

**Applying Swarm Intelligence  
to  
Rule Induction**

*Michelle Galea*

MSc Artificial Intelligence  
Division of Informatics  
University of Edinburgh  
2002



## Abstract

Ant colony optimisation is a relatively recent addition to the artificial intelligence toolbox. The induction of rules from data sets has, for a long time, been a goal for researchers in the AI community. The project set out in this dissertation applies the former to the latter. A system is implemented which successfully induces rules from data sets. This project re-implements a system first developed by Parpinelli *et al.* A number of extensions are made to the scope and capabilities of the system and the utility of these extensions is investigated in a series of experiments which use several of the standard rule induction data sets.

The system is also applied to a set of real-time engine control data. Some of the problems which arise in moving work from academia to industry are considered, and a number of avenues for future research and development effort are identified.

This work discusses other approaches to rule induction and sufficient evidence is presented to confirm that ant colony optimisation as a useful addition to the AI toolbox. We locate the approach followed by this project to the wider agenda of evolutionary computation within artificial intelligence.

## Acknowledgements

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

- Dr Qiang Shen, project supervisor, and Richard Jensen, project proposer and co-supervisor, University of Edinburgh
- Dr John Levine, University of Edinburgh, for his expert tuition on Evolutionary Computation
- Dr Rafael Parpinelli, principal author of the paper on which this project is based, CEFET-PR, Brazil, always happy to clarify matters
- Dr Robert Rae, University of Edinburgh and Mark Westwood, Max and Jamila, Software Development Team, Pilot Solutions Ltd, Edinburgh, for their expert advice on object-oriented analysis, design and programming.

I should also like to thank the rest of Qiang's team within the Approximate and Qualitative Reasoning Group, who have all, at some point or another, provided expert help.

## **Declaration**

I declare that this dissertation was composed by me, that the work contained herein is my own unless otherwise stated, and that this work has not been submitted for any other degree or professional qualification.

Michelle Galea.

# Contents

<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Project Background .....	1
1.2 Project Aim and Objectives .....	2
1.3 Project Approach.....	3
1.4 Summary of Achievements.....	3
1.5 Document Structure.....	4
<b>Chapter 2 Rule Induction</b> .....	<b>7</b>
2.1 What is Rule Induction .....	7
2.2 Approaches to Rule Induction .....	9
2.3 Ancillary Methods to Rule Induction .....	12
<b>Chapter 3 Swarm Intelligence</b> .....	<b>15</b>
3.1 What is Swarm Intelligence.....	15
3.2 Ant Colony Optimisation.....	17
3.3 Applying Ant Algorithms to Rule Induction .....	19
3.3.1 Ant Algorithms and Rule Induction.....	19
3.3.2 Ant Algorithms and Fuzzy Classification.....	24
<b>Chapter 4 Problem Investigation and Specification</b> .....	<b>27</b>
4.1 Problem Dependent Elements.....	28
4.1.1 Heuristic Function .....	28
4.1.2 Fitness Function.....	30
4.2 Problem Independent Elements .....	31
4.2.1 Transition Rule .....	32
4.2.2 Pheromone Update Strategy .....	33
<b>Chapter 5 System Analysis, Design and Implementation</b> .....	<b>35</b>
5.1 Conceptual Overview of the System .....	35
5.2 Physical Overview of the System .....	39
5.3 Modifications of Original Ant-Miner .....	42

5.3.1	Stopping Criteria for AntRuns .....	42
5.3.2	Ant Population Size .....	43
5.3.3	Pheromone Levels .....	44
5.3.4	Rule Pruning Procedure .....	46
<b>Chapter 6 Experiment Preparation.....</b>		<b>49</b>
6.1	Experimental Methodology .....	49
6.2	Data Sets .....	51
6.3	Data Pre-Processing.....	52
6.3.1	Discretization of Continuous Attributes .....	52
6.3.2	Attribute Subset Selection .....	54
<b>Chapter 7 Results and Analyses.....</b>		<b>57</b>
7.1	Test 1 – Comparison with Ant-Miner .....	58
7.2	Test 2 – Increasing Population Size.....	63
7.3	Test 3 – Changing the Transition Rule .....	66
7.4	Test 4 – Handling Missing Attribute Values .....	70
7.5	Test 5 – Use of Reduced Data Sets .....	73
7.6	Test 6 – Use of 'Tiger' Data Set .....	76
<b>Chapter 8 Conclusions and Future Work .....</b>		<b>81</b>
8.1	Summary and Conclusions .....	81
8.2	Future Work.....	82
<b>Bibliography .....</b>		<b>85</b>
<b>Appendix A User-Defined System Parameters.....</b>		<b>90</b>
<b>Appendix B Sample Outputs from Implemented System .....</b>		<b>92</b>
	Analysis of Tic-Tac-Toe Dataset.....	92
	Analysis of Ljubljana Dataset.....	101
<b>Appendix C Sample Detailed Test Results.....</b>		<b>110</b>
<b>Appendix D Comparison with Revised Ant-Miner Results.....</b>		<b>118</b>

## List of Figures

Figure 3.1 Overview of Ant-Miner Algorithm .....	20
Figure 5.1 High-level view of the system.....	36
Figure 5.2 Level I view of the system – a run of a k-fold cross-validation .....	37
Figure 5.3 Level II view of the system - creation of ants and rules.....	38
Figure 5.4 High-level view of system implementation.....	41
Figure 5.5 Creation of ants and rules in the original Ant-Miner algorithm .....	43
Figure 6.1 Sample output of C4.5 induced decision tree.....	53
Figure 6.2 Sample discretization resulting from <i>C4.5-Disc</i> .....	54
Figure 6.3 Reduct for Dermatology data set.....	55
Figure 7.1 Comparing Predictive Accuracy with Ant-Miner .....	59
Figure 7.2 Comparing Rule Sets with Ant-Miner.....	61
Figure 7.3 Comparing Elapsed Time with Increase in Q-Value.....	69
Figure 7.4 Predictive Accuracy of Rule Sets Generated from Reduced Data Sets.....	74
Figure 7.5 Size of Rule Sets Generated by Reduced Data Sets .....	75

## List of Tables

Table 3.1 Accuracy Rate of Ant-Miner vs C4.5 .....	23
Table 3.2 Simplicity of Rule Sets Discovered by Ant-Miner vs C4.5 .....	23
Table 3.3 Accuracy Rate of Ant-Miner vs CN2 .....	23
Table 3.4 Simplicity of Rule Sets Discovered by Ant-Miner vs CN2 .....	23
Table 6.1 Data Set Properties .....	51
Table 6.2 Original vs Reduced Number of Attributes in Data Sets .....	55
Table 7.1 Comparing Predictive Accuracy with Ant-Miner .....	58
Table 7.2 Comparing Rule Sets and Rules with Ant-Miner .....	61
Table 7.3 (a) and (b) - Comparing Predictive Accuracy with Increase in Population Size .....	64
Table 7.4 Comparing Elapsed Time with Increase in Population Size .....	65
Table 7.5 (a) and (b) - Comparing Predictive Accuracy with Increase in Q-Value .....	67
Table 7.6 Comparing Elapsed Time with Increase in q-Value .....	68
Table 7.7 Comparing Results for the Hepatitis Data Set with Missing Attribute Values .....	71
Table 7.8 Distribution of Missing Values for the Hepatitis Data Set .....	72
Table 7.9 Number of Attributes and Terms Reduced by Feature Selection Procedure .....	73
Table 7.10 Predictive Accuracy of Rule Sets Generated from Reduced Data Sets .....	74
Table 7.11 Size of Rule Sets and Rules Generated by Reduced Data Sets .....	75



# Chapter 1

## Introduction

This chapter sets the background for this project, describes its aims and objectives, the methodology followed, and outlines the achievements and conclusions drawn. It also describes the organisation of this document and summarises the information contained within the various chapters.

### 1.1 Project Background

The paragraph quoted below from the original project description, [Jensen & Shen, 2002a], sets the scene for this project:

'Rule induction from historical data provides an important key to the solution of the knowledge acquisition bottleneck in the development of intelligent reasoning systems. Numerous studies have been made on generating rules from example data, in research fields ranging from symbolic machine learning to non-symbolic connectionist computing. However, algorithms developed in machine learning are mainly for discrete domains, whilst neural network-based approaches work by making strong assumptions such as rules involving only one premise attribute or a small number of attributes which are pre-selected by a statistical method. Alternative approaches are necessary in order to produce rules

capable of being applied to situations where data are real-valued, dynamic, multi-dimensional and uncertain.'

The alternative approach suggested by the project proposal was Evolutionary Computation, specifically Swarm Intelligence. Swarm Intelligence is an approach whereby simple agents interact with each other and their environment to produce a global behaviour significantly more complex – and useful – than their individual actions.

## 1.2 Project Aim and Objectives

The project aim was to investigate the application of swarm intelligence to rule induction. This was to be accomplished by the application of ant algorithms, a suite of optimisation algorithms inspired by the foraging behaviour of ants.

The feasibility of applying swarm intelligence to this problem had already been established by [Casillas *et al*, 2000] within the context of learning fuzzy rules, and by [Parpinelli *et al*, 2001] for learning crisp rules using ant algorithms within the medical domain. This project sought to build on the work already done by Parpinelli *et al* by amending ant behaviour and how they interact with their environment.

The objectives of this project were designed to make exploration of the search space more explicit and yet maintain effective exploitation of learnt information. Specifically, the project was to:

- investigate the role of exploration during rule building by ants; and,
- provide more directed search during rule building.

## 1.3 Project Approach

The above objectives were to be achieved by:

- implementing the basic ant algorithm, Ant-Miner, developed by Parpinelli *et al* in the previously mentioned work;
- incorporating features from other ant algorithms to investigate the balance between exploration and exploitation of previously learnt information during rule building, and provide a more directed search by making pheromone updates count for more.

The intention was to use the same data sets that Parpinelli *et al* used, to make comparisons between their basic ant algorithm and modifications made to it. The same data set pre-processing methods were utilised.

The evaluation criteria were also kept the same – predictive accuracy achieved by the generated rule sets, and rule set simplicity (i.e. the number of rules and the number of terms in a rule antecedent). The experimental methodology employed is covered in more detail in Chapter 6.

