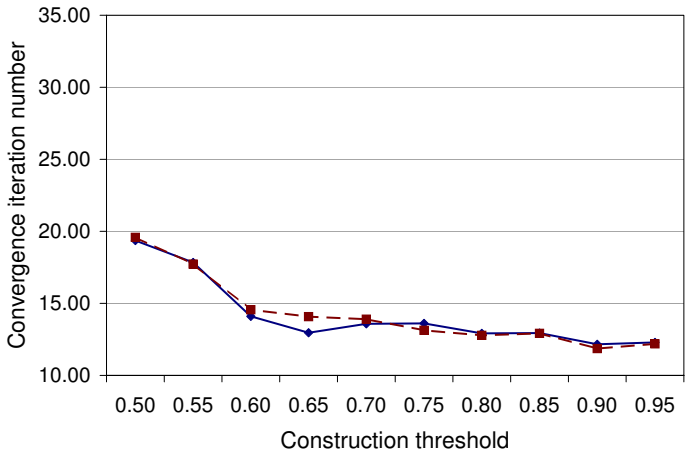


(b) *Building windows nonfloat* class

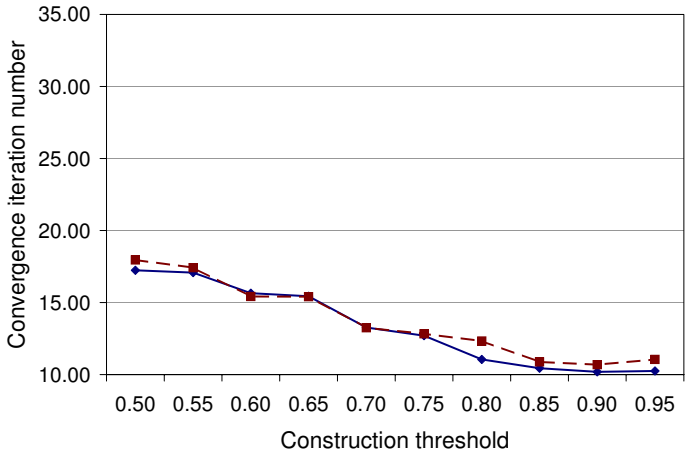


(c) *Vehicle windows float* class

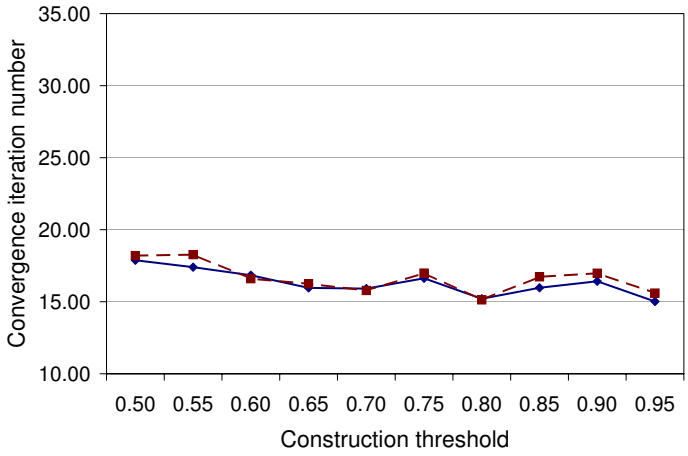
Figure C.2: Glass dataset – individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information



(d) *Containers* class

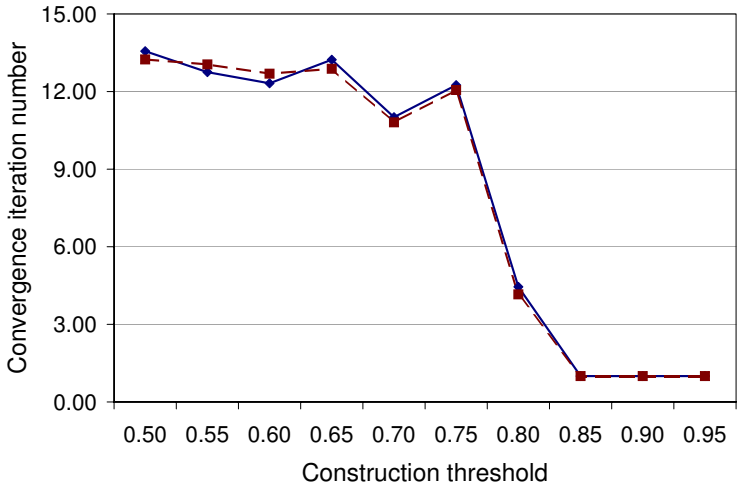


(e) *Tableware* class

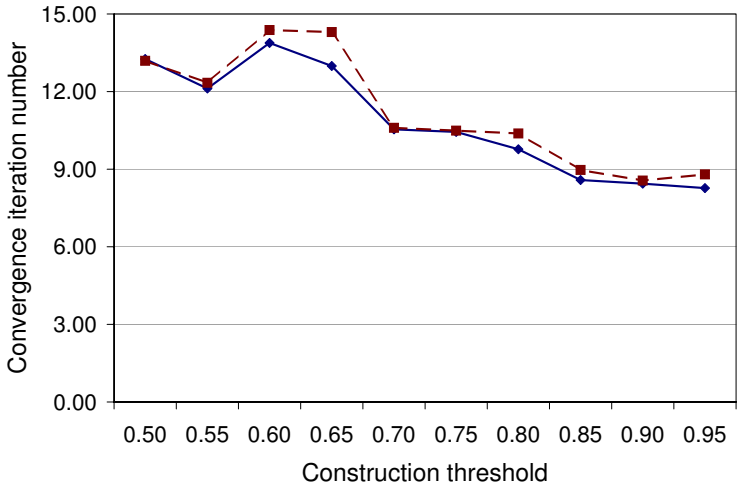


(f) *Headlamps* class

Figure C.2: Glass dataset – individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information (cont.)

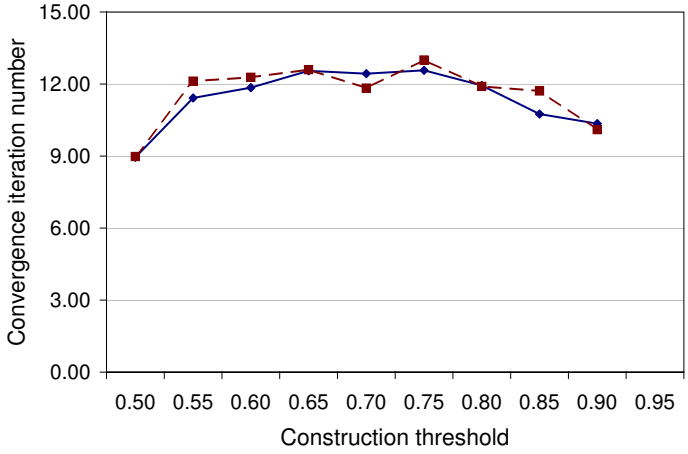


(a) *Normal* class

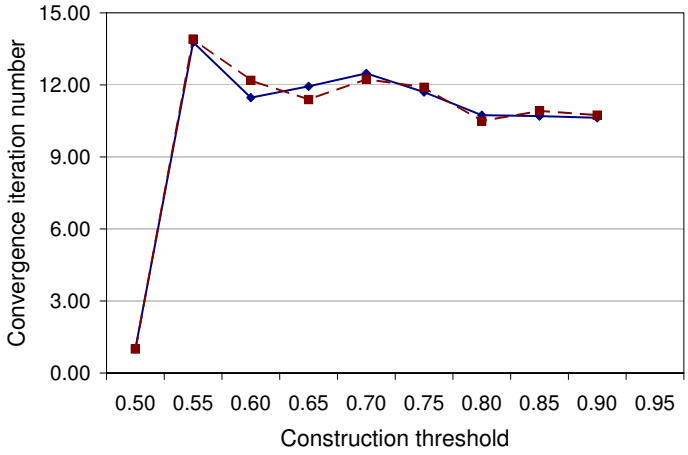


(b) *Faulty* class

Figure C.3: Water Treatment dataset – comparison of individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information



(a) *Acute lymphoblastic leukaemia (ALL)* class



(b) *Acute myeloid leukaemia (AML)* class

Figure C.4: Leukaemia dataset – individual class convergence for rules constructed with (solid line) and without (dashed line) use of heuristic information

C.2 The Pheromone Levels

Table C.2 provides a comparison of the classification accuracy and associated standard deviation for ten-fold cross-validation runs, for *FRANTIC* rulebases induced with the use of heuristic information and pheromone trails (+Heur +Pher), without use of heuristic and pheromones (−Heur +Pher), with use of heuristic and without use of pheromones (+Heur −Pher), and without use of either the heuristic or pheromones (−Heur −Pher) during rule construction. Graphs generated from these statistics are discussed in the main text in Sections 6.1 and 6.2.

Note that columns 2–5 for each dataset in Table C.2 present the same accuracy and standard deviation results as in Table 6.2 on page 87, but are reproduced here for ease of comparison.

Table C.2: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced with and without use of heuristic information and pheromones trails

(a) Wine

	+Heuristic			-Heuristic			-Pheromone			-Phero -Heur		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
Threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	82.95	1.80	8.77	83.52	1.88	9.47	83.26	2.13	8.68	83.45	2.05	8.95
0.55	86.86	1.87	7.51	85.73	2.68	7.43	86.78	2.95	7.44	86.89	1.32	7.57
0.60	86.05	1.45	7.61	87.11	1.35	7.82	86.21	0.70	7.55	86.65	1.59	7.58
0.65	85.35	2.35	7.51	85.09	1.31	7.06	84.97	1.12	8.33	86.07	1.65	7.22
0.70	88.09	1.59	9.28	88.66	1.47	8.15	87.77	0.39	10.11	87.95	0.73	9.55
0.75	90.47	1.32	5.31	90.70	1.35	5.20	91.66	0.59	3.45	91.21	0.48	3.84
0.80	93.01	0.77	5.44	92.61	0.57	5.10	94.23	0.23	4.70	93.92	0.40	4.91
0.85	92.20	1.04	5.13	91.88	1.52	5.23	93.39	0.36	4.92	93.39	0.25	5.02
0.90	85.67	0.80	8.38	86.04	0.65	8.47	86.33	0.29	8.12	86.33	0.40	8.21
0.95	84.74	0.58	7.32	85.29	0.58	7.54	84.69	0.80	7.41	84.65	0.71	7.23

(b) Iris

	+Heuristic			-Heuristic			-Pheromone			-Phero -Heur		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
Threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	86.73	0.73	11.41	86.80	0.69	11.10	86.47	0.45	11.45	86.60	0.21	11.26
0.55	86.33	0.47	10.73	86.67	0.77	10.80	85.93	0.73	10.62	86.20	0.45	10.21
0.60	85.27	0.66	11.08	85.20	1.08	11.30	85.27	0.58	10.80	84.73	0.91	11.49
0.65	88.80	0.76	9.71	88.60	0.66	9.90	88.40	0.56	9.35	88.27	0.64	9.57
0.70	93.33	0.00	7.03	93.33	0.00	7.00	93.33	0.00	7.03	93.33	0.00	7.03
0.75	75.33	0.00	7.06	75.33	0.00	7.10	75.33	0.00	7.06	75.33	0.00	7.06
0.80	76.67	0.00	6.48	76.67	0.00	6.50	76.67	0.00	6.48	76.67	0.00	6.48
0.85	76.67	0.00	6.48	76.67	0.00	6.50	76.67	0.00	6.48	76.67	0.00	6.48
0.90	75.33	0.00	6.32	75.33	0.00	6.30	75.33	0.00	6.32	75.33	0.00	6.32
0.95	63.33	0.00	11.86	63.33	0.00	11.90	63.33	0.00	11.86	63.33	0.00	11.86

(c) Glass

	+Heuristic			-Heuristic			-Pheromone			-Phero -Heur		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
Threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	53.74	1.93	8.98	54.55	1.58	8.21	54.21	1.67	10.22	53.76	2.03	8.99
0.55	53.09	2.74	11.90	53.20	1.67	11.47	52.25	1.20	11.82	52.00	1.02	12.20
0.60	55.63	1.64	9.66	55.67	0.99	9.43	56.10	0.71	9.75	56.03	0.77	10.20
0.65	49.77	0.54	11.74	50.04	0.77	12.46	49.56	0.57	11.99	49.81	0.44	12.01
0.70	45.82	1.03	15.35	45.67	1.44	15.13	45.01	0.14	14.87	45.01	0.14	14.87
0.75	42.00	0.62	16.89	41.68	0.58	17.49	42.02	0.54	17.45	42.10	0.63	17.35
0.80	47.54	0.65	16.09	47.52	0.23	16.08	47.87	0.25	15.60	47.83	0.36	15.91
0.85	44.61	0.38	15.31	44.36	0.56	15.19	44.35	0.31	15.19	44.43	0.21	15.16
0.90	42.91	0.95	11.72	43.04	0.82	11.88	43.39	0.21	10.60	43.39	0.21	10.60
0.95	45.22	0.57	14.21	45.27	0.47	14.36	45.65	0.00	14.38	45.65	0.00	14.38

Table C.2: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced with and without use of heuristic information and pheromones trails (cont.)

(d) Water Treatment

	+Heuristic			-Heuristic			-Pheromone			-Phero -Heur		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
Threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	72.24	0.00	8.00	72.24	0.00	8.00	72.24	0.00	8.00	72.24	0.00	8.00
0.55	74.35	0.00	8.39	74.35	0.00	8.39	74.35	0.00	8.39	74.35	0.00	8.39
0.60	71.89	0.00	6.90	71.89	0.00	6.90	71.89	0.00	6.90	71.89	0.00	6.90
0.65	71.89	0.00	6.90	71.89	0.00	6.90	71.89	0.00	6.90	71.89	0.00	6.90
0.70	70.84	0.00	6.50	70.84	0.00	6.50	70.84	0.00	6.50	70.84	0.00	6.50
0.75	72.16	0.00	7.38	72.16	0.00	7.38	72.16	0.00	7.38	72.16	0.00	7.38
0.80	74.83	0.00	7.72	74.83	0.00	7.72	74.83	0.00	7.72	74.83	0.00	7.72
0.85	79.02	0.00	5.92	79.02	0.00	5.92	79.02	0.00	5.92	79.02	0.00	5.92
0.90	82.78	0.00	6.11	82.78	0.00	6.11	82.78	0.00	6.11	82.78	0.00	6.11
0.95	82.78	0.00	6.11	82.78	0.00	6.11	82.78	0.00	6.11	82.78	0.00	6.11

(e) Leukaemia

	+Heuristic			-Heuristic			-Pheromone			-Phero -Heur		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
Threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	89.29	2.03	15.99	89.39	2.31	16.00	90.03	1.41	16.02	88.89	2.69	15.88
0.55	90.38	1.13	16.24	90.52	0.73	16.15	90.21	0.96	16.24	90.21	0.96	16.24
0.60	92.68	1.04	10.05	92.93	1.23	8.99	92.56	0.00	7.93	92.56	0.00	7.93
0.65	92.58	2.16	11.01	92.24	1.99	11.86	90.98	1.87	14.21	92.43	1.75	10.34
0.70	92.18	1.87	10.30	91.27	2.59	11.98	92.27	1.97	10.72	91.43	2.03	12.72
0.75	93.57	0.79	11.54	93.40	1.23	12.09	93.90	0.70	10.70	93.74	0.53	11.07
0.80	93.78	0.99	9.60	93.65	1.29	9.56	94.24	1.20	8.83	94.03	0.82	8.53
0.85	95.63	1.55	7.07	95.00	1.60	7.68	95.07	0.75	7.42	95.60	1.09	7.05
0.90	93.35	1.21	9.95	92.76	1.53	9.94	93.33	1.40	9.61	93.38	1.36	8.92
0.95	93.11	1.09	8.52	93.13	0.74	7.90	93.77	1.22	7.69	93.11	1.35	8.80

C.3 The Rule Construction Parameters

Table C.3 provides the average number of terms, and associated standard error of the mean (SEM) and standard deviation (STD), of *FRANTIC* rulebases induced in simplified IRL mode. For all datasets, the average number of terms decreases as the values for the `constructionThreshold` and `minInstPerRule` parameters increase. The graphs generated from these statistics are discussed in detail in the main text in Section 6.3 on page 98.

Table C.4 provides the accuracy, and associated standard error of the mean (SEM) and standard deviation (STD), for the rulebases whose complexity statistics are reported in Table C.3. These accuracy statistics are discussed in the main text in Chapter 7, Sections 7.1.1 on page 117 and 7.2.1 on page 141, but are presented in this supplement in order to keep all simplified IRL induced rulebase statistics and figures in the one appendix.