## 1.4 Summary of Achievements

The basic algorithm Ant-Miner was replicated as faithfully as possible. The known differences between this project's implementation of Ant-Miner and the original algorithm have been documented in Chapter 5, and a justification provided. Attempts have been made throughout the discussion of the results in Chapter 7 to account for performance differences between the original Ant-Miner and the implemented system.

The modifications made to achieve the specified objectives suggest that a balance between exploration and exploitation and more guided search can improve predictive accuracy, but

appear to have little impact on rule set simplicity. Follow-up experiments and work that may confirm these indications are suggested in Chapter 7 and Chapter 8.

In addition to the stated objectives the implementation has been adapted to handle data sets with missing attribute values, basic tests have been re-run on data sets that have been put through an attribute subset selection process first, and an additional real-world application data set has been tried. The results obtained highlight potential issues of dealing with real-world data and confirm the benefits of employing feature subset selection on data sets to improve both accuracy and execution time.

## 1.5 Document Structure

Chapter 2 introduces rule induction and the contexts in which it is applied. It also discusses several of the major approaches to rule induction and closely related procedures such as rule pruning and attribute selection.

Chapter 3 introduces swarm intelligence and in particular ant colony optimisation, a suite of algorithms applied to optimisation problems. Recent work in the application of ant algorithms to rule induction and classification is described in some detail. The work of this project is directly based on one of the papers discussed.

In Chapter 4 the possible ways of extending the work in [Parpinelli *et al*, 2002a] are discussed, and in so doing the scope for the work in this project is defined and justified.

Chapter 5 describes the main ideas of the analysis, design and implementation phases of this project. It also highlights the relatively easy and low-maintenance future extensions that may be made to the implemented system in order to further explore Ant-Miner's capabilities and limitations.

The experimental methodology employed and the data set pre-processing required are explained in Chapter 6, with Chapter 7 presenting an analysis of the results obtained. The conclusions and suggestions for future work are presented in Chapter 8.

Several appendices follow providing more detailed information on the implemented system, such as system parameters and sample outputs, and an example of more detailed test results. A brief comparison between this implementation's results and recently revised Ant-Miner figures is also made.



## **Chapter 2**

### **Rule Induction**

The first section of this chapter introduces rule induction and the contexts within which it is applied. Section 2.2 then discusses several of the major approaches to rule induction, while Section 2.3 mentions two closely related procedures and puts them within the context of rule induction and this project.

#### **2.1 What is Rule Induction**

Rule induction is the extraction of knowledge from data. If data is thought of as raw facts and figures of limited direct use in decision-making, then knowledge may be defined as a relevant and useable summary of that data, a necessity to the operational, managerial and strategic planning processes of an organisation.

For instance, with detailed data available about credit/debit card use by customers, an organisation may be able to abstract from this data new customer buying trends. This may consequently lead to an amendment of stock-reordering policies, timely sales promotions or other relevant activities that take full advantage of the emerging knowledge. This illustrates the use of rule induction as a core activity within the broader context of data mining and knowledge discovery in databases [Fayyad *et al*, 1996] .

Another important use of rule induction is within the context of developing intelligent reasoning systems. High-level knowledge, usually in the form of rules, is required for the decision-making processes of such systems. Traditionally this has been obtained via discussions with domain experts, though this approach has many problems and shortcomings – 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 [Buchanan & Wilkins, 1993]. This knowledge acquisition phase is often the main bottleneck within the knowledge engineering process and therefore considerable effort has been expended in designing rule induction algorithms that automatically or semi-automatically generate useful knowledge from detailed historical data already available.

The specific task to which a rule induction algorithm is to be used during this project is classification. The classification task as stated in [Berry & Linoff, 1997] is 'characterized by a well-defined definition of the classes, and a training set consisting of preclassified examples. The task is to build a model of some kind that can be applied to unclassified data in order to classify it'.

Perhaps the two most important evaluation criteria used in this context are predictive accuracy and comprehensibility. These same criteria are also used in this project to make comparisons between the implemented algorithm and modifications made to it.

Predictive accuracy, also called generalisation, is a measure of how well the model built by the rule induction algorithm performs in classifying previously unseen instances.

Comprehensibility refers to the extent to which the model built can be understood by a human user. Comprehensible models may therefore help to:

- validate a domain expert's knowledge,
- refine an incomplete or inaccurate domain theory [Pazzani & Kibler, 1992],
- provide confidence in the automated system now affecting the decisions,

- highlight previously undiscovered knowledge [Hunter & Klein, 1993], and,
- optimise the system performance by highlighting both significant (playing a major role in the built model) and insignificant (playing no part in the model) features of the domain [Craven & Shavlik, 1993].

## 2.2 Approaches to Rule Induction

A few of the most common approaches to rule induction, and the more relevant to this project, are outlined below, with their relative merits and limitations.

**Decision trees** are a powerful model for classification. They are produced by techniques such as *C4.5* [Quinlan, 1992] and *CART* [Breiman *et al*, 1984]. They follow a 'divide-and-conquer' strategy whereby the training data is partitioned into disjoint subsets and the algorithm is applied recursively to each subset.

An internal node of an induced tree specifies a test on an attribute of the data set. Each outgoing branch of the node corresponds to a possible result of the test and leaf nodes represent the classification, or class label, to be assigned to an instance. To classify an instance a path from the root node of the 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.

A major advantage of a decision tree is that as a derived model it explicitly captures the decision-making process. The individual paths of the tree may then be expressed in the form of the highly-accessible IF-THEN rule form, also known as production rules.

Decision trees are limited in that during their creation they attempt to identify pure subsets of instances in the training data. This can cause an overfitting to instances that are incorrect or

contradictory, and therefore decrease the generalisation power of the end-model. Ancillary procedures exist that help to overcome this problem, including [Quinlan, 1987], and rule pruning which is discussed in the next section.

**Decision lists** are similar to decision trees in that they embody an explicit representation of the knowledge extracted from the training data. However, they follow a 'separate-and-conquer' approach where one rule is built to cover a subset of the training set, and then more rules are built to cover the remaining instances recursively. This strategy was first applied in the *AQ* family of algorithms [Michalski, 1969], and subsequently used as the basis of many other algorithms such as [Rivest, 1987], *CN2* [Clark & Niblett, 1989], and Ant-Miner [Parpinelli *et al*, 2002a], the ant algorithm upon which this project is based.

The end result of the algorithm is an *ordered* list of IF-THEN rules that are applied in sequence when a new instance requires classification. If the first rule in the list does not cover the instance, i.e. does not have matching values for the attributes in both the rule and the instance, then the next one is tried. If the second does not work then the third rule down the list is tried, and so on. Once an instance is classified by a rule no more rules are tried. If none of the rules cover the instance then a default rule at the bottom of the decision list is 'fired', i.e. all unclassified instances reaching the default rule get assigned that rule's class label.

A disadvantage of ordered rule lists is that the individual rules themselves may be harder to comprehend; a rule in a list needs to be taken in the context of all its preceding rules. For instance, in the example below reproduced from [Clark & Boswell, 1991], the second rule, if considered on its own, is incorrect as birds also have two legs:

```
        if feathers = yes           then class = bird
else if legs = two                 then class = human
else ...
```

Like decision trees, decision lists tend to suffer from overfitting to noisy training data so a rule pruning process, discussed below, is generally applied. A hybrid algorithm that uses a

separate-and-conquer approach to build partial decision trees is discussed in [Eibe & Witten, 1998].

The **fuzzy logic**-based approach has the major advantage of being able to represent inexact data in a way that is more naturally understood by humans. 'Fuzzy rules represent in a straightforward way "common-sense" knowledge and skills, or knowledge that is subjective, ambiguous, vague, or contradictory' [Kasabov, 1996].

The most straightforward fuzzy rule induction methods may be considered as extensions to traditional methods where fuzzy rather than crisp attributes are used. For instance, *ID3*, a decision tree method, has been extended to produce *fuzzy ID3* in the work of [Umano *et al*, 1994] and [Janikow, 1998], while more recent work in the area of fuzzy classification includes [Chen *et al*, 2001].

Inspired by Darwinian principles of evolution, a relatively recent approach to knowledge discovery is **evolutionary computation**. Here, heuristic rules are re-iteratively applied to a population of solutions, modifying or replacing members of the population so that each new generation, on average, tends to be better than the previous one according to some predefined fitness criteria. The particular branches of evolutionary computation that have mostly been applied to the rule induction problem are genetic algorithms and genetic programming.

The appeal of this approach is that it provides an effective mechanism to conduct a global search over a large search space. It also tends to cope better with attribute interaction than greedy rule induction algorithms and may be used for the generation of both crisp and fuzzy rules. Several reports indicate that the results obtained compare very favourably with results generated using more the widely-accepted approaches mentioned above, [Kwedlo & Kretowski, 1998], [Cordon *et al*, 1999], [Mendes *et al*, 2001] and [Freitas, 2002].

Other approaches to rule induction not discussed here include instance-based learning [Aha *et al*, 1991], neural networks [Rumelhart & McClelland, 1986], Bayesian networks [Jordan, 1999], and logistic regression [Agresti, 1990]. The main drawback of many of these methods

is their lack of explanatory power, though their predictive accuracy compares well to, and in some situations is better than, the previously mentioned approaches.

## 2.3 Ancillary Methods to Rule Induction

**Rule pruning** is briefly discussed here as it forms an integral part of the algorithm implemented for this project. As stated in the previous section rule pruning is a necessary process to avoid overfitting to noisy training data.

For decision trees pruning is the removal of those lower parts of the tree where tests on the internal nodes (attributes) are chosen based on a relatively small number of instances in the training set. Pruning can take place either during the construction of the tree or afterwards and these approaches are called prepruning and postpruning respectively.

For decision lists there are also two main strategies for rule pruning. The first builds a full rule set and then simplifies it by eliminating tests (attributes) from rules or by deleting individual rules. This is done by globally optimizing the rule set according to some predefined pruning criteria. The second strategy is called incremental pruning as each rule is simplified immediately after it has been generated by the induction algorithm [Eibe, 2000].

Ant-Miner, the algorithm implemented in this project adopts the second incremental approach and the particulars of the technique will be discussed in Section 3.3.2 Ant Algorithms and Fuzzy Classification.