Table C.3: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	30.08	0.49	1.59	26.82	0.44	1.54	23.66	0.20	1.30
0.55	24.88	0.64	2.23	23.41	0.53	1.96	21.40	0.35	1.71
0.60	25.43	0.25	1.83	23.07	0.30	1.40	20.56	0.33	1.97
0.65	25.30	0.41	1.05	23.18	0.27	1.36	20.51	0.15	1.36
0.70	23.66	0.27	1.43	21.06	0.33	1.36	18.02	0.25	1.33
0.75	21.68	0.36	1.19	18.84	0.30	1.30	15.74	0.36	1.41
0.80	19.73	0.52	1.85	16.83	0.29	2.16	15.84	0.38	1.04
0.85	18.73	0.31	1.08	16.38	0.19	1.47	14.92	0.36	1.09
0.90	18.66	0.28	1.43	17.27	0.28	1.43	15.26	0.22	1.20
0.95	19.05	0.28	1.79	16.54	0.29	1.34	14.26	0.13	0.82

(b) Iris

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	12.00	0.00	0.00	12.00	0.00	0.00	11.82	0.04	0.40
0.55	12.00	0.00	0.00	12.00	0.00	0.00	11.67	0.08	0.62
0.60	12.00	0.00	0.00	11.35	0.05	0.60	11.53	0.05	0.69
0.65	11.91	0.03	0.28	11.25	0.05	0.56	10.76	0.07	0.44
0.70	11.80	0.00	0.42	11.22	0.04	0.60	10.00	0.00	0.00
0.75	10.75	0.05	0.45	10.43	0.05	0.65	9.90	0.00	0.32
0.80	10.51	0.07	0.73	9.50	0.00	0.53	9.90	0.00	0.32
0.85	9.92	0.09	0.27	9.50	0.00	0.53	9.90	0.00	0.32
0.90	10.00	0.00	0.47	10.00	0.00	0.00	10.00	0.00	0.00
0.95	10.00	0.00	0.00	9.90	0.00	0.32	9.60	0.00	0.70

(c) Glass

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	46.82	0.18	0.87	44.34	0.26	1.24	42.69	0.22	1.56
0.55	44.63	0.19	1.22	44.73	0.45	2.64	39.92	0.31	1.84
0.60	39.45	0.53	2.21	41.38	0.19	1.53	37.59	0.37	2.03
0.65	42.12	0.28	1.93	37.51	0.61	2.72	35.66	0.14	2.05
0.70	39.34	0.63	1.65	33.84	0.34	2.22	31.10	0.33	1.19
0.75	35.47	0.40	1.28	33.04	0.30	1.66	30.58	0.15	1.24
0.80	35.20	0.57	2.42	31.75	0.26	1.15	28.24	0.17	1.32
0.85	32.93	0.37	1.86	29.20	0.26	0.89	26.06	0.12	1.83
0.90	30.41	0.47	2.53	27.10	0.18	1.13	23.72	0.09	1.09
0.95	27.87	0.11	1.31	25.06	0.16	0.93	23.42	0.08	1.60

Table C.3: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	5.60	0.00	0.52	4.43	1.49	0.51	3.60	0.00	0.52
0.55	5.20	0.00	0.63	4.20	0.00	0.42	3.30	0.00	0.95
0.60	4.00	0.00	0.00	4.00	0.00	0.00	2.60	0.00	0.52
0.65	4.00	0.00	0.00	4.00	0.00	0.00	2.60	0.00	0.52
0.70	4.00	0.00	0.00	4.00	0.00	0.00	2.50	0.00	0.53
0.75	4.00	0.00	0.00	3.30	0.00	0.48	2.50	0.00	0.53
0.80	3.90	0.00	0.32	3.00	0.00	0.00	2.50	0.00	0.53
0.85	3.70	0.00	0.48	3.00	0.00	0.00	2.50	0.00	0.53
0.90	3.00	0.00	0.00	3.00	0.00	0.00	2.50	0.00	0.53
0.95	3.00	0.00	0.00	3.00	0.00	0.00	2.50	0.00	0.53

(e) Leukaemia

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	51.13	1.14	3.56	46.74	0.52	3.50	40.81	1.01	2.61
0.55	42.21	0.84	3.56	36.61	0.80	3.65	35.22	0.27	2.35
0.60	41.55	0.82	2.69	34.65	0.63	4.36	31.24	0.44	2.87
0.65	37.15	0.96	3.27	32.65	0.28	2.85	29.70	0.39	2.06
0.70	32.80	0.76	3.56	29.93	0.46	2.75	26.26	0.10	1.62
0.75	33.05	1.03	3.87	29.33	0.48	2.94	24.88	0.40	2.04
0.80	32.98	0.74	3.15	27.79	0.36	2.98	23.99	0.35	2.08
0.85	30.10	0.48	3.17	24.97	0.42	2.32	21.47	0.32	2.20
0.90	29.17	0.59	2.72	23.25	0.20	2.66	21.42	0.20	1.86
0.95	25.19	0.43	2.05	21.07	0.26	2.17	18.87	0.07	1.54

Table C.4: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	minInstPerRule=50%			minInstPerRule=60%			minInstPerRule=70%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	75.95	2.54	11.39	82.95	1.80	8.77	87.34	1.67	6.67
0.55	82.98	2.14	10.76	86.86	1.87	7.51	90.85	0.95	6.22
0.60	84.42	1.73	6.55	86.05	1.45	7.61	89.76	0.93	5.30
0.65	83.52	1.87	8.91	85.35	2.35	7.51	91.45	0.44	4.98
0.70	85.78	1.64	7.24	88.09	1.59	9.28	91.49	0.69	4.23
0.75	87.99	1.20	6.87	90.47	1.32	5.31	90.54	0.55	7.43
0.80	89.73	0.55	7.62	93.01	0.77	5.44	89.99	0.85	9.96
0.85	91.77	1.01	5.24	92.20	1.04	5.13	88.89	1.26	9.35
0.90	91.30	1.06	5.39	85.67	0.80	8.38	87.49	0.60	6.37
0.95	88.31	0.82	8.52	84.74	0.58	7.32	91.43	0.38	3.04

(b) Iris

	minInstPerRule=40%			minInstPerRule=50%			minInstPerRule=60%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	87.73	0.64	11.41	87.53	0.55	11.67	86.73	0.73	11.41
0.55	87.20	0.42	11.85	87.87	0.53	10.96	86.33	0.47	10.73
0.60	87.47	0.53	11.57	85.33	0.77	9.73	85.27	0.66	11.08
0.65	87.07	0.64	10.88	86.00	0.54	10.31	88.80	0.76	9.71
0.70	86.40	0.78	11.66	88.33	0.57	10.03	93.33	0.00	7.03
0.75	89.73	0.56	10.49	87.53	0.32	10.68	75.33	0.00	7.06
0.80	92.20	0.32	7.02	90.67	0.00	9.00	76.67	0.00	6.48
0.85	92.67	0.00	6.63	90.00	0.00	9.56	76.67	0.00	6.48
0.90	80.67	0.00	7.98	76.00	0.00	6.44	75.33	0.00	6.32
0.95	78.00	0.00	7.06	76.00	0.00	6.44	63.33	0.00	11.86

(c) Glass

	minInstPerRule=50%			minInstPerRule=60%			minInstPerRule=70%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	53.74	1.93	8.98	55.19	1.54	12.39	50.25	1.57	9.09
0.55	53.09	2.74	11.90	51.47	1.44	11.89	49.37	1.03	11.01
0.60	55.63	1.64	9.66	45.87	1.87	8.61	48.02	1.82	12.45
0.65	49.77	0.54	11.74	42.94	1.18	14.35	45.10	0.87	13.62
0.70	45.82	1.03	15.35	45.65	0.57	15.75	48.79	0.64	16.82
0.75	42.00	0.62	16.89	46.48	0.80	17.13	44.74	0.48	11.85
0.80	47.54	0.65	16.09	45.18	0.47	13.78	42.49	0.43	14.14
0.85	44.61	0.38	15.31	41.07	0.67	16.74	33.61	0.28	11.00
0.90	42.91	0.95	11.72	37.55	0.36	10.50	25.93	0.33	6.95
0.95	45.22	0.57	14.21	34.44	0.36	10.23	26.67	0.39	7.97

Table C.4: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	72.24	0.00	8.00	77.16	1.34	7.50	61.22	0.00	18.28
0.55	74.35	0.00	8.39	78.77	0.00	6.73	62.78	0.00	16.60
0.60	71.89	0.00	6.90	77.46	0.00	5.17	68.65	0.00	18.12
0.65	71.89	0.00	6.90	77.46	0.00	5.17	68.65	0.00	18.12
0.70	70.84	0.00	6.50	76.93	0.00	4.97	67.60	0.00	19.25
0.75	72.16	0.00	7.38	81.45	0.00	7.09	67.60	0.00	19.25
0.80	74.83	0.00	7.72	82.78	0.00	6.11	67.60	0.00	19.25
0.85	79.02	0.00	5.92	82.78	0.00	6.11	67.60	0.00	19.25
0.90	82.78	0.00	6.11	82.78	0.00	6.11	67.60	0.00	19.25
0.95	82.78	0.00	6.11	82.78	0.00	6.11	67.60	0.00	19.25

(e) Leukaemia

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	83.66	2.33	17.62	86.97	1.24	17.30	89.29	2.03	15.99
0.55	87.93	3.09	16.41	90.58	1.58	14.68	90.38	1.13	16.24
0.60	88.48	1.82	16.87	90.35	1.83	14.49	92.68	1.04	10.05
0.65	89.97	2.19	14.65	90.94	1.19	16.25	92.58	2.16	11.01
0.70	90.93	2.05	13.66	91.21	1.09	16.03	92.18	1.87	10.30
0.75	88.59	0.83	15.86	92.23	1.89	11.44	93.57	0.79	11.54
0.80	90.86	2.32	15.23	91.43	1.51	12.92	93.78	0.99	9.60
0.85	91.23	1.73	12.09	94.12	1.54	9.36	95.63	1.55	7.07
0.90	91.15	1.56	12.40	94.14	1.31	8.43	93.35	1.21	9.95
0.95	92.48	1.05	11.19	93.56	0.95	8.28	93.11	1.09	8.52

C.4 The Richness of the Hypothesis Language

Table C.5 and Table C.6 respectively present the average number of terms and percentage classification accuracy of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language. The associated standard error of the mean (SEM) and standard deviation (STD) are also given. 'Simple' denotes rulebases containing simple propositional rules, 'Disjunction' denotes those containing rules with internal disjunction between attribute values, 'Negation' denotes rules containing negated terms, and 'Hedges' denotes rules that contain both negated terms and hedges. Graphs generated from these tables are discussed in Section 6.4 on page 103 in the main text.

Table C.5: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	8.00	0.00	0.67	9.85	0.45	2.44	26.82	0.44	1.54	26.37	0.23	1.48
0.55	7.91	0.03	0.28	8.05	0.84	1.83	23.41	0.53	1.96	21.87	0.53	1.85
0.60	7.67	0.05	0.68	7.89	0.38	2.06	23.07	0.30	1.40	19.29	0.52	1.80
0.65	6.50	0.00	0.71	7.93	0.59	2.04	23.18	0.27	1.36	16.96	0.58	2.17
0.70	5.80	0.00	0.63	6.76	0.38	1.55	21.06	0.33	1.36	18.83	0.64	1.72
0.75	5.00	0.00	0.00	5.68	0.10	0.92	18.84	0.30	1.30	17.48	0.43	1.45
0.80	4.90	0.00	0.32	5.85	0.25	0.99	16.83	0.29	2.16	16.31	0.49	1.52
0.85	4.70	0.00	0.48	5.94	0.27	1.04	16.38	0.19	1.47	15.32	0.38	1.45
0.90	4.50	0.00	0.53	6.34	0.24	0.94	17.27	0.28	1.43	14.32	0.30	1.93
0.95	4.00	0.00	0.00	6.10	0.28	0.87	16.54	0.29	1.34	15.55	0.40	1.32

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	8.00	0.00	0.00	9.22	0.45	1.58	11.82	0.04	0.40	11.58	0.10	0.51
0.55	8.00	0.00	0.00	8.97	0.28	1.48	11.67	0.08	0.62	11.51	0.03	0.53
0.60	6.70	0.00	0.48	8.67	0.24	1.02	11.53	0.05	0.69	10.97	0.05	0.28
0.65	6.70	0.00	0.48	8.90	0.36	1.06	10.76	0.07	0.44	10.00	0.00	0.00
0.70	6.00	0.00	0.00	8.68	0.23	1.01	10.00	0.00	0.00	10.00	0.00	0.00
0.75	6.00	0.00	0.00	8.67	0.27	1.07	9.90	0.00	0.32	10.00	0.00	0.00
0.80	5.00	0.00	0.00	7.41	0.12	0.50	9.90	0.00	0.32	9.70	0.00	0.48
0.85	5.00	0.00	0.00	7.37	0.17	0.48	9.90	0.00	0.32	9.60	0.00	0.70
0.90	4.00	0.00	0.00	5.14	0.05	0.36	10.00	0.00	0.00	9.81	0.03	0.41
0.95	4.00	0.00	0.00	5.11	0.06	0.35	9.60	0.00	0.70	8.90	0.00	0.32

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	27.04	0.05	1.51	29.65	1.02	3.48	46.82	0.18	0.87	46.45	0.36	1.19
0.55	24.73	0.11	1.43	29.87	0.68	3.06	44.63	0.19	1.22	43.99	0.38	2.15
0.60	21.30	0.00	0.95	24.33	0.98	2.85	39.45	0.53	2.21	39.87	0.65	2.63
0.65	13.36	0.15	1.13	16.98	0.43	2.33	42.12	0.28	1.93	39.65	0.30	1.73
0.70	13.33	0.07	0.76	14.93	0.42	2.50	39.34	0.63	1.65	36.58	0.76	2.40
0.75	10.70	0.00	0.95	12.65	0.50	2.58	35.47	0.40	1.28	35.90	0.48	1.86
0.80	9.90	0.00	1.10	10.66	0.47	2.65	35.20	0.57	2.42	34.43	0.59	2.43
0.85	9.90	0.00	1.10	10.97	0.48	2.46	32.93	0.37	1.86	32.14	0.37	1.64
0.90	8.11	0.03	0.74	9.67	0.31	1.42	30.41	0.47	2.53	29.77	0.43	2.13
0.95	6.80	0.00	0.63	9.10	0.25	1.26	27.87	0.11	1.31	27.98	0.18	1.08

Table C.5: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	3.10	0.00	0.32	2.91	0.07	1.39	5.60	0.00	0.52	5.60	0.00	0.52
0.55	3.10	0.00	0.32	2.92	0.08	1.41	5.20	0.00	0.63	5.60	0.00	0.52
0.60	3.10	0.00	0.32	2.88	0.08	1.39	4.00	0.00	0.00	5.00	0.00	0.00
0.65	3.10	0.00	0.32	2.86	0.07	1.36	4.00	0.00	0.00	4.50	0.00	0.53
0.70	3.80	0.00	0.63	2.28	0.13	0.53	4.00	0.00	0.00	4.00	0.00	0.00
0.75	3.80	0.00	0.63	2.31	0.10	0.55	4.00	0.00	0.00	4.00	0.00	0.00
0.80	3.80	0.00	0.63	2.41	0.11	0.58	3.90	0.00	0.32	3.90	0.00	0.32
0.85	3.50	0.00	0.71	2.25	0.13	0.45	3.70	0.00	0.48	4.00	0.00	0.00
0.90	2.80	0.00	0.63	2.32	0.08	0.52	3.00	0.00	0.00	4.00	0.00	0.00
0.95	2.10	0.00	0.32	2.06	0.05	0.19	3.00	0.00	0.00	3.00	0.00	0.00

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	4.21	0.09	0.49	7.59	0.65	1.34	40.81	1.01	2.61	40.83	0.87	2.28
0.55	3.40	0.00	0.70	6.14	0.30	1.30	35.22	0.27	2.35	34.42	0.35	2.11
0.60	2.50	0.00	0.53	3.81	0.11	0.95	31.24	0.44	2.87	30.21	0.61	2.57
0.65	2.00	0.00	0.00	3.00	0.00	0.00	29.70	0.39	2.06	27.37	0.54	2.72
0.70	2.00	0.00	0.00	2.75	0.10	0.44	26.26	0.10	1.62	24.95	0.21	1.93
0.75	2.00	0.00	0.00	3.12	0.06	0.32	24.88	0.40	2.04	24.18	0.31	2.12
0.80	2.00	0.00	0.00	3.00	0.00	0.00	23.99	0.35	2.08	23.35	0.27	1.76
0.85	2.00	0.00	0.00	3.25	0.08	0.45	21.47	0.32	2.20	21.21	0.31	2.11
0.90	2.00	0.00	0.00	3.13	0.12	0.34	21.42	0.20	1.86	21.43	0.11	1.85
0.95	2.00	0.00	0.00	3.11	0.07	0.28	18.87	0.07	1.54	18.88	0.08	1.53

Table C.6: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	85.59	0.25	8.62	87.76	2.17	7.00	82.95	1.80	8.77	82.34	2.21	8.94
0.55	86.57	0.78	7.70	87.65	0.99	7.45	86.86	1.87	7.51	83.88	2.20	9.00
0.60	86.54	0.53	7.96	87.15	1.73	7.23	86.05	1.45	7.61	86.83	1.21	7.94
0.65	90.81	0.30	4.80	84.73	2.00	8.61	85.35	2.35	7.51	87.68	1.38	9.07
0.70	90.34	0.00	6.77	84.93	1.20	8.73	88.09	1.59	9.28	88.39	0.97	8.30
0.75	92.67	0.00	4.73	79.82	2.64	10.41	90.47	1.32	5.31	90.02	1.12	6.15
0.80	92.67	0.00	4.73	78.60	2.47	9.70	93.01	0.77	5.44	90.54	0.99	7.30
0.85	92.67	0.00	4.73	75.71	2.40	9.45	92.20	1.04	5.13	91.37	0.68	7.33
0.90	92.70	0.00	4.57	76.28	2.83	11.75	85.67	0.80	8.38	90.68	0.86	5.99
0.95	90.02	0.00	8.52	80.55	2.21	8.63	84.74	0.58	7.32	84.74	1.12	7.74

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	91.33	0.00	7.73	94.20	1.26	6.25	86.73	0.73	11.41	85.80	1.00	9.13
0.55	91.33	0.00	7.73	94.47	1.51	6.18	86.33	0.47	10.73	85.27	1.11	9.30
0.60	92.00	0.00	6.13	85.40	2.16	16.23	85.27	0.66	11.08	89.60	1.00	7.79
0.65	92.00	0.00	6.13	81.87	4.33	16.10	88.80	0.76	9.71	95.20	0.28	6.28
0.70	95.33	0.00	4.50	82.93	5.61	15.40	93.33	0.00	7.03	93.47	0.28	7.63
0.75	86.67	0.00	12.17	76.00	2.65	13.72	75.33	0.00	7.06	91.33	0.00	7.73
0.80	87.33	0.00	10.63	76.67	3.89	14.34	76.67	0.00	6.48	93.40	0.21	6.24
0.85	94.00	0.00	5.84	83.80	5.31	14.95	76.67	0.00	6.48	93.33	0.00	6.29
0.90	82.67	0.00	10.98	81.40	1.46	10.38	75.33	0.00	6.32	74.07	0.21	5.94
0.95	82.67	0.00	10.98	81.87	0.76	9.76	63.33	0.00	11.86	74.67	0.00	5.26

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	46.92	0.74	7.44	45.76	2.17	10.27	53.74	1.93	8.98	54.85	2.68	10.04
0.55	41.50	0.83	11.16	45.15	2.11	9.43	53.09	2.74	11.90	52.32	1.82	11.66
0.60	40.64	0.00	12.12	41.95	2.22	8.41	55.63	1.64	9.66	54.37	1.33	10.35
0.65	30.96	0.39	5.88	37.23	1.09	7.76	49.77	0.54	11.74	54.34	1.71	13.70
0.70	32.93	0.19	4.96	31.18	1.24	7.77	45.82	1.03	15.35	48.18	1.57	10.41
0.75	7.45	0.00	3.13	10.63	0.89	6.16	42.00	0.62	16.89	48.10	1.76	14.45
0.80	4.23	0.00	2.56	6.96	0.79	2.34	47.54	0.65	16.09	50.03	1.87	15.16
0.85	6.90	0.00	5.49	9.70	0.51	5.86	44.61	0.38	15.31	44.77	0.33	14.42
0.90	30.22	0.14	11.76	32.14	0.66	10.55	42.91	0.95	11.72	45.47	1.01	12.71
0.95	15.26	0.00	8.21	22.48	1.10	12.42	45.22	0.57	14.21	45.23	0.43	14.25