The other supplementary process requiring a mention is **attribute selection**. This process was performed on the data sets used throughout this project as part of one of the tests performed. In principle one may use all the attributes available in a data set, but there are reasons for reducing the dimensionality of the data on occasion, such as faster classification and the saving of storage space. However, more fundamental reasons are the curse of

dimensionality that states that the complexity of a rule induction algorithm will increase exponentially with the number of attributes in the training set, and overfitting.

There are two basic strategies one may adopt for attribute selection. The first is simply to select a subset from the original set of attributes, and the second is to define fewer new variables as transformations and combinations of the original ones. The first has the advantage that once the subset has been identified, only measurements or observations on attributes in that subset need be taken for future instances. The second is more flexible and so may lead to more powerful classifiers [Hand, 1997].

The first approach was adopted for this project and the particular technique used is described in Section 6.3 Data Pre-Processing.



## Chapter 3

### Swarm Intelligence

This chapter introduces swarm intelligence by describing its main features, benefits and applications. Ant colony optimisation, a specific sub-branch of swarm intelligence, is then described and put into the context of rule induction and classification.

#### 3.1 What is Swarm Intelligence

Swarm intelligence is a branch of evolutionary computation, which is the application of methods inspired by the natural world to hard problems in artificial intelligence. The expression swarm intelligence is applied to any work involving the design of algorithms or distributed problem-solving devices (including, for example, robotic systems), inspired by the collective behaviour of social insects and other animal societies [Bonabeau *et al*, 1999].

A fundamental concept underlying the behaviour of social insects is self-organisation – 'a spontaneously formed higher-level pattern of structure or function that is emergent through the interactions of lower-level objects' [Flake, 1998]. 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 organisation 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. Swarm

intelligence 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 other fundamental concept for swarm intelligence is 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, in mandibular or visual contact, and other physical or direct insect-to-insect contact. Indirect communication is evidenced when one individual interacts with and modifies the environment and at a later time another individual responds to these changes in the environment. This indirect communication is called stigmergy and a particular example will be given in the discussion on ant colony optimisation below.

The potential benefits therefore of imitating social insect structural models and behaviour in designing solutions to problems in artificial intelligence include:

- robustness, arising from the ability of the colony as a whole to function though some individuals may fail;
- flexibility, from adaptation to changing environments; and
- decentralisation, removing the need to program overall control.

The specialised emergent behaviour of social insects has inspired practical applications in many different areas. For example,

- the cemetery organisation and brood sorting activity of ants has led to new clustering algorithms (e.g. graph partitioning and graph colouring);
- their inclination towards division of labour has produced resource and task allocation implementations (e.g. distributed mail retrieval);
- their cooperation in moving large objects has led to new collective robotic architectures (e.g. swarm-based robotics [Bay, 1995]); and

- their foraging behaviour has inspired new optimisation algorithms called ant algorithms, or ant colony optimisation (e.g. travelling salesman problem, bin packing [Ducatellet & Levine, 2001], quadratic assignment problem, dynamic network routing).

This last item, ant colony optimisation, is the approach taken to rule induction in this project and is therefore described in more detail in the following section.

## 3.2 Ant Colony Optimisation

Ant Colony Optimisation (ACO) is an agent-based meta-heuristic motivated by the foraging strategies of real ants.

Ants have the ability to find the shortest path from their nest to a food source. When a food source is first located several ants may have taken several different paths in reaching that food source. When they move ants leave a trail of chemical substance called a pheromone along the path, and in foraging for food and taking it back to their nest they will follow the path with the greatest amount of pheromone laid upon it. Pheromone trails evaporate if more ants do not come along to reinforce the strength. The ants that find the shortest route to the food will arrive back at the nest quicker than the others and will have laid more pheromone along this shortest path. Therefore, when new ants seek to travel to the food source, since they are guided by the amount of pheromone on the path they will take the shortest route. It has been observed that eventually all foraging ants converge on the shortest path to a food source.

The first ACO or ant algorithm was developed by Marco Dorigo as his PhD thesis in 1992, and later published under the name of Ant System (AS) in [Dorigo *et al*, 1996]. The application was the travelling salesman problem (TSP), where the goal is to find a closed path of minimal length connecting  $n$  given cities with each city visited once and only once.

Any combinatorial problem for which the following listed elements can be defined may be solved by an ant algorithm. These elements will be described in the context of Dorigo's Ant System for the TSP:

- an appropriate *problem representation* is required that allows our artificial ants to incrementally build a solution using a *probabilistic transition rule*. The main idea generally is to model the problem as the search for a best path through a graph. In AS we have a fully-connected graph  $G = (N,E)$  where the set of nodes  $N$  represent the cities and the set of edges  $E$  the connections between the cities. Ants build their solution to the TSP by moving on the problem graph from one city to another until they complete a closed tour;
- a local *heuristic* provides guidance to an ant in choosing the next node for the path it is building. This heuristic is problem specific and for TSP it is the inverse of the distance between two cities;
- the *probabilistic transition rule* determines which node/city an ant should visit next. The transition rule is dependent on the *heuristic* value and the *pheromone* level associated with an edge joining two nodes/cities. The transition rule for AS is called a random proportional transition rule and will be explained in more detail in the following section;
- a *constraint satisfaction method* that forces the construction of feasible rules is also required. In the case of AS for the TSP, an ant must visit each city once and only once during its solution construction;
- a *fitness function* determines the fitness, or quality, of the solution built by an ant. For the TSP the ant that produces a closed tour of minimal length has the greatest fitness; and finally
- the *pheromone update rule* specifies how to modify the pheromone trail laid along edges of the graph. The pheromone levels are an essential part of the *transition rule* mentioned above. In AS pheromone evaporates from all edges, but new pheromone is deposited by all ants on visited edges with the value deposited proportional to the quality of the solution built by the ants.

An algorithm that will tie these elements together is described in the following section in the context of rule induction.

### 3.3 Applying Ant Algorithms to Rule Induction

The application of ant algorithms to rule induction and classification is a research area still relatively unexplored. The appeal of this approach is that it provides an effective mechanism for conducting a more global search, and, if an analogy is made with the application of genetic algorithms and genetic programming to rule induction, we might expect an ant algorithm to cope better with attribute interaction than greedy rule induction algorithms. Furthermore, the application of ant algorithms requires minimum understanding of the problem domain. The problem-specific components are the heuristic function and a fitness function both of which may be readily borrowed from other existing rule induction algorithms.

The two directly-relevant papers in this area are described in the following subsections, in particular the work of Parpinelli *et al* upon which this project is based.

#### 3.3.1 Ant Algorithms and Rule Induction

Parpinelli *et al* first worked on the application of ant algorithms to rule induction in [Parpinelli *et al*, 2001], and then extended their work and republished in [Parpinelli *et al*, 2002a]. It is this second paper that is mostly referred to in this document. However, they have done further work on optimising parameter values and on the complexity of the algorithm and this is reflected in [Parpinelli *et al*, 2002b], which is to appear in a special issue of the *IEEE Transactions on Evolutionary Computation*, 2002.

The purpose of Ant-Miner, their algorithm, is to use ants to create rules describing an underlying data set. Rules are expressed in the form:

*IF* <conditions> *THEN* <class>

The <conditions> part, or rule antecedent, contains a logical combination of terms where each term is an attribute plus a particular value from its domain, e.g. attributeA=value1. The <class> part, or rule consequent, contains the class label assigned to the particular combination of specific terms.

The overall approach of Ant-Miner is a 'separate-and-conquer' one. It starts with a full training set, creates a 'best' rule that covers a subset of the training data, adds the best rule to its `Discovered_Rule_List`, removes the instances covered by said rule from the training data, and starts again with a reduced training set. This goes on until only a few instances are left in the training data (fewer than `maxInstUncovered`), at which point a default rule is created to cover those remaining instances. Below is an overview of the algorithm adapted from [Parpinelli *et al*, 2002a]:

```

Discovered_Rule_List = [] // empty initially
Training Set = all training cases
WHILE (No. of uncovered cases in the Training Set > maxInstUncovered) // 'AntRun'
  i=0
  REPEAT // 'iteration'
    i=i+1
    Ant(i) incrementally constructs classification rule
    Assign rule consequent
    Prune just-constructed rule
    Update pheromone of trail followed by Ant(i)
  UNTIL (i >= noAnts) OR (Ant(i) constructs same rule as previous
                        maxRulesConverge-1 Ants)
  Select the best rule among all constructed rules
  Add rule to Discovered_Dule_List
  Remove training cases correctly covered by selected rule from Training Set
END WHILE
create default rule
output Discovered_Rule_List

```

Figure 3.1 Overview of Ant-Miner Algorithm

In a few respects Ant-Miner is quite different from other ant algorithms. The first difference arises from the 'separate-and-conquer' approach it uses. The parts in the figure above between `WHILE` and `END WHILE`, called an `AntRun` from now on, may be considered as an algorithm that is run in succession several times, each time being run against a reduced data set.

The other main difference from other ant algorithms is the concept of iteration and population size. In a normal ant algorithm the part between REPEAT and UNTIL would be an iteration with each iteration creating a fixed number of ants, normally greater than one. Ant-Miner may be considered as running iterations with a population size of one ant.

During an AntRun each ant starts with an empty rule, i.e. no term in its rule antecedent, and adds one term at a time. The choice of a term to be added to the current partial rule antecedent depends on both the heuristic value (based on term entropy) and the pheromone level associated with each term. The choice is made probabilistically but is biased towards terms that have relatively higher heuristic and pheromone values. These details are discussed in the next chapter. A term is not considered for inclusion in the current partial rule if it has already been previously selected, or if its associated attribute is already present in the rule antecedent due to another term having been previously selected.

An ant will stop building a rule antecedent if it has selected one term from each of the available attributes, or if whichever term that may be added next would reduce the number of training instances covered by the current rule antecedent below a pre-determined threshold, called `minInstPerRule`. This criterion acts as a control on the amount of overfitting allowed to the training data; the greater the value of this threshold, the more general the rule antecedents created by ants are forced to be.

Once an ant has stopped building a rule antecedent a rule consequent is chosen. This is done by assigning to the rule consequent the class label of the majority class among the instances covered by the built rule antecedent.

The rule is then pruned in order to improve its quality and comprehensibility (by making it shorter). The basic idea is to iteratively remove one term at a time from the rule while this process improves the rule quality as defined by a fitness function. On a first iteration each term in turn is temporarily removed from the rule antecedent, a new rule consequent is assigned, and the rule quality is reassessed. At the end of the iteration the term that is actually removed is the one that improves the rule quality the most. The next iteration starts with a slightly shorter rule, and so on, until at best one term remains in the rule antecedent.

The rule pruning process will stop before this point if at any point the removal of any term will not improve the rule quality.

At this point the pheromone levels of terms is updated. The overall effect is that terms that appear in the rule antecedent have their pheromone levels increased, while all other terms have their pheromone levels decreased.

The process whereby an ant creates a rule is repeated for at most a predefined number of ants – `noAnts`. However, the process may also stop if the current ant has just constructed a rule that is exactly the same as the previous (`maxRulesConverge-1`) rules. This is similar to the situation when real ants converge to the same shortest path between their nest and a food source.

The best rule created during this AntRun is added to the `Discovered_Rules_List`, the training set is appropriately reduced and another AntRun that will generate a best rule to cover some more instances from the training set takes place. Ant-Miner will stop when there are fewer instances in the training set than the value set by `maxInstUncovered` parameter. At this point the default rule is created. This rule has no antecedent, only a conclusion that is the same class as the majority class of the remaining instances. The default rule is added to the bottom of the `Discovered_Rules_List`. Classification of an instance using this ordered rule list is as described above in Section 2.2.

Ant-Miner was evaluated using six benchmark data sets against *C4.5* [Parpinelli *et al*, 2002a] and *CN2* [Parpinelli *et al*, 2002b]. The evaluation criteria were predictive accuracy and simplicity of rule sets (i.e. the number of rules in a rule set, and the number of terms in a rule. The fewer the better, on both counts). Ten-fold cross-validations were performed and Ant-Miner compared favourably against both algorithms. Their findings are reproduced here.

Dataset	Ant-Miner Accuracy %	C4.5 Accuracy %
Ljubljana Breast Cancer	75.42 (+/- 10.99)	73.34 (+/- 3.21)
Wisconsin Breast Cancer	96.04 (+/- 2.80)	95.02 (+/- 0.31)
Tic-Tac-Toe	73.04 (+/- 7.60)	83.18 (+/- 1.71)
Dermatology	86.55 (+/- 6.13)	89.05 (+/- 0.62)
Hepatitis	90.00 (+/-9.35)	85.96 (+/- 1.07)
Cleveland Heart Disease	59.67 (+/- 7.52)	58.33 (+/- 0.72)

Table 3.1 Accuracy Rate of Ant-Miner vs C4.5

Dataset	No. of Rules		No. of Terms/No. of Rules	
	Ant-Miner	C4.5	Ant-Miner	C4.5
Ljubljana Breast Cancer	7.2 (+/- 0.60)	6.2 (+/- 4.20)	1.36	2.06
Wisconsin Breast Cancer	6.20 (+/- 0.75)	11.1 (+/- 1.45)	1.97	3.97
Tic-Tac-Toe	8.50 (+/- 1.86)	83.0 (+/- 14.1)	1.18	4.63
Dermatology	7.00 (+/-0.00)	23.2 (+/- 1.99)	11.57	3.95
Hepatitis	3.40 (+/-0.49)	4.40 (+/- 0.93)	2.41	1.93
Cleveland Heart Disease	9.50 (+/- 0.92)	49.0 (+/- 9.40)	1.71	3.74

Table 3.2 Simplicity of Rule Sets Discovered by Ant-Miner vs C4.5

Dataset	Ant-Miner Accuracy %	CN2 Accuracy %
Ljubljana Breast Cancer	75.28 (+/- 2.24)	67.69 (+/- 3.59)
Wisconsin Breast Cancer	96.04 (+/-0.93)	94.88 (+/- 0.88)
Tic-Tac-Toe	73.04 (+/- 2.53)	97.38 (+/- 0.52)
Dermatology	94.29 (+/-1.20)	90.38 (+/-1.66)
Hepatitis	90.00 (+/-3.11)	90.00 (+/- 2.50)
Cleveland Heart Disease	59.67 (+/- 2.50)	57.48 (+/- 1.78)

Table 3.3 Accuracy Rate of Ant-Miner vs CN2

Dataset	No. of Rules		No. of Terms/No. of Rules	
	Ant-Miner	CN2	Ant-Miner	CN2
Ljubljana Breast Cancer	7.10 (+/- 0.31)	55.40 (+/- 2.07)	1.28	2.21
Wisconsin Breast Cancer	6.20 (+/- 0.25)	18.60 (+/- 1.45)	1.97	2.39
Tic-Tac-Toe	8.50 (+/- 0.62)	39.70 (+/- 2.52)	1.18	2.9
Dermatology	7.30 (+/-0.15)	18.50 (+/- 0.47)	3.16	2.47
Hepatitis	3.40 (+/-0.16)	7.20 (+/- 0.25)	2.41	1.58
Cleveland Heart Disease	9.50 (+/- 0.92)	42.40 (+/- 0.71)	1.71	2.79

Table 3.4 Simplicity of Rule Sets Discovered by Ant-Miner vs CN2

These tables set the results of the current project, which is based directly on Ant-Miner, in the wider framework of the comparison of an evolutionary approach with the more traditional deterministic approach.

The authors suggest two main directions for future research: adapting Ant-Miner to deal directly with continuous attributes, and investigating changes in the heuristic function and pheromone update strategy.

### 3.3.2 Ant Algorithms and Fuzzy Classification

As far as can be ascertained, Casillas *et al* were the first to apply ant algorithms to rule induction and classification [Casillas *et al*, 2000]. (An earlier work within the field of data mining exists, but was applied to clustering [Monmarche, 1999] ). The approach taken is quite different from that of Parpinelli *et al*, and not merely due to the fuzzy domain chosen.

The main difference lies in the problem representation. In Parpinelli's work the nodes of the problem graph are the terms that may make up a rule antecedent. An ant travels from node to node – gathering terms for the rule antecedent it is building. It need not visit every node and once a term belonging to a particular attribute is selected then the other terms of that attribute are not considered for addition to the partial rule antecedent. When an ant stops building the rule conclusion is determined and added to the rule.

In [Casillas *et al*, 2000] the authors interpreted the problem as an assignment task and adapted a previous ant algorithm designed for solving the quadratic assignment problem [Maniezzo *et al*, 1994]. In their problem graph the fixed number of nodes are rule antecedents determined by a non-evolutionary approach from the training set. An ant goes round the problem graph, visiting each and every node in turn and probabilistically assigning a rule conclusion to each node/rule antecedent. Each ant solution was evaluated by inserting the rule base created (the previously determined rule antecedents plus the rule conclusion assigned them by the ant) into a fuzzy rule-based system and evaluated on test data.

The authors developed two variants of an ant algorithm which they tested against several other techniques: genetic algorithms, simulated annealing and the two algorithms developed by [Wang & Mendel, 1992] and [Nozaki *et al*, 1997] respectively. They tested their algorithm on a three-dimensional function and a real-world electric engineering application data set. The results achieved were very encouraging and they suggested the addition of local search mechanisms as a future research direction.



## Chapter 4

### Problem Investigation and Specification

The scope of this project was initially undefined so an analysis of the possibilities was made and a decision taken thereafter on the project boundaries. This chapter discusses several important features of Ant-Miner and possibilities for modifications. More strategic changes may be made to Ant-Miner and this possibility is discussed in the section on Future Work in Chapter 8. Modifications implemented as part of this project are highlighted where appropriate and more detail on this may be found in Section 5.3 Modifications of Original Ant-Miner.

Notation:

Let  $term_{ij}$  be a rule condition of the form  $A_i = V_{ij}$ , where  $A_i$  is the  $i$ -th attribute and  $V_{ij}$  is the  $j$ -th value of the domain of  $A_i$ .

Most of the following formulas are reproduced from [Parpinelli *et al*, 2002a] and [Parpinelli *et al*, 2002b].

## 4.1 Problem Dependent Elements

This section describes those elements of an ant algorithm that must be adapted to the particular problem domain. For instance, the heuristic for the travelling salesman problem previously described was the inverse of the distance between two cities. This acts as a guide in selecting the next city to be visited by an ant. The heuristic for Ant-Miner therefore needs to be pertinent to the building of a rule antecedent and hence to the building blocks in this case, i.e. terms.

Similarly with the fitness function. The aim in the travelling salesman problem is to find a closed tour between the cities of minimal length, hence the shorter the tour built by an ant the higher the quality assigned to it. In the case of Ant-Miner and rule induction we need a fitness function that assesses how well a rule constructed by an ant covers instances in the training data.

### 4.1.1 Heuristic Function

Each  $term_{ij}$  that can be added to the current rule antecedent has an associated heuristic value  $\eta_{ij}$ . This value gives an estimate of the quality of this term with respect to its ability to improve the predictive accuracy of the rule being built. Ant-Miner's heuristic function is based on the entropy of a term which is defined by:

$$H(W|A_i = V_{ij}) = - \sum_{w=1}^k (P(w|A_i = V_{ij}) \cdot \log_2 P(w|A_i = V_{ij}))$$

Equation 1 Term Entropy

where:

- $W$  is the class attribute
- $k$  is the number of class labels

- $P(w| A_i = V_{ij})$  is the empirical probability, i.e. the frequencies determined from the training set, of observing class  $w$  conditional on having observed  $A_i = V_{ij}$

The higher the term entropy the more uniformly distributed the classes are and so the lower the predictive power of  $term_{ij}$ . Terms with a higher predictive power are naturally preferred and this was taken into account by the authors of Ant-Miner when designing the normalised heuristic function:

$$\eta_{ij} = \frac{\log_2 k - H(W| A_i = V_{ij})}{\sum_{i=1}^a x_i \cdot \sum_{j=1}^{b_i} (\log_2 k - H(W| A_i = V_{ij}))}$$

Equation 2 Term Heuristic

where:

- $a$  is the total number of attributes
- $x_i$  is set to 1 if the attribute  $A_i$  is not yet used by the current ant, otherwise 0
- $b_i$  is the number of domain values of the  $i$ -th attribute

The term entropies are calculated at the beginning of each AntRun as they depend on the instances in the training set, i.e. after the training set has been reduced they need to be recalculated. The term heuristic values are recalculated each time after a term is selected for addition to the current rule antecedent being built; when this happens the term selected needs to be ignored in subsequent term selections as do other terms belonging to the same attribute of the chosen term. Since the heuristic function is a normalised one the heuristic values of the remaining terms, i.e. the terms that may still be selected by the current ant, need to be recalculated.