Table C.6: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	77.03	0.00	28.65	90.18	0.70	13.51	72.24	0.00	8.00	78.27	0.55	6.07
0.55	77.03	0.00	28.65	90.10	0.47	13.55	74.35	0.00	8.39	78.27	0.55	6.07
0.60	77.03	0.00	28.65	90.42	0.62	13.50	71.89	0.00	6.90	68.35	0.54	6.91
0.65	77.03	0.00	28.65	90.36	0.81	13.57	71.89	0.00	6.90	69.41	0.54	7.35
0.70	30.07	0.00	39.03	68.32	0.00	12.77	70.84	0.00	6.50	71.63	0.00	6.65
0.75	30.07	0.00	39.03	68.32	0.00	12.77	72.16	0.00	7.38	73.99	0.00	8.23
0.80	30.07	0.00	39.03	68.32	0.00	12.77	74.83	0.00	7.72	77.21	0.00	8.79
0.85	37.14	0.00	40.31	68.32	0.00	12.77	79.02	0.00	5.92	79.31	0.00	5.58
0.90	68.32	0.00	12.77	68.32	0.00	12.77	82.78	0.00	6.11	79.31	0.00	5.58
0.95	31.13	0.00	40.37	31.13	0.00	40.37	82.78	0.00	6.11	82.78	0.00	6.11

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	92.57	2.16	8.99	90.76	2.65	11.99	89.29	2.03	15.99	87.85	2.65	17.02
0.55	91.07	0.00	12.56	87.17	1.96	11.05	90.38	1.13	16.24	90.00	0.97	16.73
0.60	87.74	0.00	9.54	86.71	1.98	10.93	92.68	1.04	10.05	91.99	1.05	16.18
0.65	87.74	0.00	9.54	87.56	1.36	11.40	92.58	2.16	11.01	91.26	1.01	15.95
0.70	79.64	0.00	13.81	87.07	2.49	15.36	92.18	1.87	10.30	93.33	1.69	11.13
0.75	72.26	0.00	12.87	85.59	1.33	15.71	93.57	0.79	11.54	94.90	0.70	8.45
0.80	71.57	1.18	19.80	85.56	1.68	14.95	93.78	0.99	9.60	94.07	0.94	9.23
0.85	67.86	0.00	11.60	80.38	3.37	15.33	95.63	1.55	7.07	95.19	1.13	7.34
0.90	64.26	0.66	15.20	83.62	2.57	11.00	93.35	1.21	9.95	93.83	0.95	8.91
0.95	62.26	1.09	14.09	80.24	2.59	13.59	93.11	1.09	8.52	92.96	1.02	8.22

Appendix D

Supplementary Results to Chapter 7

This appendix provides detailed and/or supplementary results for findings discussed in Chapter 7, which compares the different rule learning strategies. All results presented in Section D.1 pertain to *FRANTIC* rulebases induced using a full iterative rule learning strategy. Section D.2 contains statistics relating to rulebases induced using a simultaneous rule learning strategy. Equivalent results of *FRANTIC* rulebases induced using a simplified iterative rule learning strategy are found in Appendix C.

Tables generally present the main statistic – whether it be average accuracy or average terms in a rulebase/terms per rule/rules per rulebase – and associated standard error of the mean (SEM) and standard deviation (STD). Since 10 ten-fold cross-validations are carried out, the STD is the average of the 10 standard deviations obtained from each of the 10 cross-validations (this gives an indication of *FRANTIC*'s robustness to the different training sets used in one ten-fold cross-validation), and the SEM is the standard deviation of the 10 average accuracies from each of the 10 cross-validations (this gives an estimate of the reliability of the main statistic presented in a table).

Note that in all tables and figures 'Simple' denotes rulebases containing simple propositional rules, 'Disjunction' denotes those containing rules with internal disjunction between attribute values, 'Negation' denotes rules containing negated terms, and 'Hedges' denotes rules that contain both negated terms and linguistic hedges.

D.1 Iterative Rule Learning Performance

Table D.1 gives the accuracy of full IRL rulebases induced using a specific `minInstPerRule` value for each dataset for different forms of the hypothesis language (parameter settings are provided in Table B.5(b) on page 222). Tables D.2 and D.3 provide the corresponding complexity details for the rulebases with regards to number of rules and number of terms respectively.

Table D.4 provides the accuracy of rulebases induced using different values for the `minInstPerRule` parameter, and for rules containing negated terms in the antecedent. Tables D.5 and D.6 provide the corresponding rulebase complexity statistics with regards to number of rules and number of terms respectively.

Note that one column in each of the subtables in Table D.4 duplicates the results of the 'Negation' column in each of the corresponding subtables in Table D.1. The statistics are replicated for ease of reference.

These results, and/or figures generated from them, are discussed in Section 7.1 in the main text.

Table D.1: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	92.77	0.22	7.51	88.03	2.59	6.90	92.85	1.51	5.47	90.54	1.92	6.83
0.55	90.51	0.31	7.64	88.40	2.28	7.02	92.65	1.49	5.69	92.27	1.85	5.33
0.60	89.67	0.39	7.45	86.85	3.17	8.25	92.89	1.38	4.15	92.73	1.55	5.58
0.65	89.06	0.29	5.99	87.34	2.41	9.34	91.54	1.63	5.43	91.50	1.61	7.36
0.70	89.32	0.45	9.29	87.20	0.70	9.00	92.01	1.68	6.43	93.55	1.55	5.04
0.75	89.32	0.40	7.61	87.03	2.06	8.93	93.53	1.83	5.13	91.43	0.96	5.67
0.80	88.67	0.27	7.32	81.80	2.17	12.16	93.89	1.22	4.98	93.51	2.03	6.15
0.85	86.48	0.27	6.87	80.90	3.62	11.32	92.28	1.21	5.48	93.08	1.24	4.62
0.90	89.18	0.29	6.74	79.47	2.98	12.24	93.54	0.47	4.76	92.76	0.86	6.15
0.95	88.50	0.73	8.12	83.21	3.01	11.70	90.76	0.65	3.81	90.86	0.98	6.39

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	93.33	0.00	7.03	93.53	0.55	6.50	89.33	0.63	9.89	89.53	1.00	9.88
0.55	94.00	0.00	5.84	93.80	0.32	6.27	89.27	0.66	9.70	89.93	1.24	9.14
0.60	95.33	0.00	5.49	94.00	0.54	6.02	91.80	0.83	10.57	89.20	1.25	9.88
0.65	94.67	0.00	6.13	85.67	5.78	13.16	83.93	0.66	12.77	90.93	0.64	8.92
0.70	95.33	0.00	5.49	81.60	3.05	16.89	92.73	0.21	10.03	95.27	0.21	6.30
0.75	90.00	0.00	5.67	76.67	2.67	13.21	91.53	0.32	10.26	94.67	0.31	6.20
0.80	88.00	0.00	7.57	77.93	3.48	13.71	90.67	0.00	9.00	93.33	0.00	6.29
0.85	94.00	0.00	5.84	86.33	4.38	15.49	90.00	0.00	9.56	93.40	0.21	6.24
0.90	74.00	0.00	4.92	73.53	0.83	8.38	76.00	0.00	6.44	74.73	0.21	7.44
0.95	78.00	0.00	6.32	73.80	1.18	7.27	76.00	0.00	6.44	75.33	0.00	6.32

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	38.66	0.00	11.32	41.10	2.40	9.20	55.33	1.84	11.13	58.41	2.21	11.20
0.55	28.91	0.54	12.65	40.15	1.26	9.01	54.56	2.25	12.65	55.12	2.01	10.85
0.60	29.77	0.30	10.87	38.74	2.35	9.39	45.61	1.62	8.66	49.48	1.88	12.28
0.65	41.55	0.76	11.18	38.52	1.35	10.08	42.75	1.13	10.47	53.46	1.80	12.08
0.70	35.97	0.54	9.83	34.59	2.60	10.35	41.78	1.64	7.70	46.54	2.58	8.12
0.75	15.07	0.38	7.48	11.83	0.77	9.31	41.94	0.81	13.72	43.15	2.23	13.17
0.80	8.77	0.49	6.16	5.83	0.69	5.36	43.48	0.65	12.31	47.22	2.35	13.44
0.85	10.76	0.37	7.02	11.25	0.86	8.45	44.33	0.68	13.40	45.16	0.70	13.24
0.90	30.86	0.14	10.10	30.96	1.32	9.23	44.41	0.51	8.97	48.50	0.94	11.41
0.95	18.21	0.54	7.71	16.32	1.93	8.11	42.67	0.40	13.73	42.76	0.52	13.54

Table D.1: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	80.82	0.00	14.20	89.13	0.89	5.55	92.67	0.48	7.36	91.43	0.46	4.66
0.55	80.55	0.00	13.93	89.29	0.41	5.73	92.84	0.23	6.41	91.33	0.64	4.62
0.60	80.55	0.00	13.93	89.43	0.54	5.69	88.38	0.14	5.75	89.17	1.13	7.03
0.65	80.55	0.00	13.93	89.16	0.51	5.96	88.20	0.35	5.60	88.51	0.48	6.59
0.70	66.51	0.00	7.22	76.70	1.37	14.91	91.41	0.13	5.29	89.46	0.14	6.56
0.75	66.51	0.00	7.22	76.18	0.97	14.20	91.48	0.17	5.56	89.74	0.62	6.21
0.80	59.75	0.00	15.84	76.06	1.37	14.36	93.07	0.00	4.97	92.13	0.74	5.69
0.85	59.75	0.00	15.84	73.23	1.11	14.77	90.76	0.00	7.45	92.65	1.37	5.45
0.90	59.75	0.00	15.84	74.12	1.21	15.32	91.03	0.00	7.76	90.77	0.00	7.56
0.95	69.94	0.00	11.40	66.30	0.61	13.65	88.62	0.17	9.16	90.65	0.14	7.86

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	83.88	3.70	15.34	93.79	2.73	8.16	86.54	3.94	16.53	84.89	2.41	16.66
0.55	91.69	1.64	12.95	89.26	1.60	9.47	91.16	2.36	13.08	88.83	2.72	16.73
0.60	90.21	2.17	10.17	90.60	1.75	9.95	91.85	2.21	11.07	90.17	1.87	14.86
0.65	90.31	1.49	14.69	88.43	3.03	10.90	91.51	3.08	13.32	90.77	1.95	13.52
0.70	85.82	1.11	14.08	92.21	2.71	9.25	93.12	1.60	11.91	91.46	2.16	12.60
0.75	91.36	2.27	10.04	95.07	1.72	8.39	92.30	1.82	11.61	92.23	1.61	11.13
0.80	91.63	2.14	12.64	89.01	1.68	12.71	90.51	2.56	12.54	90.52	2.06	12.53
0.85	93.29	1.09	11.33	90.83	2.52	12.62	91.08	2.43	12.12	92.80	3.02	12.35
0.90	85.77	1.24	7.20	83.13	2.66	16.07	92.16	3.58	13.34	92.51	2.86	12.42
0.95	89.56	0.87	12.44	83.90	1.69	15.44	90.88	1.88	14.38	92.60	1.97	11.79

Table D.2: Number of rules, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	6.20	0.00	0.42	4.00	0.07	0.17	6.00	0.00	0.00	6.00	0.00	0.00
0.55	6.00	0.00	0.00	4.20	0.18	0.42	6.00	0.00	0.00	6.00	0.00	0.00
0.60	6.00	0.00	0.00	4.15	0.14	0.47	6.00	0.00	0.00	6.00	0.00	0.00
0.65	6.00	0.00	0.00	4.13	0.16	0.58	6.00	0.00	0.00	6.00	0.00	0.00
0.70	6.10	0.00	0.32	4.29	0.18	0.62	6.00	0.00	0.00	6.00	0.00	0.00
0.75	6.34	0.05	0.50	4.66	0.12	0.59	6.00	0.00	0.00	6.00	0.00	0.00
0.80	6.00	0.00	0.00	4.56	0.16	0.59	6.00	0.00	0.00	5.85	0.08	0.34
0.85	6.00	0.00	0.00	4.64	0.10	0.50	5.51	0.10	0.54	4.98	0.23	0.67
0.90	6.00	0.00	0.00	4.78	0.09	0.42	5.03	0.12	0.54	4.55	0.20	0.58
0.95	5.60	0.00	0.52	4.68	0.11	0.52	4.67	0.12	0.54	4.23	0.13	0.39

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	6.00	0.00	0.00	3.37	0.13	0.49	5.00	0.00	0.00	5.00	0.00	0.00
0.55	5.00	0.00	0.00	3.34	0.20	0.46	5.00	0.00	0.00	5.00	0.00	0.00
0.60	5.00	0.00	0.00	3.23	0.12	0.40	5.00	0.00	0.00	5.00	0.00	0.00
0.65	4.10	0.00	0.32	3.06	0.07	0.17	4.70	0.00	0.48	4.22	0.04	0.64
0.70	3.40	0.00	0.52	3.00	0.00	0.00	4.30	0.00	0.48	3.16	0.07	0.38
0.75	3.80	0.00	0.92	3.31	0.03	0.49	4.00	0.00	0.00	3.05	0.07	0.16
0.80	3.40	0.00	0.52	3.32	0.04	0.49	3.00	0.00	0.00	3.00	0.00	0.00
0.85	3.00	0.00	0.00	3.00	0.00	0.00	3.00	0.00	0.00	3.00	0.00	0.00
0.90	5.00	0.00	0.00	3.90	0.00	0.32	3.00	0.00	0.00	3.00	0.00	0.00
0.95	5.00	0.00	0.00	3.88	0.04	0.34	3.00	0.00	0.00	3.00	0.00	0.00

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	12.00	0.00	0.00	8.37	0.15	0.67	12.00	0.00	0.00	12.00	0.00	0.00
0.55	12.00	0.00	0.00	8.06	0.19	0.73	12.00	0.00	0.00	12.00	0.00	0.00
0.60	11.30	0.00	0.67	8.66	0.13	0.77	11.27	0.07	0.46	11.44	0.15	0.50
0.65	11.00	0.00	0.47	9.18	0.13	0.70	10.79	0.10	0.55	10.78	0.10	0.53
0.70	11.30	0.00	0.67	9.52	0.16	0.99	10.43	0.11	0.64	10.06	0.08	0.48
0.75	10.40	0.00	1.26	9.10	0.09	0.65	9.48	0.10	0.64	9.47	0.09	0.66
0.80	11.00	0.00	0.94	9.60	0.00	0.52	9.37	0.07	0.67	9.32	0.04	0.82
0.85	11.80	0.00	1.55	10.03	0.07	0.70	9.88	0.08	0.34	9.50	0.09	0.66
0.90	12.30	0.00	0.48	10.43	0.09	0.57	8.91	0.14	0.98	8.23	0.16	0.77
0.95	13.00	0.00	0.67	11.35	0.16	0.85	8.83	0.15	0.69	8.41	0.10	0.58

Table D.2: Number of rules, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction threshold	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	3.10	0.00	0.32	2.13	0.05	0.35	4.00	0.00	0.00	4.00	0.00	0.00
0.55	3.00	0.00	0.00	2.03	0.05	0.09	4.00	0.00	0.00	4.00	0.00	0.00
0.60	3.00	0.00	0.00	2.04	0.05	0.13	3.80	0.00	0.42	3.81	0.03	0.41
0.65	3.00	0.00	0.00	2.02	0.04	0.06	3.81	0.03	0.41	3.80	0.00	0.42
0.70	3.00	0.00	0.00	3.00	0.00	0.00	3.80	0.00	0.42	3.80	0.00	0.42
0.75	3.00	0.00	0.00	3.00	0.00	0.00	3.80	0.00	0.42	3.80	0.00	0.42
0.80	3.00	0.00	0.00	3.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
0.85	3.00	0.00	0.00	3.00	0.00	0.00	3.90	0.00	0.32	3.97	0.05	0.09
0.90	3.00	0.00	0.00	3.00	0.00	0.00	3.90	0.00	0.32	3.90	0.00	0.32
0.95	3.00	0.00	0.00	3.00	0.00	0.00	3.90	0.00	0.32	3.90	0.00	0.32

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction threshold	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	5.28	0.08	0.46	2.22	0.14	0.37	6.00	0.00	0.00	6.00	0.00	0.00
0.55	4.00	0.00	0.00	2.27	0.18	0.40	4.00	0.00	0.00	4.00	0.00	0.00
0.60	4.02	0.04	0.06	2.17	0.09	0.38	4.00	0.00	0.00	4.00	0.00	0.00
0.65	4.00	0.00	0.00	2.49	0.16	0.50	4.00	0.00	0.00	3.98	0.06	0.04
0.70	4.26	0.07	0.75	2.43	0.12	0.51	4.00	0.00	0.00	3.82	0.09	0.39
0.75	4.08	0.11	0.53	2.82	0.06	0.38	4.00	0.00	0.00	3.64	0.13	0.48
0.80	4.06	0.05	0.19	2.78	0.04	0.43	4.00	0.00	0.00	3.67	0.18	0.46
0.85	4.21	0.03	0.43	3.04	0.08	0.22	3.93	0.08	0.18	3.60	0.17	0.55
0.90	4.02	0.04	0.06	3.02	0.04	0.11	3.62	0.10	0.50	3.63	0.08	0.50
0.95	4.38	0.09	0.69	3.17	0.05	0.39	4.00	0.00	0.00	3.63	0.08	0.50

Table D.3: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	16.12	0.04	4.97	13.58	0.55	2.73	48.23	0.68	2.65	46.49	0.62	2.44
0.55	13.32	0.04	0.83	14.29	1.67	3.46	43.57	0.40	2.51	41.37	0.76	1.92
0.60	13.21	0.03	0.79	13.88	0.77	3.93	44.12	0.67	2.85	41.51	0.62	2.31
0.65	12.91	0.03	0.73	13.42	0.80	3.60	43.02	0.53	1.89	39.97	0.47	2.32
0.70	13.06	0.20	3.12	13.71	0.89	3.94	41.63	0.53	1.89	38.82	0.69	2.54
0.75	12.92	0.45	3.42	14.53	0.59	3.75	44.59	0.50	2.72	42.51	0.79	3.20
0.80	10.64	0.10	1.19	12.63	0.47	2.99	45.49	0.51	2.35	42.72	1.30	5.13
0.85	10.90	0.00	1.10	10.72	0.85	2.88	40.38	1.40	8.14	33.11	3.21	9.36
0.90	10.30	0.00	1.16	10.73	0.40	2.59	34.49	1.74	7.81	30.29	3.05	8.28
0.95	8.60	0.00	1.84	8.97	0.31	2.43	31.46	1.87	6.86	23.70	2.24	5.90

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	17.50	0.00	0.53	11.53	0.47	2.79	20.00	0.00	0.00	20.00	0.00	0.00
0.55	12.20	0.00	0.42	9.77	0.70	2.04	19.99	0.03	0.03	20.00	0.00	0.00
0.60	11.10	0.00	0.32	9.44	0.21	1.73	19.34	0.05	0.63	19.45	0.08	0.67
0.65	8.40	0.00	0.70	8.80	0.64	1.45	18.05	0.05	2.18	15.60	0.18	3.21
0.70	6.90	0.00	1.20	8.61	0.33	1.12	16.42	0.06	2.48	10.67	0.29	1.58
0.75	7.20	0.00	1.40	9.14	0.16	1.15	14.36	0.10	0.82	10.19	0.26	0.60
0.80	6.20	0.00	0.79	9.29	0.17	1.11	9.50	0.00	0.53	10.00	0.00	0.00
0.85	5.20	0.00	0.42	7.67	0.15	0.93	9.50	0.00	0.53	10.00	0.00	0.00
0.90	7.00	0.00	0.00	8.90	0.00	0.32	10.00	0.00	0.00	10.00	0.00	0.00
0.95	7.00	0.00	0.00	8.88	0.04	0.34	9.90	0.00	0.32	9.90	0.00	0.32

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	45.47	0.21	2.19	41.47	1.79	6.14	93.05	0.20	1.65	92.01	0.62	1.93
0.55	45.06	0.24	3.35	37.44	1.68	4.97	91.68	0.34	2.07	89.06	0.52	2.53
0.60	43.94	0.13	4.60	36.49	1.42	5.49	79.31	0.73	4.89	80.44	1.49	4.98
0.65	32.69	0.11	3.61	27.47	1.26	4.28	72.59	0.66	6.00	70.70	1.69	6.01
0.70	30.98	0.18	4.96	25.96	1.14	5.88	70.22	0.73	6.79	62.43	1.25	5.29
0.75	25.51	0.09	7.33	23.63	0.44	3.87	58.49	1.25	5.83	59.15	1.08	5.76
0.80	24.27	0.13	5.09	21.16	0.53	3.39	54.14	0.51	7.03	54.74	0.59	8.72
0.85	23.02	0.06	5.89	21.73	0.63	3.37	53.33	0.75	4.79	50.69	1.21	6.20
0.90	18.49	0.13	3.74	17.59	0.38	1.93	47.07	1.03	8.35	40.90	1.60	6.20
0.95	16.44	0.13	4.00	17.85	0.63	2.68	42.29	1.44	6.10	38.51	0.77	4.99

Table D.3: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	8.70	0.00	0.95	5.23	0.26	1.11	15.56	0.08	0.96	16.26	0.08	0.96
0.55	8.60	0.00	0.97	5.07	0.29	0.78	15.48	0.06	0.88	16.19	0.07	1.03
0.60	8.60	0.00	0.97	5.11	0.27	0.88	13.30	0.00	2.36	14.36	0.24	2.34
0.65	8.60	0.00	0.97	4.98	0.18	0.61	13.27	0.22	2.31	13.91	0.07	2.73
0.70	7.90	0.00	1.29	6.23	0.21	1.94	11.90	0.00	2.13	13.40	0.00	2.41
0.75	7.90	0.00	1.29	6.16	0.16	1.96	11.90	0.00	2.13	13.20	0.12	2.38
0.80	7.40	0.00	1.26	5.80	0.14	1.39	12.90	0.00	0.99	13.88	0.06	0.81
0.85	7.30	0.00	1.16	5.52	0.13	1.50	12.00	0.00	1.56	13.09	0.34	0.94
0.90	7.30	0.00	1.16	5.60	0.14	1.54	11.50	0.00	1.78	11.80	0.00	1.48
0.95	5.70	0.00	0.82	5.32	0.09	1.17	10.70	0.00	1.57	11.10	0.00	1.60

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	57.50	4.63	25.01	35.46	5.88	17.20	210.70	1.63	9.41	210.41	2.22	9.08
0.55	14.61	0.40	1.52	28.10	8.11	16.30	102.34	1.96	6.74	101.32	1.92	6.16
0.60	12.00	0.91	2.53	19.19	4.11	8.56	94.58	1.74	5.73	91.85	1.73	5.02
0.65	14.07	0.84	4.35	16.27	2.31	8.57	95.84	1.96	7.27	95.75	3.23	11.40
0.70	35.33	1.84	19.00	11.36	1.42	4.69	90.96	2.19	6.70	95.41	2.31	18.96
0.75	8.25	2.17	6.97	10.29	0.46	2.98	91.36	2.32	7.70	96.64	7.24	22.26
0.80	5.24	0.13	0.57	7.35	0.45	2.86	92.39	2.47	9.16	94.70	8.19	19.87
0.85	8.66	0.77	7.51	6.94	1.06	4.09	88.30	2.49	15.75	83.64	6.87	20.63
0.90	4.38	0.80	1.20	5.00	0.44	1.68	70.15	5.91	22.70	74.57	4.22	24.74
0.95	7.89	0.97	7.91	5.76	0.53	2.57	85.80	4.87	8.88	73.71	4.81	23.97

Table D.4: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	minInstPerRule=50%			minInstPerRule=60%			minInstPerRule=70%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	92.57	1.73	5.30	92.27	0.68	4.90	92.85	1.51	5.47
0.55	90.47	1.42	5.60	91.73	2.17	5.68	92.65	1.49	5.69
0.60	92.32	1.08	4.98	94.53	1.26	4.91	92.89	1.38	4.15
0.65	93.32	1.05	4.62	93.32	1.49	5.47	91.54	1.63	5.43
0.70	92.39	1.75	4.85	93.44	1.26	5.64	92.01	1.68	6.43
0.75	92.33	1.96	5.68	92.54	1.20	6.02	93.53	1.83	5.13
0.80	93.01	1.83	5.82	94.63	2.02	5.20	93.89	1.22	4.98
0.85	92.96	1.20	5.80	94.47	1.26	5.16	92.28	1.21	5.48
0.90	90.51	1.27	5.71	92.31	0.92	4.91	93.54	0.47	4.76
0.95	90.90	1.75	5.51	94.00	0.88	4.60	90.76	0.65	3.81

(b) Iris

	minInstPerRule=40%			minInstPerRule=50%			minInstPerRule=60%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	88.87	0.55	10.35	89.33	0.63	9.89	88.73	0.21	10.68
0.55	88.87	0.45	10.58	89.27	0.66	9.70	89.40	0.38	11.19
0.60	89.13	0.63	10.09	91.80	0.83	10.57	89.40	0.38	11.10
0.65	87.60	1.10	10.83	83.93	0.66	12.77	92.73	0.49	9.86
0.70	86.80	0.28	11.53	92.73	0.21	10.03	93.33	0.00	7.03
0.75	93.33	0.77	8.67	91.53	0.32	10.26	75.33	0.00	7.06
0.80	89.40	0.97	10.40	90.67	0.00	9.00	76.67	0.00	6.48
0.85	90.00	0.00	10.54	90.00	0.00	9.56	76.67	0.00	6.48
0.90	74.00	0.00	7.34	76.00	0.00	6.44	75.33	0.00	6.32
0.95	75.33	0.00	5.49	76.00	0.00	6.44	63.33	0.00	11.86

(c) Glass

	minInstPerRule=50%			minInstPerRule=60%			minInstPerRule=70%		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	50.47	1.88	10.29	52.17	1.63	13.58	55.33	1.84	11.13
0.55	48.58	2.23	10.64	54.42	1.63	13.28	54.56	2.25	12.65
0.60	48.55	3.03	11.23	48.73	2.95	10.97	45.61	1.62	8.66
0.65	46.32	1.99	10.87	47.18	1.79	10.39	42.75	1.13	10.47
0.70	48.31	1.61	11.04	45.86	0.64	7.85	41.78	1.64	7.70
0.75	46.71	1.09	7.58	42.95	0.84	10.27	41.94	0.81	13.72
0.80	41.84	1.30	10.61	47.39	0.70	10.79	43.48	0.65	12.31
0.85	46.51	1.78	8.71	40.41	0.76	7.24	44.33	0.68	13.40
0.90	42.93	0.75	9.48	38.83	0.55	14.87	44.41	0.51	8.97
0.95	38.78	1.18	8.83	36.66	0.65	10.91	42.67	0.40	13.73

Table D.4: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	92.67	0.48	7.36	91.07	0.31	7.40	81.41	0.25	8.16
0.55	92.84	0.23	6.41	90.46	0.29	6.65	82.84	0.00	8.39
0.60	88.38	0.14	5.75	82.34	0.44	13.59	80.10	0.00	6.66
0.65	88.20	0.35	5.60	82.45	0.46	13.57	80.64	0.00	7.07
0.70	91.41	0.13	5.29	88.48	0.00	9.06	79.56	0.08	6.81
0.75	91.48	0.17	5.56	88.43	0.17	9.04	77.69	0.28	5.55
0.80	93.07	0.00	4.97	88.75	0.00	9.13	76.40	0.00	5.01
0.85	90.76	0.00	7.45	86.44	0.00	9.60	79.02	0.00	5.92
0.90	91.03	0.00	7.76	86.71	0.00	9.64	82.78	0.00	6.11
0.95	88.62	0.17	9.16	85.13	0.00	8.82	82.78	0.00	6.11

(e) Leukaemia

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	86.54	3.94	16.53	89.17	2.24	16.53	88.21	3.17	16.52
0.55	91.16	2.36	13.08	91.57	2.29	13.93	88.24	2.03	16.16
0.60	91.85	2.21	11.07	90.86	2.70	13.09	89.14	1.73	14.06
0.65	91.51	3.08	13.32	92.66	1.70	12.38	90.99	1.75	14.41
0.70	93.12	1.60	11.91	90.55	2.93	13.19	89.09	3.20	15.20
0.75	92.30	1.82	11.61	90.07	2.41	15.76	88.05	1.84	16.63
0.80	90.51	2.56	12.54	89.40	2.43	15.95	90.18	2.24	14.91
0.85	91.08	2.43	12.12	90.53	2.39	12.84	92.23	2.44	13.59
0.90	92.16	3.58	13.34	89.97	2.44	13.99	88.48	2.28	15.08
0.95	90.88	1.88	14.38	89.80	2.53	14.61	91.10	2.46	12.74

Table D.5: Number of rules, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	`minInstPerRule=30%`			`minInstPerRule=40%`			`minInstPerRule=50%`		
`construction threshold`	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	8.23	0.05	0.44	7.49	0.15	0.52	6.00	0.00	0.00
0.55	8.10	0.00	0.32	6.00	0.00	0.00	6.00	0.00	0.00
0.60	7.40	0.24	0.57	6.00	0.00	0.00	6.00	0.00	0.00
0.65	6.26	0.14	0.45	6.00	0.00	0.00	6.00	0.00	0.00
0.70	6.17	0.12	0.35	6.00	0.00	0.00	6.00	0.00	0.00
0.75	6.00	0.00	0.00	6.00	0.00	0.00	6.00	0.00	0.00
0.80	6.00	0.00	0.00	6.00	0.00	0.00	6.00	0.00	0.00
0.85	6.00	0.00	0.00	6.00	0.00	0.00	5.51	0.10	0.54
0.90	6.00	0.00	0.00	6.00	0.00	0.00	5.03	0.12	0.54
0.95	6.00	0.00	0.00	5.92	0.04	0.25	4.67	0.12	0.54

(b) Iris

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	5.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00
0.55	5.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00
0.60	5.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00
0.65	5.00	0.00	0.00	4.70	0.00	0.48	4.00	0.00	0.00
0.70	5.00	0.00	0.00	4.30	0.00	0.48	3.00	0.00	0.00
0.75	4.01	0.03	0.03	4.00	0.00	0.00	3.00	0.00	0.00
0.80	3.70	0.00	0.48	3.00	0.00	0.00	3.00	0.00	0.00
0.85	3.11	0.03	0.33	3.00	0.00	0.00	3.00	0.00	0.00
0.90	4.00	0.00	0.00	3.00	0.00	0.00	3.00	0.00	0.00
0.95	3.10	0.00	0.32	3.00	0.00	0.00	3.00	0.00	0.00

(c) Glass

	`minInstPerRule=30%`			`minInstPerRule=40%`			`minInstPerRule=50%`		
`construction threshold`	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	14.16	0.12	0.59	12.55	0.10	0.59	12.00	0.00	0.00
0.55	13.97	0.07	0.61	12.01	0.03	0.03	12.00	0.00	0.00
0.60	12.74	0.12	0.57	12.00	0.00	0.00	11.27	0.07	0.46
0.65	12.85	0.10	0.69	11.53	0.15	0.50	10.79	0.10	0.55
0.70	12.06	0.08	0.44	11.20	0.17	0.60	10.43	0.11	0.64
0.75	11.32	0.04	0.49	10.46	0.05	0.70	9.48	0.10	0.64
0.80	11.20	0.07	0.44	10.62	0.06	0.89	9.37	0.07	0.67
0.85	10.89	0.10	0.54	9.25	0.05	0.59	9.88	0.08	0.34
0.90	11.13	0.12	0.47	9.32	0.06	0.64	8.91	0.14	0.98
0.95	11.31	0.03	0.49	9.18	0.15	0.97	8.83	0.15	0.69

Table D.5: Number of rules, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	4.00	0.00	0.00	4.00	0.00	0.00	3.40	0.00	0.52
0.55	4.00	0.00	0.00	4.00	0.00	0.00	3.40	0.00	0.52
0.60	3.80	0.00	0.42	3.40	0.00	0.52	3.00	0.00	0.00
0.65	3.81	0.03	0.41	3.40	0.00	0.52	3.00	0.00	0.00
0.70	3.80	0.00	0.42	3.40	0.00	0.52	3.00	0.00	0.00
0.75	3.80	0.00	0.42	3.40	0.00	0.52	2.70	0.00	0.48
0.80	4.00	0.00	0.00	3.60	0.00	0.52	2.10	0.00	0.32
0.85	3.90	0.00	0.32	3.50	0.00	0.53	2.00	0.00	0.00
0.90	3.90	0.00	0.32	3.50	0.00	0.53	2.00	0.00	0.00
0.95	3.90	0.00	0.32	3.50	0.00	0.53	2.00	0.00	0.00

(e) Leukaemia

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
0.50	6.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
0.55	4.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
0.60	4.00	0.00	0.00	4.00	0.00	0.00	4.00	0.00	0.00
0.65	4.00	0.00	0.00	4.00	0.00	0.00	3.75	0.08	0.45
0.70	4.00	0.00	0.00	3.94	0.05	0.19	3.58	0.11	0.51
0.75	4.00	0.00	0.00	3.58	0.14	0.50	3.05	0.07	0.14
0.80	4.00	0.00	0.00	3.74	0.10	0.45	3.05	0.05	0.16
0.85	3.93	0.08	0.18	3.50	0.16	0.50	3.10	0.11	0.48
0.90	3.62	0.10	0.50	3.61	0.14	0.49	2.67	0.11	0.54
0.95	4.00	0.00	0.00	3.50	0.13	0.58	2.37	0.08	0.50

Table D.6: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	`minInstPerRule=30%`			`minInstPerRule=40%`			`minInstPerRule=50%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	77.14	0.89	6.23	70.78	2.18	7.24	48.23	0.68	2.65
0.55	71.76	0.68	4.72	47.28	0.65	1.97	43.57	0.40	2.51
0.60	66.03	2.60	8.86	44.67	0.65	2.73	44.12	0.67	2.85
0.65	50.06	1.47	6.10	43.76	0.50	1.93	43.02	0.53	1.89
0.70	46.39	1.55	5.06	42.62	0.51	1.68	41.63	0.53	1.89
0.75	43.13	0.43	1.61	41.74	0.66	1.95	44.59	0.50	2.72
0.80	40.22	0.68	1.64	40.21	0.41	1.57	45.49	0.51	2.35
0.85	39.92	0.69	1.55	40.42	0.61	2.07	40.38	1.40	8.14
0.90	38.10	0.37	1.99	38.68	0.65	2.66	34.49	1.74	7.81
0.95	37.97	0.61	2.44	40.75	0.57	5.14	31.46	1.87	6.86

(b) Iris

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	19.98	0.04	0.06	20.00	0.00	0.00	19.83	0.08	0.38
0.55	20.00	0.00	0.00	19.99	0.03	0.03	19.67	0.08	0.60
0.60	20.00	0.00	0.00	19.34	0.05	0.63	19.59	0.07	0.64
0.65	19.92	0.04	0.25	18.05	0.05	2.18	14.76	0.07	0.44
0.70	19.80	0.00	0.42	16.42	0.06	2.48	10.00	0.00	0.00
0.75	14.79	0.19	0.59	14.36	0.10	0.82	9.90	0.00	0.32
0.80	13.29	0.06	2.58	9.50	0.00	0.53	9.90	0.00	0.32
0.85	10.35	0.20	1.40	9.50	0.00	0.53	9.90	0.00	0.32
0.90	14.00	0.00	0.47	10.00	0.00	0.00	10.00	0.00	0.00
0.95	10.30	0.00	0.95	9.90	0.00	0.32	9.60	0.00	0.70

(c) Glass

	`minInstPerRule=30%`			`minInstPerRule=40%`			`minInstPerRule=50%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	118.80	1.14	5.27	102.10	0.77	4.79	93.05	0.20	1.65
0.55	114.00	0.96	6.21	91.18	0.76	2.80	91.68	0.34	2.07
0.60	93.79	1.29	5.83	87.88	0.44	2.65	79.31	0.73	4.89
0.65	92.68	1.02	7.83	79.62	1.65	6.00	72.59	0.66	6.00
0.70	84.48	0.79	4.69	75.39	1.68	6.88	70.22	0.73	6.79
0.75	77.37	0.77	4.83	68.04	0.75	7.49	58.49	1.25	5.83
0.80	69.84	0.82	5.30	64.49	0.67	8.34	54.14	0.51	7.03
0.85	65.45	1.03	6.78	52.13	0.59	7.06	53.33	0.75	4.79
0.90	68.88	1.21	4.86	51.83	0.83	7.14	47.07	1.03	8.35
0.95	68.75	0.76	6.07	49.58	1.19	10.18	42.29	1.44	6.10

Table D.6: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-fullIRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	15.56	0.08	0.96	14.70	0.00	1.06	10.63	0.05	3.15
0.55	15.48	0.06	0.88	14.77	0.05	1.01	10.50	0.00	3.03
0.60	13.30	0.00	2.36	10.30	0.00	3.77	8.00	0.00	0.00
0.65	13.27	0.22	2.31	10.10	0.00	3.70	7.40	0.00	0.52
0.70	11.90	0.00	2.13	9.00	0.00	3.33	7.12	0.04	0.56
0.75	11.90	0.00	2.13	9.02	0.06	3.20	6.07	0.09	1.58
0.80	12.90	0.00	0.99	10.30	0.00	3.80	4.20	0.00	0.63
0.85	12.00	0.00	1.56	9.60	0.00	3.47	3.70	0.00	0.48
0.90	11.50	0.00	1.78	9.10	0.00	3.57	3.00	0.00	0.00
0.95	10.70	0.00	1.57	8.10	0.00	3.18	3.00	0.00	0.00

(e) Leukaemia

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	210.70	1.63	9.41	112.66	1.77	5.19	210.70	1.03	9.41
0.55	102.34	1.96	6.74	112.14	3.60	12.38	102.34	2.47	6.74
0.60	94.58	1.74	5.73	98.92	1.56	6.94	94.58	2.46	5.73
0.65	95.84	1.96	7.27	101.93	2.26	7.51	95.84	5.26	7.27
0.70	90.96	2.19	6.70	98.82	2.97	12.43	90.96	5.62	6.70
0.75	91.36	2.32	7.70	86.81	6.05	23.47	91.36	3.92	7.70
0.80	92.39	2.47	9.16	92.60	5.28	20.90	92.39	3.34	9.16
0.85	88.30	2.49	15.75	78.66	7.00	23.83	88.30	5.69	15.75
0.90	70.15	5.91	22.70	88.97	8.06	21.48	70.15	4.62	22.70
0.95	85.80	4.87	8.88	89.93	7.34	25.07	85.80	4.05	8.88

D.2 Simultaneous Rule Learning Performance

This section presents equivalent tables to those presented in the previous section, but for SRL induced rulebases. These results, and/or figures generated from them, are discussed in Section 7.2 in the main text.