It should be noted that this heuristic is a *local* heuristic as it applies to individual terms and this makes it sensitive to attribute interaction. The pheromone levels of a term actually act as another heuristic in an ant algorithm, but of a more global kind. The pheromone values are

changed depending on the fitness of the rule as a whole, i.e. taking into account interaction among the attributes occurring in the rule [Freitas, 2001].

The heuristic used here comes from the decision tree world and is similar to the heuristics used in algorithms such as *C4.5*. There are literally dozens of other heuristics that may have been borrowed and used in Ant-Miner, whether they be derived from information theory (as was term entropy), distance measures or dependence measures [Ben-Bassat, 1982]. However, several comparative studies have been conducted and the general conclusion is that no particular feature selection criterion is significantly better than others, including [Baker & Jain, 1976], [Mingers, 1989] and [Breiman *et al*, 1984]. This is why it was decided to concentrate on modifying other aspects of Ant-Miner during this project.

#### 4.1.2 Fitness Function

The quality of a rule in Ant-Miner, denoted by  $Q$ , is determined by:

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

Equation 3 Rule Fitness Function

where:

- $TP$  is true positives, the number of instances covered by the rule that have the same class label as the rule,
- $FP$  is false positives, the number of instances covered by the rule that have a different class label from the rule,
- $FN$  is false negatives, the number of instances that are not covered by the rule but have the same class label as the rule, and

- $TN$  is true negatives, the number of instances that are not covered by the rule and do not have the same class label as the rule.

So, sensitivity is the accuracy among positive instances, and specificity is the accuracy among negative instances.

As in genetic algorithms, the fitness function could be made to work harder. It currently takes into account only the rule accuracy but it may be extended to consider the rule length, i.e. comprehensibility [Liu & Kwok, 2000], and even how 'interesting' it is [Freitas, 1999]. We could end up with a weighted fitness function such as:

$$Q = w1(accuracy) + w2(length) + w3(interestingness)$$

This is another interesting research avenue and much may be borrowed from existing work on comprehensive fitness functions, including the works referenced in the preceding paragraph.

## 4.2 Problem Independent Elements

The transition rule and pheromone update strategy are the more problem independent features of an algorithm. Although there are several variations for both, usually they arise not out of problem domain peculiarity, but out of a requirement to balance exploitation and exploration during solution construction, or obtain more or less search direction from pheromone updates. Since little work has yet been done in applying ant algorithms to rule induction, all the references in this section are for ant algorithms that have been applied to other optimisation problems such as the previously mentioned TSP. However, since the transition rule and pheromone updating are, after all, problem independent, then they may be readily tested on Ant-Miner.

### 4.2.1 Transition Rule

The transition rule used by Ant-Miner ants during rule construction is the same as the one used in AS for the TSP. It is called a random proportional transition rule and is given by:

$$P_{ij} = \frac{[\eta_{ij}]^\alpha \cdot [\tau_{ij}]^\beta}{\sum_{i=1}^a \sum_{j=1}^{b_i} (\eta_{ij} \cdot \tau_{ij}(t)), \forall i \in I}$$

Equation 4 Random Proportional Transition Rule

where:

- $P_{ij}$  is the probability that  $term_{ij}$  is selected to be added to the current partial rule antecedent
- $\eta_{ij}$  is the heuristic value associated with  $term_{ij}$
- $\tau_{ij}(t)$  is the amount of pheromone associated with a  $term_{ij}$  at iteration  $t$
- $a$  is the total number of attributes
- $b_i$  is the number of domain values of the  $i$ -th attribute
- $I$  are the attributes not yet used by the ant
- $\alpha$  and  $\beta$  are two adjustable parameters that control the relative weight of the heuristic and pheromone values respectively. In Ant-Miner the authors kept these fixed at 1, thereby giving them equal importance.

Therefore, a  $term_{ij}$  is chosen to be added to the current partial rule antecedent with probability proportional to the value of the above equation.

The current project implemented an additional transition rule – pseudo-random proportional transition – that allows explicitly for exploration. This was first used in [Gambardella &

Dorigo, 1995] by a family of ant algorithms they called Ant-Q. These algorithms were used with different transition rules and applied to the TSP. Algorithms applying the pseudo-random transition rule were found to outperform the others with different transition rules, including the random proportional rule. The pseudo-random proportional transition rule is given by:

$$s = \begin{cases} \operatorname{argmax} \{[\eta_{ij}]^\alpha \cdot [\tau_{ij}(t)]^\beta\} & \text{if } q \leq q_0 \\ S & \text{if } q > q_0 \end{cases}$$

Equation 5 Pseudo-Random Proportional Transition Rule

How an ant chooses a term  $s$  to add to the rule antecedent depends on  $q$ , which is a random variable uniformly distributed over  $[0,1]$ , and  $q_0$  (called q-value in later discussions) an adjustable parameter with range  $[0,1]$ . If  $q \leq q_0$  then the term with the highest proportion of heuristic and pheromone value is selected from the terms that may still be considered for inclusion. Otherwise, the selection  $S$  is made in an identical manner as for Equation 4 Random Proportional Transition Rule above. Therefore,  $q \leq q_0$  corresponds to an exploitation of the knowledge available about the problem, whereas  $q > q_0$  contributes towards more exploration. Limiting exploration by adjusting  $q_0$  allows ants to concentrate on best solutions instead of exploring constantly. The results obtained with experimenting with this transition are discussed in Section 7.3 Test 3 – Changing the Transition Rule.

#### 4.2.2 Pheromone Update Strategy

At the start of an AntRun the pheromones of all terms are initialised and given equal value – the inverse of the total number of terms.

Each time an ant completes the construction of a rule, the amount of pheromone for all terms is updated. The pheromones of terms that occur in the constructed rule  $R$  is increased in proportion to the quality,  $Q$ , of the said rule:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q, \quad \forall i, j \in R$$

Equation 6 Pheromone Increase for Terms in a Rule

The pheromone levels of *all* terms are then normalised (each pheromone level is divided by the summation of all pheromone levels), so that the end result is that terms occurring in rule *R* have their pheromone levels reinforced while other terms have their pheromone levels decreased.

There are various ways in which a change may be made to the pheromone update strategy, including the initialisation of the pheromone level and in the setting of minimum and/or maximum bounds on the levels reached. For instance, putting a bound on the maximum value of a pheromone level avoids some terms being reinforced to the point where it is impossible to make a path without them, and so favours exploration [Stülze & Hoos, 2000].

In Ant-Miner, as previously explained, iterations are run with a population of one ant. In ant algorithms where the population size is greater choices need to be made as to how to update the pheromone levels. All the ants within an iteration may be used to update pheromones, or perhaps the *n* best ants are used, or a purely elitist approach suggests that only the best ant should be used for pheromone updating, ensuring that exploration is more directed [Dorigo & Gambardella, 1997].

This last approach, of updating pheromone levels using only the best ant in an iteration was tried in this project. Naturally, Ant-Miner's population size had to be increased first. The results are discussed in Section 7.2 Test 2 – Increasing Population Size.

## Chapter 5

### System Analysis, Design and Implementation

This chapter gives a conceptual overview of the system and highlights the principal features of the implementation and differences from the original Ant-Miner algorithm. Ways of improving and extending the system are also noted.

#### 5.1 Conceptual Overview of the System

The system implemented runs a k-fold cross-validation experiment on a named pre-processed data set. The data set will already have been split into k folds, or parts, and the original version of Ant-Miner is executed by the simple command:

```
java Experiment <datasetName> <noFolds>
```

Figure 5.1 below helps to describe the overall process. More diagrams follow decomposing parts of previous diagrams (indicated by a dashed line border) to provide more detail. Boxes with sharp corners indicate processes, while boxes with rounded corners indicate data such as training data, test data and rule sets. An arrow from a process box indicates an output of that process, while an arrow into a process box indicates data required by that process.