Table D.7 gives the accuracy of SRL induced rulebases, using a specific `minInstPerRule` value for each dataset for different forms of the hypothesis language (parameter settings are provided in Table B.5(c) on page 222). Table D.8 provides the corresponding complexity details for the rulebases.

Table D.9 provides the accuracy of rulebases induced using different values for the `minInstPerRule` parameter, and for rules containing negated terms in the antecedent. Table D.10 provides the corresponding rulebase complexity statistics. Note that one column in each of the subtables in Table D.9 duplicates the results of the 'Negation' column in each of the corresponding subtables in Table D.7. The statistics are replicated for ease of reference.

Table D.7: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	88.87	0.94	7.57	63.73	3.85	15.37	88.01	2.15	7.83	87.68	2.27	6.64
0.55	88.32	0.41	7.49	66.53	3.78	13.29	90.07	1.07	6.04	88.69	1.99	6.08
0.60	88.11	0.18	6.26	67.28	4.46	11.84	89.70	1.82	6.56	90.37	2.51	5.82
0.65	90.06	0.27	5.04	64.28	5.10	12.12	88.61	1.63	6.84	89.24	1.48	5.81
0.70	89.53	0.28	6.21	64.16	3.31	13.33	88.15	1.80	6.26	89.36	2.34	6.93
0.75	92.00	0.57	5.88	65.74	3.47	12.26	90.12	1.74	6.59	90.07	1.16	7.22
0.80	92.00	0.57	5.20	67.93	3.55	11.98	90.03	1.67	7.25	90.52	1.53	7.23
0.85	92.67	0.00	4.73	67.10	3.30	12.15	91.87	0.91	7.16	93.07	1.30	6.04
0.90	92.11	0.00	5.49	66.35	3.97	15.77	90.85	2.10	7.09	89.92	1.66	6.31
0.95	90.41	0.00	7.18	68.65	3.16	9.67	88.31	2.04	7.91	89.98	2.19	8.52

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	91.33	0.00	7.73	56.33	3.49	14.04	86.27	0.90	10.37	91.87	1.36	9.32
0.55	91.33	0.00	7.73	54.47	3.80	12.65	88.80	0.53	10.47	89.07	1.23	9.08
0.60	92.00	0.00	6.13	51.93	3.42	12.28	88.13	0.61	9.40	92.87	1.26	7.37
0.65	92.00	0.00	6.13	52.53	2.86	12.07	90.53	1.25	8.04	94.93	0.34	6.58
0.70	95.33	0.00	4.50	54.60	3.04	15.32	92.07	0.49	9.11	95.13	0.55	5.81
0.75	86.67	0.00	12.17	52.53	5.34	15.94	75.67	0.35	7.22	95.93	0.21	4.65
0.80	88.67	0.00	11.35	47.53	4.88	18.50	75.20	0.53	7.34	92.87	0.63	5.38
0.85	96.00	0.00	5.62	58.40	6.40	19.70	75.13	0.45	7.27	93.33	0.83	6.45
0.90	82.67	0.00	10.98	54.20	5.99	13.90	74.40	0.34	7.19	72.87	0.45	7.31
0.95	82.67	0.00	10.98	54.47	1.96	14.96	63.33	0.00	11.86	72.40	0.56	6.71

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	49.76	2.44	12.98	35.24	3.29	12.51	56.12	3.23	10.93	55.08	2.28	11.86
0.55	53.66	1.23	11.22	31.60	4.08	10.86	56.75	1.87	10.87	55.43	2.52	11.37
0.60	36.41	0.70	14.28	26.18	2.01	13.79	54.70	2.33	10.54	54.73	1.53	10.94
0.65	36.39	0.34	9.04	26.31	5.75	13.15	51.32	2.36	11.45	56.48	1.93	12.24
0.70	32.03	0.67	4.92	21.52	3.23	14.40	55.02	2.63	12.57	55.76	2.33	11.13
0.75	12.14	0.00	6.27	9.20	2.24	8.83	53.52	2.55	11.45	52.81	2.13	12.50
0.80	9.30	0.00	7.06	6.54	1.41	4.70	56.65	1.84	13.46	55.22	1.55	13.48
0.85	8.78	0.00	4.73	12.71	2.41	9.60	56.37	2.27	11.89	57.56	1.63	12.97
0.90	30.65	0.24	9.95	24.08	2.67	10.29	52.37	0.81	10.17	56.80	1.79	13.71
0.95	15.85	0.31	8.39	16.33	2.34	10.05	50.72	0.95	9.44	50.68	2.10	8.84

Table D.7: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpIRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	80.97	0.00	29.08	35.20	6.08	30.41	77.22	1.08	9.90	82.12	0.75	7.30
0.55	80.61	0.77	29.04	34.21	9.66	27.97	75.69	0.59	9.95	84.61	0.34	6.82
0.60	80.97	0.00	29.08	35.00	9.07	29.74	76.60	1.29	10.58	81.61	0.60	8.92
0.65	80.97	0.00	29.08	34.91	7.40	30.84	76.54	1.36	10.55	81.80	0.28	8.66
0.70	30.07	0.00	39.03	23.36	7.58	25.99	75.58	0.00	8.43	85.25	0.00	8.78
0.75	30.07	0.00	39.03	20.28	4.18	26.07	75.45	0.28	8.44	85.25	0.00	8.78
0.80	30.07	0.00	39.03	23.65	5.24	28.32	76.93	0.00	4.97	85.17	0.25	8.88
0.85	37.14	0.00	40.31	22.93	4.62	27.71	79.02	0.00	5.92	79.13	0.39	5.69
0.90	68.32	0.00	12.77	31.18	6.02	27.56	82.78	0.00	6.11	79.31	0.00	5.58
0.95	31.13	0.00	40.37	21.55	3.52	24.98	82.78	0.00	6.11	82.78	0.00	6.11

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
threshold	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	94.17	0.00	7.66	81.26	2.76	17.63	90.69	1.63	16.44	90.68	0.82	15.89
0.55	91.07	0.00	12.56	80.11	4.66	16.61	91.62	0.79	15.95	91.35	1.05	16.38
0.60	88.74	0.53	10.14	63.63	7.59	21.44	93.02	1.15	11.74	92.01	1.34	13.35
0.65	86.31	1.51	9.64	66.57	6.19	19.05	90.85	1.25	15.09	90.62	0.45	15.93
0.70	78.39	0.00	13.58	58.86	5.18	21.12	91.68	1.87	12.73	93.19	1.60	11.47
0.75	72.26	0.00	12.87	63.67	6.68	20.67	92.36	1.71	11.79	92.89	1.52	11.72
0.80	69.95	2.15	19.53	58.44	6.61	21.01	92.08	2.25	11.46	92.29	2.25	12.01
0.85	65.36	0.88	15.53	59.05	5.71	22.14	92.23	1.48	11.09	92.68	1.60	9.53
0.90	62.38	0.00	13.86	60.71	7.63	21.01	92.35	1.65	8.50	91.98	1.52	9.54
0.95	58.90	0.66	10.56	64.03	5.91	19.93	92.85	1.08	8.48	92.81	1.31	8.18

Table D.8: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different degrees of richness of the hypothesis language

(a) Wine

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	7.13	0.09	0.77	9.40	0.43	1.77	22.77	0.57	1.79	22.71	0.55	2.12
0.55	7.12	0.13	0.77	9.22	0.57	1.43	22.14	0.45	1.33	22.17	0.62	1.68
0.60	7.69	0.17	0.71	8.73	0.38	1.32	20.51	0.46	1.91	20.62	0.71	2.18
0.65	6.47	0.07	0.67	8.55	0.28	1.14	20.56	0.81	2.25	19.96	0.97	2.32
0.70	6.00	0.00	0.47	8.16	0.28	1.06	19.33	0.48	1.62	18.98	0.60	1.70
0.75	5.00	0.00	0.00	8.21	0.27	0.89	18.51	0.48	1.46	17.63	0.32	1.60
0.80	4.90	0.00	0.32	7.88	0.15	0.86	17.50	0.47	1.71	17.23	0.45	1.13
0.85	4.70	0.00	0.48	7.86	0.21	0.95	17.35	0.32	1.48	16.43	0.36	1.22
0.90	4.50	0.00	0.53	7.67	0.23	0.74	16.36	0.43	1.28	15.45	0.36	1.73
0.95	4.00	0.00	0.00	7.57	0.18	0.60	15.91	0.38	1.13	15.37	0.34	0.96

(b) Iris

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	8.00	0.00	0.00	11.07	0.43	1.08	11.50	0.09	0.52	10.56	0.14	0.68
0.55	8.00	0.00	0.00	10.99	0.26	0.99	11.00	0.15	0.65	11.00	0.13	0.66
0.60	6.70	0.00	0.48	11.09	0.23	0.90	10.86	0.08	0.62	10.67	0.08	0.53
0.65	6.70	0.00	0.48	11.23	0.32	1.02	10.18	0.09	0.39	10.00	0.00	0.00
0.70	6.00	0.00	0.00	10.66	0.17	0.57	10.10	0.05	0.30	10.00	0.00	0.00
0.75	6.00	0.00	0.00	10.74	0.14	0.45	9.90	0.00	0.32	9.93	0.08	0.18
0.80	5.00	0.00	0.00	8.91	0.12	0.34	9.90	0.00	0.32	9.70	0.00	0.48
0.85	5.00	0.00	0.00	8.85	0.16	0.32	9.90	0.00	0.32	9.58	0.06	0.72
0.90	4.00	0.00	0.00	6.80	0.15	0.42	9.90	0.00	0.32	9.00	0.00	0.42
0.95	4.00	0.00	0.00	6.85	0.12	0.40	9.60	0.00	0.52	9.04	0.08	0.50

(c) Glass

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	27.32	0.42	1.58	27.07	1.22	3.34	48.98	0.58	1.77	49.95	0.49	1.76
0.55	23.88	0.27	1.35	25.29	0.87	2.57	48.37	0.49	1.57	49.04	0.30	1.48
0.60	19.93	0.34	1.79	21.02	0.94	2.60	44.71	0.43	2.38	45.93	0.65	1.89
0.65	13.35	0.08	0.94	16.78	0.45	1.94	44.43	0.62	1.92	43.44	0.64	2.33
0.70	13.96	0.05	0.13	15.58	0.44	1.65	42.25	0.52	2.11	41.96	0.30	2.28
0.75	9.22	0.15	1.52	14.77	0.58	2.21	39.86	0.60	2.06	40.66	0.55	2.50
0.80	9.60	0.00	0.97	13.25	0.46	1.94	39.18	0.74	2.25	38.58	0.65	2.42
0.85	9.90	0.00	1.10	13.17	0.33	1.85	36.82	0.30	1.98	36.19	0.52	1.91
0.90	8.11	0.03	0.73	12.79	0.18	1.30	34.04	0.76	2.72	33.00	0.74	2.21
0.95	6.82	0.04	0.65	11.57	0.51	1.16	30.74	0.47	2.20	30.33	0.48	1.90

Table D.8: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different degrees of richness of the hypothesis language (cont.)

(d) Water Treatment

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	3.10	0.00	0.32	2.79	0.41	1.26	4.34	0.18	0.65	5.05	0.05	0.47
0.55	3.12	0.06	0.35	2.88	0.39	1.28	4.40	0.11	0.91	4.42	0.13	0.52
0.60	3.10	0.00	0.32	2.92	0.43	1.32	4.14	0.13	0.32	4.22	0.06	0.65
0.65	3.10	0.00	0.32	2.77	0.46	1.26	4.13	0.09	0.30	4.20	0.00	0.63
0.70	3.80	0.00	0.63	2.66	0.24	1.29	4.10	0.00	0.32	4.00	0.00	0.00
0.75	3.80	0.00	0.63	2.75	0.12	1.29	4.12	0.04	0.34	4.00	0.00	0.00
0.80	3.80	0.00	0.63	2.54	0.16	1.19	4.00	0.00	0.00	4.01	0.03	0.03
0.85	3.50	0.00	0.71	2.58	0.13	1.24	3.70	0.00	0.48	4.02	0.04	0.06
0.90	2.80	0.00	0.63	2.74	0.31	1.20	3.00	0.00	0.00	4.00	0.00	0.00
0.95	2.10	0.00	0.32	2.18	0.16	1.99	3.00	0.00	0.00	3.00	0.00	0.00

(e) Leukaemia

	Simple			Disjunction			Negation			Hedges		
construction threshold	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	4.10	0.00	0.57	10.70	0.21	1.45	41.80	0.28	2.41	41.49	0.51	1.98
0.55	3.40	0.00	0.70	9.17	0.43	1.14	35.86	0.24	2.45	36.42	0.40	2.15
0.60	2.49	0.03	0.53	4.91	0.07	1.06	32.16	0.43	2.49	32.06	0.61	2.66
0.65	2.00	0.00	0.00	3.86	0.21	0.25	30.98	0.36	2.67	29.79	0.33	2.60
0.70	2.00	0.00	0.00	3.92	0.09	0.32	27.63	0.22	2.16	26.68	0.59	2.26
0.75	2.00	0.00	0.00	3.87	0.12	0.28	27.00	0.32	1.68	26.81	0.25	1.55
0.80	2.00	0.00	0.00	3.90	0.12	0.38	25.67	0.16	2.34	25.27	0.39	2.30
0.85	2.00	0.00	0.00	3.91	0.14	0.28	22.83	0.56	2.36	22.68	0.39	2.43
0.90	2.00	0.00	0.00	3.91	0.18	0.50	22.47	0.33	2.30	22.36	0.42	2.20
0.95	2.00	0.00	0.00	4.01	0.29	0.71	19.51	0.40	1.59	19.67	0.24	1.54

Table D.9: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	88.77	2.16	7.65	88.01	2.15	7.83	90.12	2.13	7.63
0.55	88.74	1.62	6.53	90.07	1.07	6.04	90.24	1.96	7.25
0.60	89.43	2.07	6.65	89.70	1.82	6.56	90.36	1.90	5.74
0.65	88.79	1.36	6.51	88.61	1.63	6.84	90.84	2.18	6.12
0.70	90.84	2.34	6.97	88.15	1.80	6.26	91.53	1.94	5.53
0.75	89.54	1.50	6.88	90.12	1.74	6.59	89.22	1.19	8.39
0.80	90.10	1.77	6.82	90.03	1.67	7.25	91.24	1.09	7.68
0.85	89.88	1.05	6.39	91.87	0.91	7.16	92.94	1.93	7.10
0.90	90.08	1.89	6.80	90.85	2.10	7.09	90.08	1.18	5.45
0.95	88.81	1.49	7.77	88.31	2.04	7.90	91.35	0.74	3.75

(b) Iris

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	87.67	0.57	12.40	85.20	1.91	11.90	86.27	0.90	10.37
0.55	87.80	0.95	12.48	85.73	1.41	11.49	88.80	0.53	10.47
0.60	85.73	1.61	11.27	91.27	0.73	9.16	88.13	0.61	9.40
0.65	86.67	1.51	10.93	88.07	0.97	10.51	90.53	1.25	8.04
0.70	89.60	0.90	11.69	87.33	0.44	9.01	92.07	0.49	9.11
0.75	91.67	0.96	9.33	87.40	0.73	8.31	75.67	0.35	7.22
0.80	90.47	0.95	7.33	90.67	0.44	9.92	75.20	0.53	7.34
0.85	91.60	0.72	7.46	90.80	0.61	9.17	75.13	0.45	7.27
0.90	81.67	0.35	9.12	76.87	0.55	12.81	74.40	0.34	7.19
0.95	78.47	0.77	13.74	72.67	0.44	9.90	63.33	0.00	11.86

(c) Glass

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-	ACC %	SEM +/-	STD +/-
0.50	56.12	3.23	10.93	57.51	1.80	10.80	58.89	1.78	10.91
0.55	56.75	1.87	10.87	57.43	1.85	9.20	59.63	2.10	11.35
0.60	54.70	2.33	10.54	57.80	1.66	10.00	61.81	1.28	12.65
0.65	51.32	2.36	11.45	54.28	2.70	12.33	50.92	1.77	12.56
0.70	55.02	2.63	12.57	52.73	1.92	7.88	52.47	1.96	12.72
0.75	53.52	2.55	11.45	51.04	1.60	8.04	52.56	0.43	13.89
0.80	56.65	1.84	13.46	51.68	2.34	9.51	52.03	0.62	15.03
0.85	56.37	2.27	11.89	53.47	1.04	14.92	48.24	0.66	14.02
0.90	52.37	0.81	10.17	43.83	0.82	14.56	33.92	0.49	8.64
0.95	50.72	0.95	9.44	43.02	1.32	10.24	34.58	0.20	7.23

Table D.9: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction`	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
`threshold`	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	77.22	1.08	9.90	78.96	0.25	7.24	61.22	0.00	18.28
0.55	75.69	0.59	9.95	78.96	0.25	7.24	62.78	0.00	16.60
0.60	76.60	1.29	10.58	77.93	0.41	6.71	68.65	0.00	18.12
0.65	76.54	1.36	10.55	78.09	0.33	6.70	68.65	0.00	18.12
0.70	75.58	0.00	8.43	76.93	0.00	4.97	67.60	0.00	19.25
0.75	75.45	0.28	8.44	81.45	0.00	7.09	67.60	0.00	19.25
0.80	76.93	0.00	4.97	82.78	0.00	6.11	67.60	0.00	19.25
0.85	79.02	0.00	5.92	82.78	0.00	6.11	67.60	0.00	19.25
0.90	82.78	0.00	6.11	82.78	0.00	6.11	67.60	0.00	19.25
0.95	82.78	0.00	6.11	82.78	0.00	6.11	67.60	0.00	19.25

(e) Leukaemia

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction`	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
`threshold`	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
0.50	89.45	2.02	16.96	91.25	1.18	16.42	90.69	1.63	16.44
0.55	89.26	1.87	17.71	89.76	1.51	16.30	91.62	0.79	15.95
0.60	91.43	2.16	16.18	91.68	2.32	13.80	93.02	1.15	11.74
0.65	91.68	1.87	12.73	92.74	1.75	13.48	90.85	1.25	15.09
0.70	91.42	2.15	16.17	92.55	1.70	14.17	91.68	1.87	12.73
0.75	92.44	1.57	13.62	92.95	1.49	9.83	92.36	1.71	11.79
0.80	92.11	2.25	12.89	91.73	1.21	13.63	92.08	2.25	11.46
0.85	93.41	1.76	10.75	93.28	2.06	12.26	92.23	1.48	11.09
0.90	91.89	1.79	14.41	94.29	1.29	8.66	92.35	1.65	8.50
0.95	91.08	2.12	14.68	92.37	1.68	12.54	92.85	1.08	8.48

Table D.10: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms.