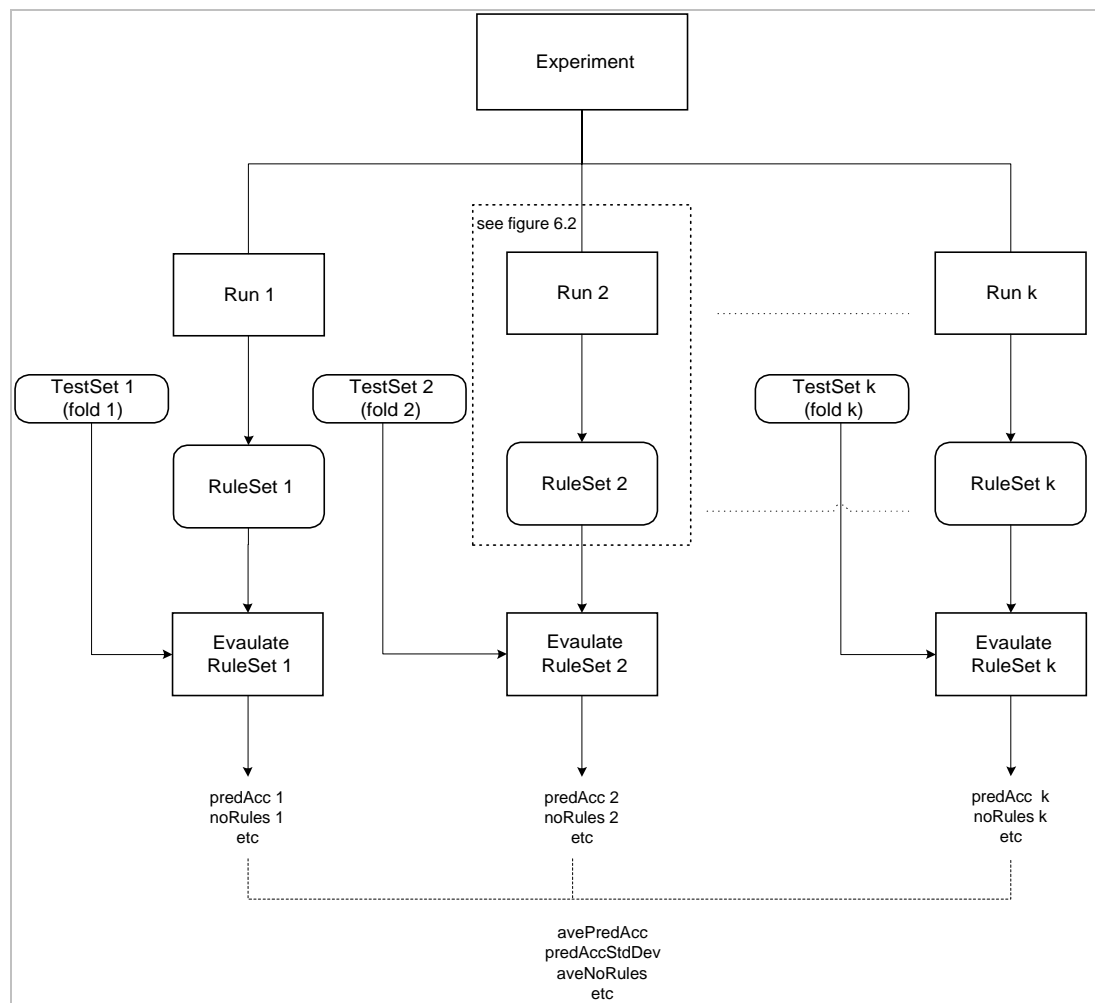


Figure 5.1 High-level view of the system

An experiment consists of generating  $k$  runs. Each of these runs uses a different fold of the data as a test set. This test set is applied to the ordered rule set generated from the remaining  $(k-1)$  folds by Ant-Miner, in order to evaluate how effective the rule set is. At the end of the run, statistics such as the predictive accuracy of the rule set and the number of rules in a rule set are calculated. At the end of the  $k$  runs these individual statistics are collated and summary statistics such as averages and standard deviations are calculated (more detail about this in Chapter 7 Results and Analyses).

The following diagram, Figure 5.2, illustrates in more detail what happens during each run.

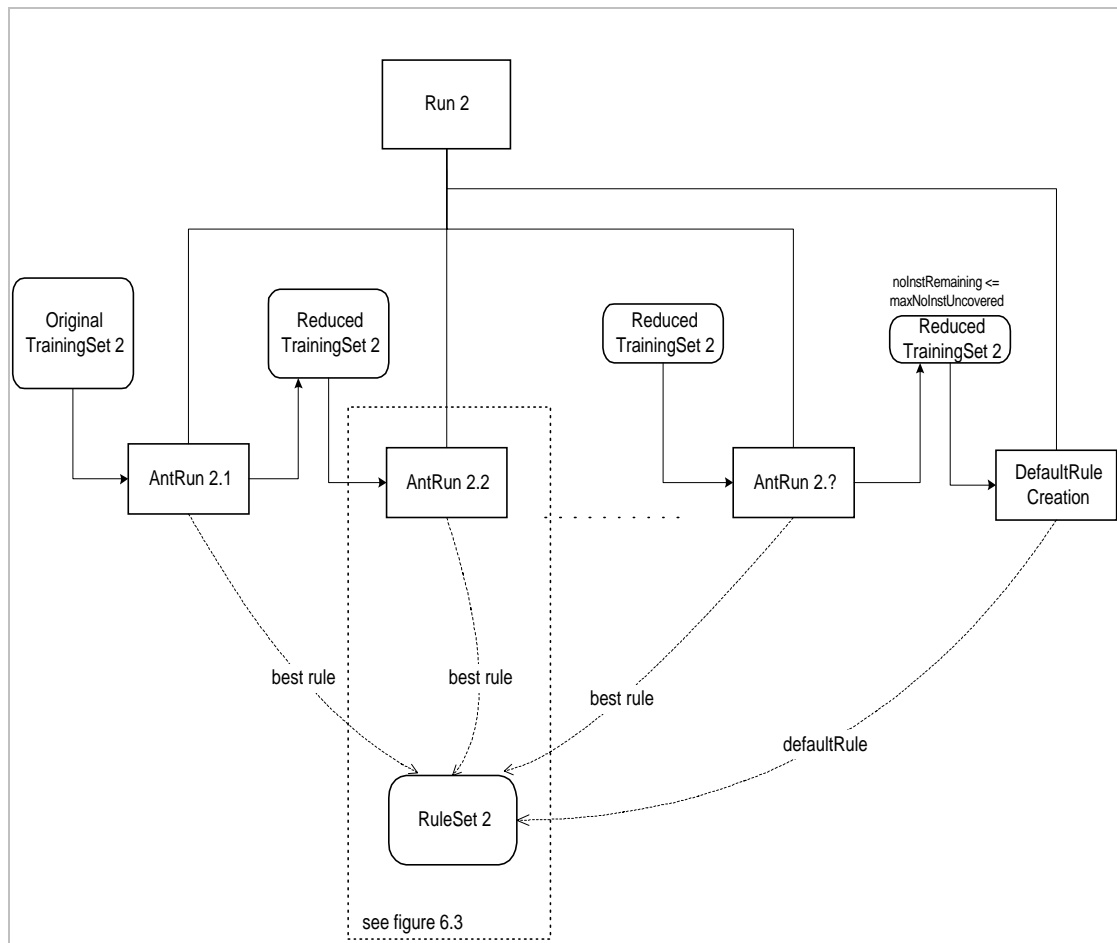


Figure 5.2 Level I view of the system – a run of a k-fold cross-validation

Each run itself generates, in succession, a number of what have been termed AntRuns. An AntRun uses the training set (the instances *not* in the fold to be used as the test set), to generate a number of if-then rules. At the end of an AntRun the best rule generated is added to the rule set for that run (RuleSet 2 in the above diagram, which Parpinelli *et al* call the discovered rule list). All instances in the training set covered by this newly added rule are removed, and the resulting reduced training set is then used by the next AntRun.

The number of AntRuns is determined dynamically based on the predetermined value of a parameter: `maxInstUncovered`, i.e. the maximum number of instances in the training set that may be left uncovered by the rules in the rule set. No more AntRuns will be generated if the remaining number of instances in the reduced training set is less than or equal to the

value of this predefined parameter. At this point a default rule is created to cover the remaining instances in the training set – the rule only has a conclusion and this is the majority class of the remaining instances.

It is worth remembering at this point that Ant-Miner creates an ordered rule set with the rules being applied to a new instance for classification in the order that they were created and added to the final rule set. The default rule is applied to an instance only if none of the previous rules in the rule set are applicable.

The next diagram, Figure 5.3, explains the creation of ants and rules by the system.

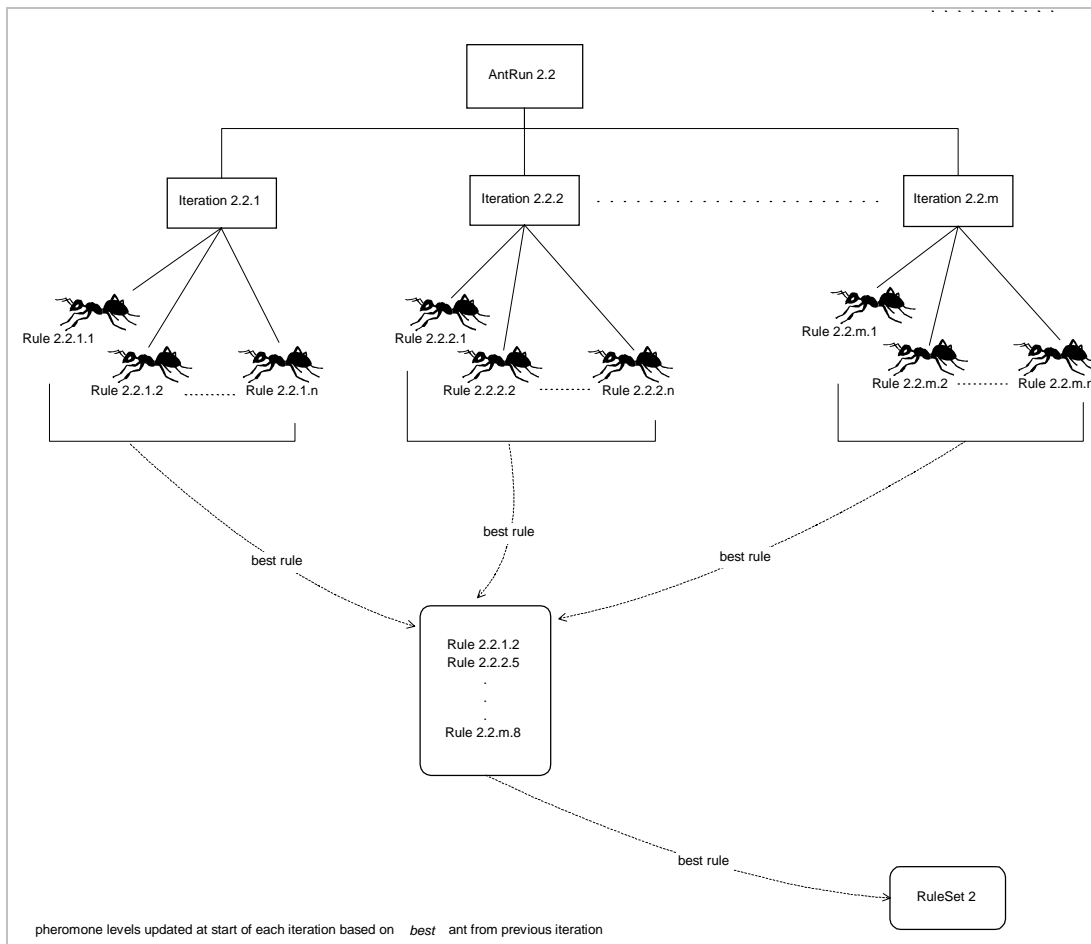


Figure 5.3 Level II view of the system - creation of ants and rules

Each AntRun executes a number of iterations in sequence. An iteration has a fixed population of ants each of which creates a single rule. After pruning, each rule is assigned a quality value based on how well it covers instances in the training set. The best rule of each iteration is saved in a temporary storage structure, *BestRulesOfIterations*, and is used to update the pheromone levels of the terms before the next iteration is initiated.