(a) Wine

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	25.02	0.58	1.79	22.77	0.57	1.79	20.40	0.66	1.87
0.55	23.96	0.48	1.73	22.14	0.45	1.33	18.76	0.63	1.87
0.60	23.21	0.53	1.93	20.51	0.46	1.91	19.02	0.72	1.92
0.65	22.58	0.64	1.95	20.56	0.81	2.25	18.53	0.42	1.69
0.70	21.86	0.49	1.39	19.33	0.48	1.62	17.21	0.22	1.14
0.75	20.33	0.44	1.71	18.51	0.48	1.46	15.36	0.40	1.73
0.80	19.20	0.19	1.72	17.50	0.47	1.71	15.02	0.21	1.49
0.85	18.26	0.45	1.51	17.35	0.32	1.48	13.83	0.28	1.39
0.90	17.58	0.36	1.50	16.36	0.43	1.28	14.97	0.21	1.35
0.95	18.10	0.48	1.97	15.91	0.38	1.13	14.83	0.17	0.90

(b) Iris

	`minInstPerRule=40%`			`minInstPerRule=50%`			`minInstPerRule=60%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	12.00	0.00	0.00	11.90	0.05	0.30	11.50	0.09	0.52
0.55	12.00	0.00	0.00	11.92	0.04	0.25	11.00	0.15	0.65
0.60	11.97	0.05	0.09	10.69	0.11	0.49	10.86	0.08	0.62
0.65	11.85	0.08	0.33	11.10	0.07	0.58	10.18	0.09	0.39
0.70	11.51	0.07	0.73	10.74	0.05	0.94	10.10	0.05	0.30
0.75	10.63	0.11	0.57	9.95	0.13	0.88	9.90	0.00	0.32
0.80	10.09	0.10	0.66	9.50	0.00	0.53	9.90	0.00	0.32
0.85	10.00	0.07	0.27	9.50	0.00	0.53	9.90	0.00	0.32
0.90	9.53	0.12	0.57	8.88	0.09	0.43	9.90	0.00	0.32
0.95	8.96	0.12	0.42	9.12	0.04	0.34	9.60	0.00	0.52

(c) Glass

	`minInstPerRule=50%`			`minInstPerRule=60%`			`minInstPerRule=70%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	48.98	0.58	1.77	47.12	0.41	1.53	43.39	0.45	1.75
0.55	48.37	0.49	1.57	45.30	0.68	2.17	42.07	0.60	1.79
0.60	44.71	0.43	2.38	42.79	0.39	1.68	39.32	0.51	1.88
0.65	44.43	0.62	1.92	40.33	0.41	1.94	36.96	0.27	1.70
0.70	42.25	0.52	2.11	36.66	0.61	2.15	33.01	0.40	1.49
0.75	39.86	0.60	2.06	34.90	0.54	2.08	31.72	0.21	1.65
0.80	39.18	0.74	2.25	34.10	0.43	2.25	30.84	0.23	2.10
0.85	36.82	0.30	1.98	31.36	0.37	1.66	28.30	0.19	1.81
0.90	34.04	0.76	2.72	29.86	0.46	2.10	25.47	0.28	1.05
0.95	30.74	0.47	2.20	27.02	0.40	1.50	24.46	0.22	1.18

Table D.10: Number of terms, standard error of the mean (SEM) and standard deviation (STD) of *FRANTIC-simpSRL* rulebases induced using different values for `constructionThreshold` and `minInstPerRule`. All rules are induced with possibility of containing negated terms (cont.)

(d) Water Treatment

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	4.34	0.18	0.65	4.12	0.06	0.35	3.60	0.00	0.52
0.55	4.40	0.11	0.91	4.12	0.06	0.35	3.30	0.00	0.95
0.60	4.14	0.13	0.32	4.28	0.10	0.72	2.60	0.00	0.52
0.65	4.13	0.09	0.30	4.24	0.08	0.67	2.60	0.00	0.52
0.70	4.10	0.00	0.32	4.00	0.00	0.00	2.50	0.00	0.53
0.75	4.12	0.04	0.34	3.30	0.00	0.48	2.50	0.00	0.53
0.80	4.00	0.00	0.00	3.00	0.00	0.00	2.50	0.00	0.53
0.85	3.70	0.00	0.48	3.00	0.00	0.00	2.50	0.00	0.53
0.90	3.00	0.00	0.00	3.00	0.00	0.00	2.50	0.00	0.53
0.95	3.00	0.00	0.00	3.00	0.00	0.00	2.50	0.00	0.53

(e) Leukaemia

	`minInstPerRule=70%`			`minInstPerRule=80%`			`minInstPerRule=90%`		
`construction threshold`	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
0.50	51.69	1.21	3.47	46.39	0.57	2.76	41.80	0.28	2.41
0.55	47.39	0.88	3.43	39.48	0.59	3.37	35.86	0.24	2.45
0.60	45.32	0.81	2.52	38.06	1.05	3.37	32.16	0.43	2.49
0.65	41.98	0.82	2.96	36.38	0.65	3.22	30.98	0.36	2.67
0.70	39.52	0.48	2.73	33.55	0.69	2.84	27.63	0.22	2.16
0.75	37.46	0.48	2.74	30.75	0.48	2.81	27.00	0.32	1.68
0.80	34.51	0.44	2.58	29.49	0.57	2.63	25.67	0.16	2.34
0.85	31.78	0.42	2.47	26.71	0.49	3.16	22.83	0.56	2.36
0.90	30.30	0.41	3.21	25.23	0.74	3.09	22.47	0.33	2.30
0.95	28.49	0.51	2.60	22.27	0.33	2.60	19.51	0.40	1.59

Appendix E

Supplementary Results to Chapter 8

This appendix presents detailed results in table form that support various graphs presented in the main text of this thesis. It also presents the parameter settings for the algorithms that are used to obtain these results.

Note that in all the following Tables, *NEFC* denotes results for the *NEFCLASS* system, and *FRAN* denotes results produced by F*RANTIC-simpSRL*.

E.1 Model Accuracy

Table E.1 provides the classification accuracy (ACC), standard error of the mean (SEM), and standard deviation (STD) achieved by the models produced by each algorithm on each dataset. The STD is the standard deviation of the ten models produced by a ten-fold cross-validation, and the SEM is the standard deviation divided by the square root of the sample size (i.e. by $\sqrt{10}$). Note that in the case of *FRANTIC* 10 ten-fold cross-validations have been carried out – the STD is therefore the average of the 10 standard deviations obtained from each of the 10 cross-validations, and the SEM is standard deviation of the 10 average accuracies from each of the 10 cross-validations. The results in this table are presented in graphical form (Fig 8.1) and discussed in the main text of this thesis in Section 8.1.

Table B.4 on page 221 lists the parameter settings that lead to the *FRANTIC* rulebases whose results are presented in Chapter 8. As mentioned in Section 5.1 on page 72, and detailed in Section B.2 on page 215, several of the algorithms against which *FRANTIC* is compared are run with different parameter settings, and the best results obtained for each are the ones used to compare the algorithms against each other. Table E.2 indicates the settings for the different algorithms that are used to obtain the results for each dataset:

Table E.1: Classification accuracy (ACC), standard error of the mean (SEM) and standard deviation (STD) of models induced by all algorithms

	Wine			Iris			Glass			WT			Leuk		
	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD	ACC	SEM	STD
	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-	%	+/-	+/-
C4.5	94.38	1.17	3.70	95.33	1.42	4.50	67.39	2.56	8.10	98.43	0.44	1.40	87.62	4.36	13.80
NB	98.36	0.82	2.60	96.67	1.11	3.50	49.84	2.25	7.10	98.68	0.44	1.40	98.33	1.68	5.30
SMO	94.96	0.98	3.10	97.33	1.49	4.70	63.83	2.85	9.00	98.69	0.44	1.40	92.32	3.59	11.35
QSBA	92.20	1.74	5.50	92.00	2.21	7.00	50.84	2.21	7.00	77.69	3.60	11.40	96.90	2.06	6.50
FSBA	88.26	1.71	5.40	89.33	3.16	10.00	43.10	2.50	7.90	54.32	4.64	14.68	82.80	4.99	15.79
FID3.4	84.63	4.74	15.00	94.67	1.93	6.10	59.53	3.10	9.80	98.43	0.44	1.40	94.58	2.25	7.10
NEFC	90.48	2.28	7.20	95.33	1.42	4.50	28.35	3.64	11.50	97.08	0.95	3.00	91.25	3.95	12.50
FRAN	93.07	1.30	6.04	96.00	0.00	5.62	61.81	1.28	12.65	77.22	1.08	9.90	94.29	1.29	8.66

Table E.2: Algorithm settings leading to results in Table E.1

	Wine	Iris	Glass	WT	Leuk
C4.5	+/-pruning	+/-pruning	red.error	any	red.error
NB	norm.dist.	kernel est.	kernel est.	kernel est.	kernel est.
SMO	poly.kern	poly.kern	poly.kern	poly.kern	poly.kern
FSBA	$\alpha = 0.9$, $\beta = 0.5 - 1.0$	$\alpha = 0.7$, $\beta = 0.5 - 1.0$	$\alpha = 0.85$, $\beta = 0.5 - 1.0$	$\alpha = 1$, $\beta = 1.0$	$\alpha = 0.9$, $\beta = 0.5 - 1.0$
FID3.4	pruning=1	pruning=0.5/1	pruning=1	any	pruning=0/0.5

- *C4.5* – '+/-pruning' indicates that the same results are obtained when no tree pruning is used, as when subtree replacement and subtree raising pruning is applied; 'red.error' indicates results obtained with reduced-error pruning; 'any' indicates that the same results are obtained irrespective of whether pruning is used or not.
- *NB* – 'norm.dist' indicates that a normal distribution assumption is made when dealing with continuous attributes, while 'kernel est.' indicates that such an assumption is not made and a kernel estimator is used instead.
- *SMO* – 'poly.kern' indicates results obtained using a polynomial kernel function to determine the maximum margin hyperplane (instead of a radial basis function).
- *FSBA* – the settings for the algorithm's two parameters, α and β.
- *FID3.4* – 'pruning=xx' indicates the setting level of the coefficient for tree pruning; 'any' indicates that the same accuracy results are obtained irrespective of the value of the pruning parameter, and even when no pruning is applied.

Table E.3: Number of terms in a rulebase or decision tree, and associated standard error of the mean (SEM) and standard deviation (STD)

	Wine			Iris			Glass			WT			Leuk		
	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-	Terms	SEM +/-	STD +/-
C4.5	13.70	1.32	4.16	12.50	0.77	2.42	45.10	5.54	17.51	13.50	3.55	11.22	3.80	0.49	1.55
QSBA	117.00	0.00	0.00	36.00	0.00	0.00	162.00	0.00	0.00	26.00	0.00	0.00	300.00	0.00	0.00
FSBA	21.20	0.25	0.79	12.00	0.00	0.00	36.10	0.38	1.20	2.10	0.10	0.32	59.50	0.37	1.18
FID3.4	29.20	6.25	19.76	8.40	1.06	3.34	74.40	7.86	24.84	0.00	0.00	0.00	3.80	0.49	1.55
NEFC	439.10	51.77	163.72	3.00	0.00	0.00	77.70	18.02	56.97	5.50	0.60	1.90	168.50	33.04	104.47
FRAN	16.43	0.36	1.22	5.00	0.00	0.00	39.32	0.51	1.88	4.34	0.18	0.65	25.23	0.74	3.09

Table E.4: Number of rules in a rulebase or decision tree, and associated standard error of the mean (SEM) and standard deviation (STD)

	Wine			Iris			Glass			WT			Leuk		
	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-	Rules	SEM +/-	STD +/-
C4.5	5.40	0.31	0.97	4.70	0.15	0.48	10.30	0.80	2.54	4.60	0.83	2.63	2.60	0.16	0.52
QSBA	3.00	0.00	0.00	3.00	0.00	0.00	6.00	0.00	0.00	2.00	0.00	0.00	2.00	0.00	0.00
FSBA	3.00	0.00	0.00	3.00	0.00	0.00	6.00	0.00	0.00	2.00	0.00	0.00	2.00	0.00	0.00
FID3.4	8.00	0.63	2.00	4.80	0.36	1.14	15.20	1.26	3.97	1.00	0.00	0.00	2.60	0.16	0.52
NEFC	61.20	3.48	10.99	3.00	0.00	0.00	16.00	2.11	6.68	2.80	0.13	0.42	25.90	4.26	13.48
FRAN	3.00	0.00	0.00	3.00	0.00	0.00	6.00	0.00	0.00	2.00	0.00	0.00	2.00	0.00	0.00

E.2 Model Comprehensibility

Table E.3 provides the corresponding model complexity details for the models whose accuracy results are presented in Table E.1; the average number of terms in a rulebase or decision tree are presented, and the associated standard error of the mean (SEM) and standard deviation (STD). The SEM and STD are calculated in the same manner as in the preceding Section E.1. The results in Table E.3 are also presented in graphical form (Fig. 8.4) and discussed in the main text of this thesis in Section 8.2 on page 190.

The average number of rules in a rulebase or decision tree, and the average number of terms in a rule are additional measures for estimating the complexity of an induced model. These may be useful alternatives when comparing two algorithms that obtain similar measures with regards to the average number of terms in a rulebase. Table E.4 therefore shows the average number of rules, and Table E.5 the average number of terms in a rule for models induced by all algorithms utilised in this work. The standard deviation is shown in brackets ($+/-$).

Table E.5: Number of terms per rule in a rulebase or decision tree, and associated standard error of the mean (SEM) and standard deviation (STD)

	Wine			Iris			Glass			WT			Leuk		
	Terms/ rule	SEM +/-	STD +/-	Terms/ rule	SEM +/-	STD +/-	Terms/ rule	SEM +/-	STD +/-	Terms/ rule	SEM +/-	STD +/-	Terms/ rule	SEM +/-	STD +/-
C4.5	2.50	0.08	0.24	2.64	0.09	0.27	4.29	0.22	0.68	2.21	0.45	1.42	1.40	0.11	0.34
QSBA	39.00	0.00	0.00	12.00	0.00	0.00	27.00	0.00	0.00	13.00	0.00	0.00	150.00	0.00	0.00
FSBA	7.07	0.08	0.26	4.00	0.00	0.00	6.02	0.06	0.20	1.10	0.10	0.32	29.75	0.19	0.59
FID3.4	3.42	0.52	1.65	1.70	0.08	0.26	4.83	0.21	0.67	0.00	0.00	0.00	1.40	0.11	0.34
NEFC	7.04	0.47	1.50	1.00	0.00	0.00	4.51	0.38	1.20	1.90	0.16	0.50	5.73	0.68	2.15
FRAN	5.48	0.12	0.10	1.67	0.00	0.00	6.55	0.08	0.10	2.17	0.09	0.00	12.62	0.37	0.20

E.3 Example Models

This section provides examples of the models induced by several of the algorithms whose results are presented in Chapter 8. As previously described, *SVM* and *NB* models are not particularly informative, and so they are excluded from this section.

Figures E.1–E.4 show fuzzy rulebases describing the Water Treatment dataset induced by *FRANTIC*, *QSBA*, *FSBA* and *NEFCLASS*. As discussed in Section 8.1.1, it is a highly imbalanced dataset that presents a challenge to most of the learning algorithms utilised here.

The decision tree algorithms – *FID3.4* and *C4.5* – are two of the better performers on the six-class Glass dataset. Figures E.5 and E.6 depict a decision tree induced by each algorithm to describe this dataset.

R1	IF COND-E is NOT High AND SED-S is Low THEN OUTCOME is NORMAL
R2	IF PH-D is NOT High AND DBO-D is NOT Normal THEN OUTCOME is FAULTY

Figure E.1: *FRANTIC* rulebase for the Water Treatment dataset (74% accuracy, *TP_Rate*=1, *FP_Rate*=0.27)

R1	IF DBO-D is NOT Normal THEN OUTCOME is FAULTY
R2	IF *Membership*(FAULTY)$<$0.5 THEN OUTCOME is NORMAL

Figure E.2: *FSBA* rulebase for the Water Treatment dataset (32% accuracy, *TP_Rate*=1, *FP_Rate*=0.7)

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

Figure E.3: *QSBA* rulebase for the Water Treatment dataset (95% accuracy, *TP_Rate*=1, *FP_Rate*=0.05)

R1	IF SED-S is Low AND RD-SED-G is High THEN OUTCOME is NORMAL
R2	IF SED-S is Low AND RD-SED-G is Low THEN OUTCOME is NORMAL
R3	IF SED-S is High AND RD-SED-G is Low THEN OUTCOME is FAULTY

Figure E.4: *NEFCLASS* rulebase for the Water Treatment dataset (97% accuracy, *TP_Rate*=0, *FP_Rate*=0)

```
Mg <= 2.68
|   Na <= 14.04
|   |   Mg <= 2.24
|   |   |   RI <= 1.5241: 4 (12.0/5.0)
|   |   |   RI > 1.5241: 2 (7.0)
|   |   Mg > 2.24: 5 (3.0/1.0)
|   Na > 14.04
|   |   Ba <= 0.15: 5 (5.0/1.0)
|   |   Ba > 0.15: 6 (12.0)
Mg > 2.68
|   RI <= 1.51711
|   |   Ca <= 8.28
|   |   |   Ba <= 0
|   |   |   |   Fe <= 0.22: 2 (19.0)
|   |   |   |   Fe > 0.22: 1 (3.0/1.0)
|   |   |   Ba > 0: 2 (2.0/1.0)
|   |   Ca > 8.28: 3 (5.0/2.0)
|   RI > 1.51711
|   |   Mg <= 3.82: 1 (53.0/14.0)
|   |   Mg > 3.82
|   |   |   RI <= 1.51994: 2 (7.0)
|   |   |   RI > 1.51994: 1 (2.0/1.0)

Number of Leaves  :     12
```

Figure E.5: Pruned *C4.5* decision tree for the Glass dataset (80% accuracy)

```
[] : IN=0.84 PN=193.00 : bldng=63.00 bldng=69.00 veh-f=15.00 conta=12.00 table=8.00 headl=26.00
   [RI=RI1][Ba=Ba1] : IN=0.75 PN=166.78 : bldng=62.00 bldng=67.69 veh-f=15.00 conta=10.78 table=8.00 headl=3.31
      [RI=RI1][Ba=Ba1][K=K1] : IN=0.58 PN=13.00 : bldng=1.00 bldng=3.00 veh-f=1.00 conta=0.00 table=8.00 headl=0.00
      [RI=RI1][Ba=Ba1][K=K2] : IN=0.67 PN=151.78 : bldng=61.00 bldng=64.69 veh-f=14.00 conta=8.78 table=0.00 headl=3.31
         [RI=RI1][Ba=Ba1][K=K2][Mg=Mg1] : IN=0.51 PN=15.09 : bldng=0.00 bldng=6.00 veh-f=0.00 conta=7.78 table=0.00 headl=1.31
            [RI=RI1][Ba=Ba1][K=K2][Mg=Mg1][Si=Si2][Fe=Fe1] : IN=0.48 PN=9.07 : bldng=0.00 bldng=2.07 veh-f=0.00 conta=6.00 table=0.00 headl=1.00
            [RI=RI1][Ba=Ba1][K=K2][Mg=Mg1][Si=Si2][Fe=Fe3] : IN=-0.00 PN=0.93 : bldng=0.00 bldng=0.93 veh-f=0.00 conta=0.00 table=0.00 headl=0.00
         [RI=RI1][Ba=Ba1][K=K2][Mg=Mg2] : IN=-0.00 PN=1.00 : bldng=0.00 bldng=0.00 veh-f=0.00 conta=1.00 table=0.00 headl=0.00
         [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3] : IN=0.57 PN=135.69 : bldng=61.00 bldng=58.69 veh-f=14.00 conta=0.00 table=0.00 headl=2.00
            [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3][Si=Si1] : IN=0.38 PN=1.23 : bldng=0.00 bldng=0.56 veh-f=0.00 conta=0.00 table=0.00 headl=0.67
            [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3][Si=Si2] : IN=0.56 PN=134.46 : bldng=61.00 bldng=58.13 veh-f=14.00 conta=0.00 table=0.00 headl=1.33
               [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3][Si=Si2][Al=Al3][Ca=Ca1] : IN=-0.00 PN=1.00 : bldng=0.00 bldng=1.00 veh-f=0.00 conta=0.00 table=0.00 headl=0.00
               [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3][Si=Si2][Al=Al3][Ca=Ca2] : IN=0.29 PN=25.49 : bldng=4.82 bldng=20.53 veh-f=0.14 conta=0.00 table=0.00 headl=0.00
               [RI=RI1][Ba=Ba1][K=K2][Mg=Mg3][Si=Si2][Al=Al3][Ca=Ca3] : IN=0.57 PN=107.97 : bldng=56.18 bldng=36.61 veh-f=13.86 conta=0.00 table=0.00 headl=1.33
      [RI=RI1][Ba=Ba1][K=K3] : IN=-0.00 PN=2.00 : bldng=0.00 bldng=0.00 veh-f=0.00 conta=2.00 table=0.00 headl=0.00
   [RI=RI1][Ba=Ba2] : IN=0.30 PN=26.22 : bldng=1.00 bldng=1.31 veh-f=0.00 conta=1.22 table=0.00 headl=22.69

Number of Leaves  :    10
```

Figure E.6: Pruned *FID3.4* decision tree for the Glass dataset (76% accuracy)

Bibliography

[1] Cancer Program Data Sets. The Broad Institute, Cambridge MA, USA (2006). http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi.

[2] Abraham, A., Grosan, C. and Ramos, V. *Swarm Intelligence in Data Mining*. Studies in Computational Intelligence. Springer-Verlag (2006).

[3] Abraham, A. and Ramos, V. Web Usage Mining using Artificial Ant Colony Clustering and Genetic Programming. In: *Proceedings of the IEEE Congress on Evolutionary Computation*, 1384–1391 (2003).

[4] Al-Ani, A. Feature Subset Selection using Ant Colony Optimization. *International Journal of Computational Intelligence*, 2:53–58 (2005).

[5] Alcala, R., Cordón, O., Herrera, F. and Zwir, I. An Iterative Learning Methodology to Design Hierarchical Systems of Linguistic Rules for Linguistic Modeling. In: *Accuracy Improvements in Linguistic Fuzzy Modeling*, 277–301. Springer-Verlag (2003).

[6] Andrews, R., Diederich, J. and Tickle, A. B. A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*, 8(6):373–389 (1995).

[7] Asuncion, A. and Newman, D. J. UCI Machine Learning Repository. Department of Information and Computer Science, University of California, Irvine CA, USA (1998). http://www.ics.uci.edu/~mlearn/MLRepository.html.

[8] Backer, M. D., Haesen, R., Martens, D. and Baesens, B. A Stigmergy Based Approach to Data Mining. In: *Lecture Notes in Computer Science 3809*, 975–978. Springer (2005).

[9] Baeck, T. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press (1996).

[10] Bardossy, L. and Duckstein, L. *Fuzzy Rule-Based Modeling with Application to Geophysical, Biological and Engineering Systems*. CRC Press (1995).

[11] Bellman, R. *Dynamic Programming*. Princeton University Press (1957).

[12] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press (1981).

[13] Birattari, M., Caro, G. D. and Dorigo, M. For a Formal Foundation of the Ant Programming Approach to Combinatorial Optimization. Part I: The Problem, the Representation, and the General Solution Strategy. Tech. Rep. TR-H-301, ATR Human Information Processing Research Laboratories, Kyoto, Japan (2000).

[14] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press (1995).

[15] Blum, C. Beam-ACO – Hybridizing Ant Colony Optimization with Beam Search: An Application to Open Shop Scheduling. *Computers and Operations Research* (Article in Press).

[16] Boltyanskii, V. *Optimal Control of Discrete Systems*. John Wiley & Sons (1978).

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

[18] Bonabeau, E., Sobkowski, A., Theraulaz, G. and Deneubourg, J.-L. Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects. In: *Bio-Computation and Emergent Computing*, 36–45. World Scientific (1997).

[19] Bondalapati, K. and Prasanna, V. K. Reconfigurable Meshes: Theory and Practice. In: *Proceedings of the Reconfigurable Architectures Workshop (RAW'97)* (1997).

[20] Braga-Neto, U. M. and Dougherty, E. R. Is Cross-Validation Valid for Small-Sample Microarray Classification? *Bioinformatics*, 20(3):374–380 (2004).

[21] Breiman, L. Bagging Predictors. *Machine Learning*, 24(2):123–140 (1996).

[22] Buchanan, B. G. and Wilkins, D. C. *Readings in Knowledge Acquisition and Learning: Automating the construction and improvement of expert systems*. Morgan Kaufmann Publishers (1993).

[23] Bullnheimer, B., Kotsis, G. and Strauss, C. Parallelization Strategies for the Ant System. In: *High Performance Algorithms and Software in Nonlinear Optimization*, 87–100. Springer (1998).

[24] Buntine, W. and Niblett, T. A further Comparison of Splitting Rules for Decision-Tree Induction. *Machine Learning*, 8:75–85 (1989).

[25] Campos, M., Bonabeau, E., Theraulaz, G. and Deneubourg, J.-L. Dynamic Scheduling and Division of Labor in Social Insects. *Adaptive Behavior*, 8(2):83–95 (2000).

[26] Carmona, P. and Castro, J. L. Using Ant Colony Optimization for Learning Maximal Structure Fuzzy Rules. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 702–707 (2005).

[27] Casillas, J., Cordón, O. and Herrera, F. Improving the Wang and Mendel's Fuzzy Rule Learning Method by Inducing Cooperation among Rules. In: *Proceedings of the 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference*, 1682–1688 (2000).

[28] Casillas, J., Cordón, O. and Herrera, F. Learning Fuzzy Rules using Ant Colony Optimization Algorithms. In: *Proceedings of the 2nd International Workshop on Ant Algorithms (ANTS 2000)*, 13–21 (2000).

[29] Casillas, J., Cordón, O., Herrera, F. and Jesus, M. J. D. Genetic Tuning of Fuzzy Rule-Based Systems Integrating Linguistic Hedges. In: *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 1570–1574 (2001).

[30] Casillas, J., Cordón, O., Herrera, F. and Magdalena, L. *Accuracy Improvements in Linguistic Fuzzy Modeling*, vol. 129 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag (2003).

[31] Casillas, J., Cordón, O., Herrera, F. and Magdalena, L. *Interpretability Issues in Fuzzy Modeling*, vol. 128 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag (2003).

[32] Cerny, V. A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, 45:41–51 (1985).

[33] Chakraborty, D. and Pal, N. R. A Neuro-Fuzzy Scheme for Simultaneous Feature Selection and Fuzzy Rule-Based Classification. *IEEE Transactions on Neural Networks*, 15(1):110–123 (2004).

[34] Chan, A. and Freitas, A. A. A New Classification-Rule Pruning Procedure for an Ant Colony Algorithm. In: *Lecture Notes in Artificial Intelligence 3871*, 25–35. Springer-Verlag (2005).

[35] Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W. SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357 (2002).

[36] Chen, S. M., Lee, S. H. and Lee, C. H. A New Method for Generating Fuzzy Rules from Numerical Data for Handling Classification Problems. *Applied Artificial Intelligence*, 15(7):645–664 (2001).

[37] Chen, Y. and Chen, L. Parallel Ant Colony Algorithm for Mining Classification Rules. In: *Proceedings of the IEEE International Conference on Granular Computing*, 85–90 (2006).

[38] Cheong, F. and Lai, R. Constrained Optimization of Genetic Fuzzy Systems. In: *Accuracy Improvements in Linguistic Fuzzy Modeling*, 46–71. Springer-Verlag (2003).

[39] Clark, P. and Niblett, T. The CN2 Induction Algorithm. *Machine Learning*, 3(4):261–283 (1989).

[40] Cloete, I. and van Zyl, J. Fuzzy Rule Induction in a Set Covering Framework. *IEEE Transactions on Fuzzy Systems*, 14(1):93–110 (2006).

[41] Cordón, O., del Jesus, M. J. and Herrera, F. A Proposal on Reasoning Methods in Fuzzy Rule-Based Classification Systems. *International Journal of Approximate Reasoning*, 20:21–45 (1999).

[42] Cordón, O., del Jesus, M. J., Herrera, F. and Lozano, M. MOGUL: A Methodology to Obtain Genetic Fuzzy Rule-Based Systems Under the Iterative Rule Learning Approach. *International Journal of Intelligent Systems*, 14(11):1123–1153 (1999).

[43] Cordón, O., Gonzalez, A., Herrera, F. and Perez, R. Encouraging Cooperation in the Genetic Iterative Rule Learning Approach for Qualitative Modeling. In: *Computing with Words in Intelligent/Information Systems*, 95–117. Physica-Verlag (1998).

[44] Cordón, O., Herrera, F., Hoffmann, F. and Magdalena, L. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific Publishing (2001).

[45] Darab, M. A. D., Ebrahimi, M. and Shamshiri, M. R. Pass Evaluating in Simulated Soccer Domain using Ant-Miner Algorithm. In: *Proceedings of the IADIS Virtual Multi Conference on Computer Science and Information Systems* (2006).

[46] de Jong, K. A., Spears, W. M. and Gordon, D. F. Using Genetic Algorithms for Concept Learning. *Machine Learning*, 13:161–188 (1993).

[47] Dorigo, M., Bonabeau, E. and Theraulaz, G. Ant Algorithms and Stigmergy. *Future Generation Computer Systems*, 16(8):851–871 (2000).

[48] Dorigo, M. and Gambardella, L. M. Ant Colony System: Optimization by a Colony of Cooperating Ants. *IEEE Transactions on Systems, Man and Cybernetics B*, 26(1):29–41 (1996).

[49] Dorigo, M., Maziezzo, V. and Colorni, A. The Ant System: Optimization by a Colony of Cooperating Ants. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 26(1):29–41 (1996).

[50] Dorigo, M. and Stützle, T. *Ant Colony Optimization*. A Bradford Book, The MIT Press (2004).

[51] Duch, W., Adamczak, R. and Grabczewski, K. A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules. *IEEE Transactions on Neural Networks*, 11(2):1–31 (2000).

[52] Eberhart, R. C., Shi, Y. and Kennedy, J. *Swarm Intelligence*. Morgan Kaufmann (2001).

[53] Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap*. Chapman & Hall (1993).

[54] Eggermont, J. Evolving Fuzzy Decision Trees with Genetic Programming and Clustering. In: *Lecture Notes in Computer Science 2278*, 71–82. Springer-Verlag (2002).

[55] Fayyad, U. M., Piatetsky-Shapiro, G. and Smyth, P. Knowledge Discovery and Data Mining: Towards a Unifying Approach. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 82–88 (1996).

[56] Fisher, R. A. The use of Multiple Measurements in Taxonomic Problems. *Annual Eugenics*, II:179–188 (1936).

[57] Flake, G. W. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems and Adaptation*. The MIT Press (1998).

[58] Fogel, D. B. The Advantages of Evolutionary Computation. In: *Biocomputing and Emergent Computation*, 1–11. World Scientific Publishing Company (1997).

[59] Fogel, L. J., Owens, A. J. and Walsh, M. J. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons (1966).

[60] Forgy, C. L. Rete: A Fast Algorithm for the Many pattern/many Object Pattern Match Problem. *Artificial Intelligence*, 19:17–37 (1982).

[61] Freund, Y. and Schapire, R. E. Experiments with a New Boosting Algorithm. In: *Proceedings of the 13th International Conference on Machine Learning*, 148–156 (1996).

[62] Fürnkranz, J. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13:3–54 (1999).

[63] Galea, M. *Applying Swarm Intelligence to Rule Induction*. MSc dissertation, Division of Informatics, University of Edinburgh, UK (2002).

[64] Galea, M. and Shen, Q. Fuzzy Rules from Ant-Inspired Computation. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1691–1696 (2004).

[65] Galea, M. and Shen, Q. Iterative vs Simultaneous Fuzzy Rule Induction. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 767–772 (2005).

[66] Galea, M. and Shen, Q. Linguistic Hedges for Ant-Generated Fuzzy rules. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1973–1980 (2006).

[67] Galea, M. and Shen, Q. Simultaneous Ant Colony Optimization Algorithms for Learning Linguistic Fuzzy Rules. In: *Swarm Intelligence in Data Mining*, 75–100. Springer (2006).

[68] Galea, M., Shen, Q. and Levine, J. Evolutionary Approaches to Fuzzy Modelling for Classification. *The Knowledge Engineering Review*, 19(1):27–59 (2004).

[69] Galea, M., Shen, Q. and Singh, V. Encouraging Complementary Fuzzy Rules within Iterative Rule Learning. In: *Proceedings of the UK Workshop on Computational Intelligence (UKCI05)*, 15–22 (2005).

[70] Gambardella, L. M. and Dorigo, M. Ant-Q: A Reinforcement Learning Approach to the Travelling Salesman Problem. In: *Proceedings of the 12th International Conference on Machine Learning*, 252–260 (1995).

[71] Gambardella, L. M., Taillard, E. and Agazzi, G. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In: *New Ideas in Optimization*, 63–76. McGraw-Hill (1999).

[72] Gath, I. and Geva, B. Unsupervised Optimal Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781 (1989).

[73] Giraud, T., Pedersen, J. S. and Keller, L. Evolution of Supercolonies: The Argentine Ants of Southern Europe. *Proceedings of the National Academy of Sciences*, 99(9):6075–6079 (2002).

[74] Glover, F. and Laguna, M. *Tabu Search*. Kluwer (1997).

[75] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. and Lander, E. S. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531–537 (1999).

[76] Gonzalez, A. and Perez, R. A Learning System of Fuzzy Control Rules. In: *Genetic Algorithms and Soft Computing*, 202–225. Physica-Verlag (1996).

[77] Gonzalez, A. and Perez, R. Completeness and Consistency Conditions for Learning Fuzzy Rules. *Fuzzy Sets and Systems*, 96:37–51 (1998).

[78] Gonzalez, A. and Perez, R. SLAVE: A Genetic Learning System Based on an Iterative Approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191 (1999).

[79] Gonzalez, A. and Perez, R. A Study about the Inclusion of Linguistic Hedges in a Fuzzy Rule Learning Algorithm. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 7(3):257–266 (1999).

[80] Gonzalez, A., Perez, R. and Verdegay, J. L. Learning the Structure of a Fuzzy Rule: A Genetic Approach. *Fuzzy Systems and Artificial Intelligence*, 3(1):57–70 (1994).

[81] Goss, S., Aron, S., Deneubourg, J. L. and Pasteels, J. M. Self-Organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, 76(12):579–581 (1989).

[82] Green, D. M. and Swets, J. A. *Signal Detection Theory and Psychophysics*. John Wiley & Sons (1966).

[83] Gutjahr, W. J. A Graph-Based Ant System and its Convergence. *Future Generation Computer Systems*, 16(8):873–888 (2000).

[84] Gutjahr, W. J. ACO Algorithms with Guaranteed Convergence to the Optimal Solution. *Information Processing Letters*, 82(3):145–153 (2002).

[85] Hall, L. and Kanade, P. Swarm Based Fuzzy Clustering with Partition Validity. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 991–995 (2005).

[86] Hand, D. J., Mannila, H. and Smyth, P. *Principles of Data Mining*. MIT Press (2001).

[87] Hand, D. J. and Yu, K. Idiot's Bayes - Not so Stupid After all? *International Statistical Review*, 69(3):385–398 (2001).

[88] Harris, R. Fuzzifying GABIL: Evolving a Fuzzy Rulebase for Adaptive Machine Learning. Tech. rep., University of Nevada, Reno NV, USA (2002).

[89] Higashi, M. and Klir, G. J. Measures of Uncertainty and Information Based on Possibility Distributions. *International Journal of General Systems*, 9:43–58 (1983).

[90] Hirota, K. and Sugeno, M. *Industrial Applications of Fuzzy Technology*, vol. 2 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific (1993).

[91] Hoffmann, F. Combining Boosting and Evolutionary Algorithms for Learning of Fuzzy Classification Rules. *Fuzzy Sets and Systems*, 141(1):47–58 (2004).