There are two user-defined parameters that control the number of iterations that may be executed by an AntRun: `maxRulesConverge` and `maxNoIterations`. The first parameter sets the value,  $p$ , for the maximum number of best rules from successive iterations that are allowed to be the same. If this value is reached, that is if the  $p$  best rules from  $p$  successive iterations are equal, then the current AntRun will be halted. At this point the best rule from `BestRulesOfIterations` is selected and added to the final rule set for the run.

The `maxNoIterations` parameter controls the maximum number of iterations that may be executed if the `maxRulesConverge` value is not reached during that AntRun.

## 5.2 Physical Overview of the System

The main goals during the system analysis, design and implementation phase were computational efficiency and future flexibility. Object-oriented analysis and design principles were followed, and the system was implemented in the Java programming language (SDK version 1.3.1).

Computational efficiency is reflected mainly in the discussions on stopping criteria for AntRuns, Section 5.3.1 on page 42, and extreme pheromone levels, Section 5.3.3 on page 44.

Future flexibility is reflected in the design of the system implemented. During this project the principal aims were to investigate the balance between exploration and exploitation, and the effect of more discriminatory term pheromone updating in Ant-Miner. However, as

analysed earlier in Chapter 4 Problem Investigation and Specification, there are other features of Ant-Miner that may be changed and investigated, such as the fitness function and the local heuristic method. This was taken into consideration during the design and implementation of the system, and is highlighted in Figure 5.4 High-level view of system implementation.

Figure 5.4 provides a high-level view of the implemented system. It shows all inheritance and composition relations and the principal other associations between the implemented classes. For the purpose of clarity, not all fields and methods are shown and Java library classes such as `BufferedReader`, `BufferedWriter` and `Random` are also omitted.

Abstract classes (name of class in italics) have been implemented for the parts of Ant-Miner that may be readily changed – the transition rule, fitness function, heuristic method and the pheromone updating strategy.

Concrete subclasses of these abstract classes then define the actual techniques that are employed by Ant-Miner, denoted on the diagram by `Ant-MinerHeuristic`, for instance, or `Ant-MinerPheromone`. For the purpose of illustrating the flexible design for implementing different heuristic methods or pheromone strategies another subclass has been shown labelled `OtherMethod`. Testing Ant-Miner with different heuristics, fitness functions, pheromone update strategies or transition rules is simply a matter of defining a new subclass of the appropriate abstract class.

This is what has already been done for the transition rule – Ant-Miner's transition rule, generally called a random proportional transition rule has been implemented by the subclass `RandomProp(Ant-Miner)`, while a different transition rule called pseudo-random proportional transition has been implemented by the subclass `PseudoRandProp`. By default the system will use Ant-Miner's original transition rule but the user can change that by inputting an over-riding option on the command line.

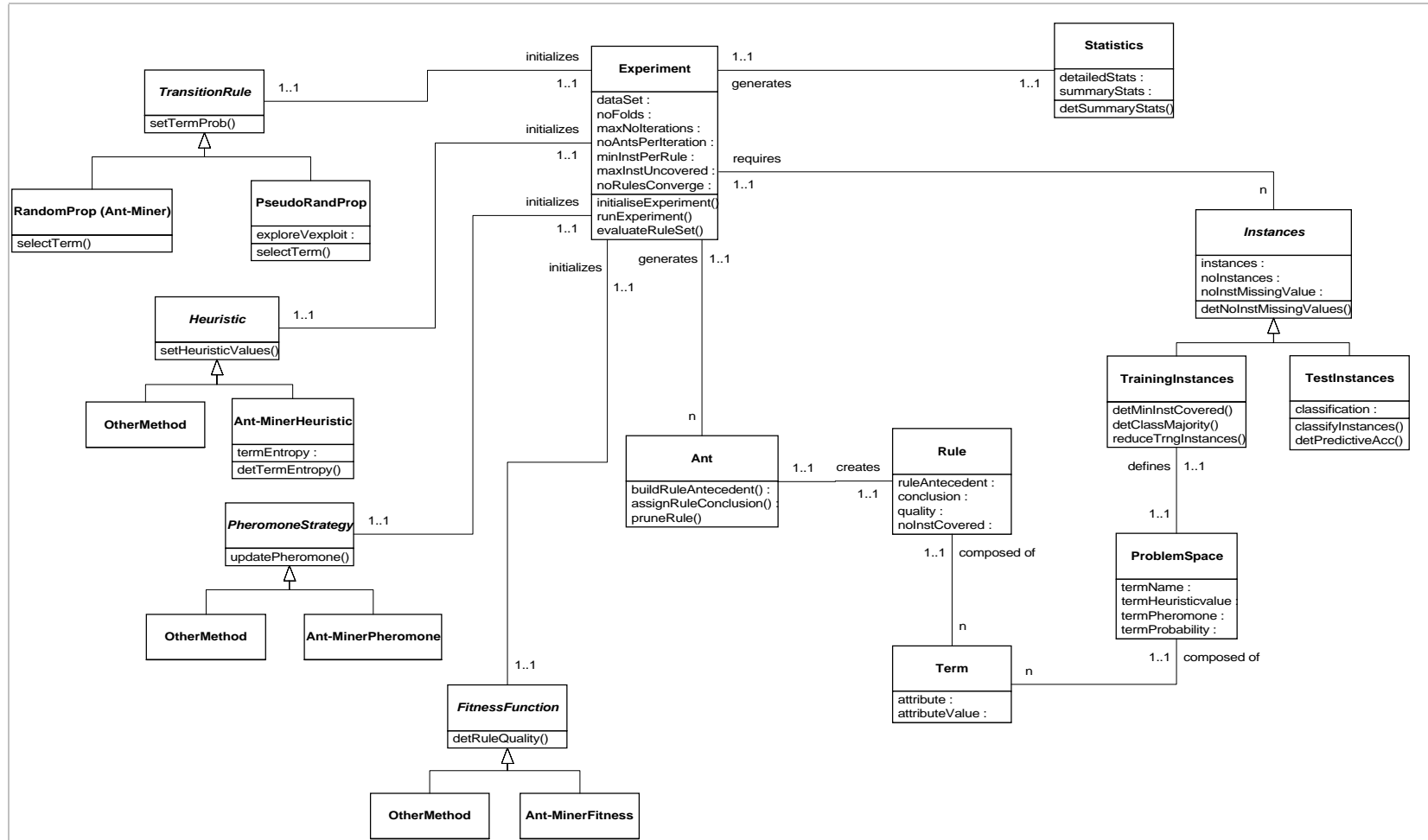


Figure 5.4 High-level view of system implementation

## 5.3 Modifications of Original Ant-Miner

Detailed below are the known differences between the original Ant-Miner algorithm and this implementation.

### 5.3.1 Stopping Criteria for AntRuns

As mentioned previously (Section 5.1, Figure 5.2), the number of AntRuns is determined dynamically based on the number of instances in the training set still uncovered by any of the previously generated rules. However, an additional stopping criterion for the number of AntRuns has also been implemented.

Occasionally, towards the end of a run, the best rule generated by an AntRun has a quality value of zero, either because the sensitivity or the specificity part of the fitness function is zero. This happens when the instances remaining in the training set are few and disparate, and the minimum number of instances that must be covered by any rule is generally too great at this point to generate useable rules.<sup>1</sup>

Therefore, if the best rule of the current AntRun has quality equal to zero, though the `maxInstCovered` criteria may not yet be satisfied, no more AntRuns are generated, and the default rule to cover the remaining instances in the training data is created. (The 'best' rule is *not* added to the final rule set.)

The consequence is that the overall performance of the rule set *may* be adversely affected as the default rule will cover too many disparate instances. If, on the other hand, more AntRuns are generated until the `maxInstCovered` criterion is satisfied, then it is possible that eventually, one or more useful rules are produced before the default rule is finally created.

---

<sup>1</sup> This was confirmed also via email by the principal author of Ant-Miner, Dr Rafael Parpinelli.

This would, in turn, mean a longer running time for the system since most of the AntRuns will be generating purely unuseable rules (and it is quite possible that *no* rule generated is able to meet all the user defined parameter values at some point).

The results of Test 1 – Comparison with Ant-Miner, detailed in Chapter 7 discusses the impact of this modification further.

### 5.3.2 Ant Population Size

A major difference between this implementation and the original Ant-Miner algorithm is illustrated in the following diagram. Please also refer to Figure 5.3 on page 36 for comparison.