[92] Holden, N. and Freitas, A. A. Web Page Classification with an Ant Colony Algorithm. In: *Lecture Notes in Computer Science 3242*, 1092–1102. Springer (2005).

[93] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975).

[94] Holland, J. H. and Reitman, J. S. Cognitive Systems Based on Adaptive Algorithms. In: *Pattern-Directed Inference Systems*. Academic Press (1978).

[95] Hoos, H. H. and Stützle, T. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann (2005).

[96] Hu, G.-M. and Wang, X.-Z. A Comparison between Fuzzy-ID3 and OFFSS-Based Fuzzy-ID3. In: *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, 4171–4173 (2004).

[97] Hunter, L. and Klein, T. Finding Relevant Biomolecular Features. 190–197 (1993).

[98] Iredi, S., Merkle, D. and Middendorf, M. Bi-Criterion Optimization with Multi Colony Ant Algorithms. In: *Lecture Notes in Computer Science 1993*, 359. Springer (2000).

[99] Ishibuchi, H., Nakashima, T. and Kuroda, T. A Fuzzy Genetics-Based Machine Learning Method for Designing Linguistic Classification Systems with High Comprehensibility.

In: *Proceedings of the 6th International Conference on Neural Information Processing*, 597–602 (1999).

[100] Ishibuchi, H., Nakashima, T. and Kuroda, T. A Hybrid Fuzzy GBML Algorithm for Designing Compact Fuzzy Rule-Based Classification Systems. In: *Proceedings of the 9th IEEE International Conference on Fuzzy Systems*, 706–711 (2000).

[101] Ishibuchi, H., Nakashima, T. and Morisawa, T. Voting in Fuzzy Rule-Based Systems for Pattern Classification Problems. *Fuzzy Sets and Systems*, 103(2):223–238 (1999).

[102] Ishibuchi, H., Nakashima, T. and Murata, T. A Fuzzy Classifier System that Generates Fuzzy if-then Rules for Pattern Classification Problems. In: *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, 759–764 (1995).

[103] Ishibuchi, H., Nakashima, T. and Murata, T. Genetic-Algorithm-Based Approaches to the Design of Fuzzy Systems for Multi-Dimensional Pattern Classification Problems. In: *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, 229–234 (1996).

[104] Ishibuchi, H., Nakashima, T. and Murata, T. Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 29(5):601–618 (1999).

[105] Ishibuchi, H., Nakashima, T. and Nii, M. *Classification and Modeling with Linguistic Information Granules: Advanced Approaches to Linguistic Data Mining*. Springer-Verlag (2005).

[106] Janikow, C. Z. A Genetic Algorithm Method for Optimizing Fuzzy Decsion Trees. *Information Sciences*, 89:275–296 (1996).

[107] Janikow, C. Z. Fuzzy Decision Trees: Issues and Methods. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 28(1):1–14 (1998).

[108] Japkowicz, N. Learning from Imbalanced Data Sets: A Comparison of various Strategies. In: *Learning from Imbalanced Data Sets: Papers from the AAAI Workshop*, 10–15 (2000).

[109] Japkowicz, N. Class Imbalances: Are we Focusing on the Right Issue? In: *Proceedings of the Workshop on Learning from Imbalanced Datasets II, International Conference on Machine Learning* (2003).

[110] Jensen, R. and Shen, Q. Fuzzy-Rough Data Reduction with Ant Colony Optimization. *Fuzzy Sets and Systems*, 149:5–20 (2005).

[111] Jeong, J. and Oh, S.-Y. Automatic Rule Generation for Fuzzy Logic Controllers using Rule-Level Co-Evolution of Subpopulations. In: *Proceedings of the 1999 Congress on Evolutionary Computation (CEC-1999)*, 2151–2156 (1999).

[112] Kallel, L., Naudts, B. and Rogers, A. *Theoretical Aspects of Evolutionary Computing*. Springer (2001).

[113] Kearns, M. A Bound on the Eorror of Cross-Validation using the Approximation and Estimation Rates. In: *Advances in Neural Information Processing Systems*, 183–189. The MIT Press (1996).

[114] Kent, M. J. and Hirschberg, D. S. Small Sample Statistics for Classification Error Rates

I: Error Rate Measurements. Tech. Rep. 96-21, Department of Information and Computer Science, University of California, Irvine CA, USA (1996).

[115] Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. Optimization by Simulated Annealing. *Science*, 220(4598):671–680 (1983).

[116] Klir, G. J. and Yuan, B. Operation of Fuzzy Sets. In: *Handbook of Fuzzy Computation*. Institute of Physics Publishing (1998).

[117] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1137–1143 (1995).

[118] Kosko, B. Fuzzy Entropy and Conditioning. *Information Sciences*, 40(2):165–174 (1986).

[119] Koza, J. R. Concept Formation and Decision Tree Induction using the Genetic Programming Paradigm. In: *Parallel Problem Solving from Nature*, 124–128. Springer-Verlag (1991).

[120] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Bradford Book, The MIT Press (1992).

[121] Krishnapuram, R. and Keller, J. M. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110 (1993).

[122] Kubat, M. and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One-Sided Sampling. In: *Proceedings of the 14th International Conference on Machine Learning*, 179–186 (1997).

[123] Leguizamon, G., Michalewicz and Zbigniew. A New Version of Ant System for Subset Problems. In: *Proceedings of the 1999 Congress on Evolutionary Computation*, 1459–1464 (1999).

[124] Ling, C. and Li, C. Data Mining for Direct Marketing: Problems and Solutions. In: *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, 73–79 (1998).

[125] Liu, B., Abbass, H. A. and McKay, B. Density-Based Heuristic for Rule Discovery with Ant-Miner. In: *Proc. 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, 180–184 (2002).

[126] Liu, B., Abbass, H. A. and McKay, B. Classification Rule Discovery with Ant Colony Optimization. *IEEE Computational Intelligence Bulletin*, 3(1):31–35 (2004).

[127] Liu, J. J. and Kwok, J. T.-Y. An Extended Genetic Rule Induction Algorithm. In: *Proceedings of the 2000 Congress on Evolutionary Computation*, 458–463 (2000).

[128] Luca, A. D. and Termini, S. A Definition of a Nonprobabilistic Entropy in the Setting of Fuzzy Set Theory. *Information and Control*, 20(4):301–312 (1972).

[129] Mamdani, E. H. Advances in the Linguistic Synthesis of Fuzzy Controllers. *Journal of Man-Machine Studies*, 8:669–678 (1976).

[130] Marin-Blazquez, J. G. and Shen, Q. From Approximate to Descriptive Fuzzy Classifiers. *IEEE Transactions on Fuzzy Systems*, 10(4):484–497 (2002).

[131] Mendes, R., de B. Voznika, F., Freitas, A. A. and Nievola, J. C. Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution. In: *Lecture Notes in Artificial Intelligence 2168*, 314–325. Springer-Verlag (2001).

[132] Mendil, B. and Benmahammed, K. Activation and Defuzzification Methods for Fuzzy Rule-Based Systems. *Journal of Intelligent and Robotic Systems*, 32:437–444 (2001).

[133] Merkle, D. and Middendorf, M. Fast Ant Colony Optimization on Runtime Reconfigurable Processor Arrays. *Genetic Programming and Evolvable Machines*, 3:345–361 (2002).

[134] Middendorf, M., Reischle, F. and Schmeck, H. Multi Colony Ant Algorithms. *Journal of Heuristics*, 8:305–320 (2002).

[135] Mingers, J. An Empirical Comparison of Selection Measures for Decision Tree Induction. *Machine Learning*, 3:319–342 (1989).

[136] Mühlenbein, H., Bendisch, J. and Voigt, H. M. From Recombination of Genes to the Estimation of Distributions II: Continuous Parameters. In: *Lecture Notes in Computer Science 1141*, 188–197. Springer (1996).

[137] Mühlenbein, H. and Paass, G. From Recombination of Genes to the Estimation of Distributions I: Binary Parameters. In: *Lecture Notes in Computer Science 1141*, 178–187. Springer (1996).

[138] Nakashima, T., Ishibuchi, H. and Murata, T. Evolutionary Algorithms for Constructing Linguistic Rule-Based Systems for High-Dimensional Pattern Classification Problems. In: *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, 752–757 (1998).

[139] Nauck, D. and Kruse, R. What are Neuro-Fuzzy Classifiers? In: *Proceedings of the 7th International Fuzzy Systems Association World Congress (IFSA'97)*, 228–233 (1997).

[140] Nauck, D. and Kruse, R. NEFCLASS-X: A Soft Computing Tool to Build Readable Fuzzy Classifiers. *BT Technology Journal*, 16(3) (1998).

[141] Newell, A. and Simon, H. A. *Human Problem Solving*. Prentice-Hall (1972).

[142] Nguyen, K. D. and Bourgeois, A. G. Ant Colony Optimal Algorithm: Fast Ants on the Optical Pipelined R-Mesh. In: *Proceedings of the 2006 International Conference on Parallel Processing (ICPP'06)*, 347–354 (2006).

[143] Nojima, Y. and Ishibuchi, H. Designing Fuzzy Ensemble Classifiers by Evolutionary Multiobjective Optimization with an Entropy-Based Diversity Criterion. In: *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, 59–62 (2006).

[144] Nomura, H., Hayashi, I. and Wakami, N. A Learning Method of Fuzzy Inference Rules by Descent Method. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 203–210 (1992).

[145] Oakes, M. P. Ant Colony Optimisation for Stylometry: The Federalist Papers. In: *Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC2004)*, 86–91 (2004).

[146] Paetz, J. Towards Ant Colony Optimization of Neuro-Fuzzy Interval Rules. In: *Proc.*

2005 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2005), 658–663 (2005).

[147] Pan, J., DeSouza, G. N. and Kak, A. C. FuzzyShell: A Large-Scale Expert System Shell using Fuzzy Logic for Uncertainty Reasoning. *IEEE Transactions on Fuzzy Systems*, 6(4):563–581 (1998).

[148] Parpinelli, R. S., Lopes, H. S. and Freitas, A. A. Data Mining with an Ant Colony Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332 (2002).

[149] Parsons, S. *Qualitative Methods for Reasoning Under Uncertainty*. The MIT Press (2001).

[150] Pazzani, M. J., Mani, S. and Shankle, W. R. Comprehensible Knowledge-Discovery in Databases. 596–601 (1997).

[151] Pedrycz, W. *Fuzzy Modelling: Paradigms and Practice*. Kluwer Academic Press (1996).

[152] Pedrycz, W. and Gomide, F. *An Introduction to Fuzzy Sets: Analysis and Design*. The MIT Press (1998).

[153] Pena-Reyes, C. A. and Sipper, M. FuzzyCoCo: A Cooperative-Coevolutionary Approach to Fuzzy Modeling. *IEEE Transactions on Fuzzy Systems*, 9(5):727–737 (2001).

[154] Phokharatkul, P. and Phaiboon, S. Handwritten Numerals Recognition using an Ant-Miner Algorithm. In: *International Conference on Control, Automation and Systems* (2005).

[155] Platt, J. C. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: *Advances in Kernel Methods – Support Vector Learning*, 185–208. The MIT Press (1998).

[156] Prati, R. C., Batista, G. and Monard, M. C. Class Imbalances Versus Class Overlapping: An Analysis of a Learning System Behaviour. In: *Lecture Notes in Artificial Intelligence 2972*, 312–321. Springer Verlag (2004).

[157] Quinlan, J. R. Induction of Decision Trees. *Machine Learning*, 1(1):81–106 (1986).

[158] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1992).

[159] Ragot, N. and Anquetil, E. A New Hybrid Learning Method for Fuzzy Decision Trees. In: *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, 1380–1383 (2001).

[160] Randall, M. A Parallel Implementation of Ant Colony Optimization. *Journal of Parallel and Distributed Computing*, 62:1421–1432 (2002).

[161] Rasmani, K. and Shen, Q. Weighted Linguistic Modelling Based on Fuzzy Subsethood Values. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 714–719 (2003).

[162] Rechenberg, I. *Evolutionstrategie: Optimierung Technisher Systeme Nach Prinzipien Der Biologishen Evolution*. Fromman-Holzboog Verlag (1973).

[163] Romao, W., Freitas, A. A. and Pacheco, R. C. S. A Genetic Algorithm for Discovering Interesting Fuzzy Prediction Rules: Applications to Science and Technology Data. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, 343–350 (2002).

[164] Rubinstein, R. Y. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190 (1999).

[165] Scheuermann, B., So, K., Guntsch, M., Middendorf, M., Diessel, O., Elgindy, H. and Schmeck, H. FPGA Implementation of Population-Based Ant Colony Optimization. *Applied Soft Computing*, 4:303–322 (2003).

[166] Sebag, M. and Rouveirol, C. Tractable Induction and Classification in First Order Logic Via Stochastic Matching. In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, 888–893 (1997).

[167] Shannon, C. E. and Weaver, W. *The Mathematical Theory of Communication*. The University of Illinois Press (1949).

[168] Shelokar, P. S., Jayaraman, V. K. and Kulkarni, B. D. An Ant Colony Classifier System: Application to some Process Engineering Problems. *Computers and Chemical Engineering*, 28:1577–1584 (2004).

[169] Shen, Q. and Chouchoulas, A. A Rough-Fuzzy Approach for Generating Classification Rules. *Pattern Recognition*, 35(11):2425–2438 (2002).

[170] Shen, Q. and Jensen, R. Selecting Informative Features with Fuzzy-Rough Sets and its Application for Complex Systems Monitoring. *Pattern Recognition*, 37(7):1351–1363 (2004).

[171] Slawinski, T., Krone, A., Hammel, U., Wiesmann, D. and Krause, P. A Hybrid Evolutionary Search Concept for Data-Based Generation of Relevant Fuzzy Rules in High Dimensional Spaces. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1432–1437 (1999).

[172] Smaldon, J. and Freitas, A. A. A New Version of the Ant-Miner Algorithm Discovering Unordered Rule Sets. In: *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'06*, 43–50 (2006).

[173] Smith, S. F. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Computer Science Department, University of Pittsburgh, USA (1980).

[174] Soroush, E., Abadeh, M. S. and Habibi, J. A Boosting Ant-Colony Optimization Algorithm for Computer Intrusion Detection. In: *Proceedings of the 2006 International Symposium on Frontiers in Networking with Applications (FINA 2006)* (2006).

[175] Soroush, E., Habibi, J. and Abadeh, M. S. Intrusion Detection using a Boosting Ant-Colony-Based Data Miner. In: *Proceedings of the 11th International CSI Computer Conference CSICC 2006*, 563–566 (2006).

[176] Srinivasan, A. and King, R. D. Feature Construction with Inductive Logic Programming: A Study of Quantitative Predications of Biological Activity Aided by Structural Attributes. *Data Mining and Knowledge Discovery*, 3:37–57 (1999).

[177] Stützle, T. and Dorigo, M. A Short Convergence Proof for a Class of ACO Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365 (2002).

[178] Stützle, T. and Hoos, H. MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8):889–914 (2000).

[179] Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press (1998).

[180] Takagi, T. and Sugeno, M. Fuzzy Identification of Systems and its Application to Modeling and Control. *IEEE Transactions on Systems, Man and Cybernetics*, 15:116–132 (1985).

[181] Theraulaz, G., Bonabeau, E. and Deneubourg, J.-L. Threshold Reinforcement and Division of Labour in Insect Socities. *Proceedings of the Royal Society of London Series B-Biologicial Sciences*, 265:327–335 (1998).

[182] Umano, M., Okamoto, H., Hatono, I., Tamura, H., Kawachi, F., Umedzu, S. and Kinoshita, J. Fuzzy Decision Trees by Fuzzy ID3 Algorithm and its Applications to Diagnosis Systems. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2113–2118 (1994).

[183] Vapnik, V. *Statistical Learning Theory*. Wiley-Interscience (1998).

[184] Walter, D. and Mohan, C. K. ClaDia: A Fuzzy Classifier System for Disease Diagnosis. In: *Proceedings of the Congress on Evolutionary Computation*, 1429–1435 (2000).

[185] Wang, C.-H., Liu, J.-F., Hong, T.-P. and Tseng, S.-S. A Fuzzy Inductive Learning Strategy for Modular Rules. *Fuzzy Sets and Systems*, 103:91–105 (1999).

[186] Wang, L.-X. and Mendel, J. M. Generating Fuzzy Rules by Learning from Examples. *IEEE Transactions on Systems, Man and Cybernetics*, 22(6):1414–1427 (1992).

[187] Wang, X., Chen, B., Qian, G. and Ye, F. On the Optimization of Fuzzy Decision Trees. *Fuzzy Sets and Systems*, 112:117–125 (2000).

[188] Wang, X. Z., Yeung, D. S. and Tsang, E. C. C. A Comparative Study on Heuristic Algorithms for Generating Fuzzy Decision Trees. *IEEE Transactions on Systems, Man and Cybernetics–B*, 31(2):215–226 (2001).

[189] Wang, Z. and Feng, B. E. . *Classification Rule Mining with an Improved Ant Colony Algorithm* (2004).

[190] Whitley, D. An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls. *Information and Software Technology*, 43(14):817–831 (2001).

[191] Witten, I. H. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann (2005).

[192] Yang, L., Widyantoro, D. H., Ioerger, T. and Yen, J. An Entropy-Based Adaptive Genetic Algorithm for Learning Classification Rules. In: *Proceedings of the Congress on Evolutionary Computation*, 790–796 (2001).

[193] Yuan, Y. F. and Shaw, M. J. Induction of Fuzzy Decision Trees. *Fuzzy Sets and Systems*, 69(2):125–139 (1995).

[194] Yuan, Y. F. and Zhuang, H. A Genetic Algorithm for Generating Fuzzy Classification Rules. *Fuzzy Sets and Systems*, 84(1):1–19 (1996).

[195] Zadeh, L. A. Fuzzy Sets. *Information Control*, 8:338–353 (1965).

[196] Zadeh, L. A. The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I. *Information Sciences*, 8(3):199–249 (1975).

[197] Zadeh, L. A. The Concept of a Linguistic Variable and its Application to Approximate Reasoning - II. *Information Sciences*, 8(4):301–357 (1975).

[198] Zadeh, L. A. The Concept of a Linguistic Variable and its Application to Approximate Reasoning - III. *Information Sciences*, 9(1):43–80 (1975).

[199] Zadeh, L. A. Fuzzy Logic. *IEEE Computer*, 21(4):83–92 (1988).

[200] Zlochin, M., Birattari, M., Meuleau, N. and Dorigo, M. Model-Based Search for Combinatorial Optimization: A Critical Survey. *Annals of Operations Research*, 131:373–395 (2004).

[201] Zwick, R., Carlstein, E. and Budescu, D. V. Measures of Similarity among Fuzzy Concepts: A Comparative Analysis. *International Journal of Approximate Reasoning*, 1(2):221–242 (1987).