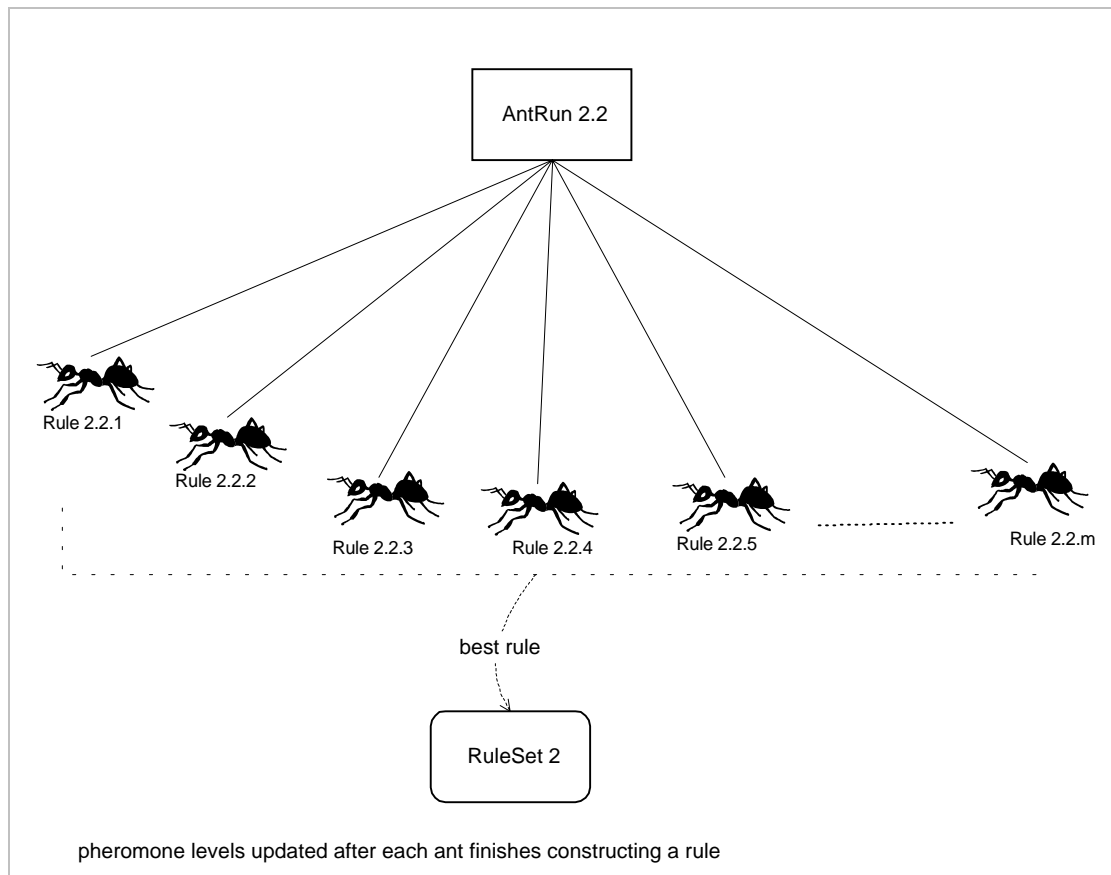


Figure 5.5 Creation of ants and rules in the original Ant-Miner algorithm

Figure 5.5 above highlights the fact that the original Ant-Miner algorithm does not include the notion of an iteration layer as such. Ant-Miner runs a number of ants and the `noAnts` parameter replaces the previously mentioned `maxNoIterations` parameter. The stopping conditions for the number of ants created is similar to the stopping conditions for iterations mentioned previously – if  $p$  ants in succession generate the same rule then the current AntRun will halt. Otherwise, the AntRun halts as soon as it has created a previously determined maximum number of ants.

Note that in Ant-Miner the pheromone levels of terms are updated after the creation of each ant, while in this implementation the pheromone levels are updated once at the start of each iteration, based on the *best* ant from the previous iteration. This makes the pheromone updating of terms more discriminatory and therefore provides more direction to the search of ants in successive iterations.

The system implemented can, however, reproduce the same pheromone updating strategy as in Ant-Miner - this is achieved by setting the ant population size for each iteration equal to one.

The system parameters are listed in Appendix A User-Defined System Parameters.

### 5.3.3 Pheromone Levels

Towards the end of the system development phase it was discovered that the pheromone levels for some terms in some of the data sets were getting exceptionally low. This caused some difficulties in the execution of the program when the values went beyond the range for Java `doubles` – double-precision, 64-bit format IEEE 754 floating-point values with a range of  $4.9406564581246544E-324$  to  $1.79769313486231570E+308$ .

Several solutions were possible including:

- the use of a specific Java class (`BigDecimal`) that would enable the system to deal with exceptionally small values. This would be extremely costly in terms of execution time;

- setting a minimum value for the pheromone levels of terms, in a similar manner to that of [Stülze & Hoos, 2000], where both a minimum and a maximum value for pheromone levels was set;
- ignore terms with such small pheromone levels, i.e. do not consider them for inclusion in the rule being built by the current ant.

Dr Parpinelli confirmed that he had taken the first approach and Ant-Miner does actually handle extremely small values for pheromone levels.

However, his work also emphasises that in most cases an AntRun stops because the `maxRulesConverge` criterion is satisfied long before the stopping criterion of the maximum number of 3000 ants is reached. Remember that in this implementation the relevant parameter is `maxNoIterations`, 3000, each with a population of one ant. The decision was therefore made that for the purpose of this project the `maxNoIterations` setting for some of the data sets would be reduced in a way that would ensure that the pheromone levels would not go beyond Java's limitations on double-precision floating-values.

Another alternative would have been to change the setting for the `maxRulesConverge` criterion from 10 to a lower value. Upon a detailed investigation of when an AntRun tried to generate the maximum 3000 iterations, it was found that one of two cases had arisen:

- there were too few instances remaining in the training set and the AntRun was finding it difficult to generate a rule with a quality above the value of zero (this was discussed in Section 5.3.1 - Stopping Criteria for AntRuns). At this point the default rule is created for the current rule set; or
- as was generally the case when pheromone levels were too low, the same few good rules were being generated by the iterations of an AntRun. This would cause the pheromone levels of terms in these rules to increase but the pheromone levels of all other terms to decrease due to evaporation. The same rules were being created over and over but never with one of them being generated 10 times in succession (3, 4, 5, 6 even 7 times in

succession but not 10), at which point the `maxRulesConverge` criterion would be satisfied, halting the `AntRun`, outputting the best rule for inclusion in the rule set, and starting a new `AntRun` on the reduced training set.

Since test results indicate that an excessive number of iterations do not add a great benefit to a generated rule set and its generalisation power, it is likely that the system will be modified at a later stage to ensure that pheromone levels do not fall below a predetermined minimum amount. This point is further discussed in Chapter 8 Conclusions and Future Work.

### 5.3.4 Rule Pruning Procedure

The rule pruning procedure implemented may be slightly different from Ant-Miner's (to be confirmed with the author at a later date).

As described in [Parpinelli *et al*, 2002a] Ant-Miner's rule pruning procedure outputs a shorter rule if the new rule has an *improved* quality. If a shorter rule with an improved quality is not possible, the implemented rule pruning procedure will output a shorter rule with an *equal* quality if possible. This may be the reason why the system implemented for this project in general produces rule sets with shorter rules (see the results of Test 1 – Comparison with Ant-Miner, page 58 for more details).

Another point that needs clarification is whether the `minInstCovered` criterion should still be satisfied during the rule pruning procedure. While a rule is being created by an ant this parameter setting determines the minimum number of training instances that must be covered by the rule. Before a rule is pruned the implementation ensures that this criterion is satisfied. However, a pruned rule, though it may be shorter and have an improved quality, in this implementation it may also end up covering fewer instances than the `minInstCovered` parameter setting requires. Again, this may account for the fact that this implementation occasionally produces rule sets with a larger number of rules than the original Ant-Miner published results.

A sample output of a ten-fold cross-validation test on one of the data sets may be found in Appendix B Sample Outputs from Implemented System.



## Chapter 6

### Experiment Preparation

This chapter describes the approach taken to test the modifications made to the Ant-Miner algorithm, the data sets used and the pre-processing done on the data sets.

#### 6.1 Experimental Methodology

All modifications to the Ant-Miner algorithm are evaluated empirically on benchmark data sets. Section 6.2 below describes the data sets used.

The published Ant-Miner performance results, [Parpinelli et al, 2002a], were obtained using a ten-fold cross-validation. In a k-fold cross-validation the data set is randomly shuffled and split into k approximately equally-sized mutually exclusive subsets. Each of the k subsets is used once for testing while the other (k-1) subsets are used for training. This produces k individual sets of performance statistics – such as predictive accuracy and number of rules in a rule set – that are averaged to obtain the final estimates. Standard deviations for the individual performance statistics may also be obtained. There are empirical and theoretical results that support the choice of k=ten as producing the most reliable estimates, on average, of a classifier's true performance ([Kohavi, 1995] and [Kearns, 1996] respectively).

For the purpose of comparing this implementation with the Ant-Miner results (Test 1 – Comparison with Ant-Miner), and with the later modifications made to the implementation (Test 2 – Increasing Population Size through to Test 5 – Use of Reduced Data Sets), *ten* ten-fold cross-validations were run. The averages from each of the ten ten-fold cross-validations were then averaged to obtain the final statistic that appears in the tabulated results of Chapter 7 - Results and Analyses. This provides a more reliable estimate of the implementation's performance.

Appendix C Sample Detailed Test Results shows results of all ten ten-fold cross-validations for Test 3 – Changing the Transition Rule, for the Ljubljana Breast Cancer data set.

Since the algorithm implemented is based on a stochastic process and the results produced therefore vary from one ten-fold cross-validation to the next, the same folds were used for each of the ten ten-fold cross-validation tests. That is, the data set was not re-shuffled and split into  $k$  different subsets before each of the ten ten-fold cross-validation runs. This was done in order to test the deviation in the performance statistics arising from the implemented algorithm, and not due to any changes in the folds used.

For a similar reason, these same folds were used in all the tests made (i.e. Test 1 through to Test 5), as the aim was to test changes arising due to the modifications made to the algorithm, and not due to different folds used between tests. Apart from Test 4 – Handling Missing Attribute Values, none of the folds of the data sets contained instances with missing attribute values.

The results obtained from the tests are tabulated, presented graphically and discussed in Chapter 7 - Results and Analyses.

## 6.2 Data Sets

For a fair comparison between this implementation and Ant-Miner, the same data sets that Ant-Miner was originally tested on were used.

The data sets were downloaded from the website of the University of California at Irvine Machine Learning Repository<sup>2</sup>. Special thanks go to M Zwitter and M Soklic, University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia, for the use of their breast cancer database.

Five of the six data sets used are from the medical domain whilst the sixth, Tic-Tac-Toe, encodes the complete set of possible board configurations at the end of a tic-tac-toe game. Four of the data sets have binary class labels whilst the other two represent multi-class domains. Half of the data sets have a mixture of nominal and continuous attributes, and the other half have only nominal attributes. Table 6.1 below lists the main properties of the data sets.

Data Set	Total # Instances	Instances w. Missing Values	Missing Values (%)	Categorical Attributes	Continuous Attributes	Classes
Ljubljana Breast Cancer	286	9	0.3	9	-	2
Wisconsin Breast Cancer	699	16	0.3	9	-	2
Tic-Tac-Toe	958	0	0.0	9	-	2
Dermatology	366	8	0.1	33	1	6
Hepatitis	155	70	5.4	13	6	2
Cleveland Heart Disease	303	6	0.2	8	5	5

Table 6.1 Data Set Properties

The first column gives the total number of instances in the data set, the second the number of instances with one or more missing attribute values, the third the proportion of terms with missing attribute values, the fourth the number of categorical attributes (excluding the conclusion attribute), the fifth the number of continuous attributes (also excluding the conclusion attribute), and the final column gives the number of class labels.

<sup>2</sup> <http://www.ics.uci.edu/~mlearn/MLRepository.html> etc

## 6.3 Data Pre-Processing

The computer algebra system *Mathematica* (v4.2) was used for the random shuffling and splitting of the data sets into ten folds for use in a ten-fold cross validation experiment. The following sub-sections describe the discretization of continuous attributes necessary for all the Tests conducted, and the feature subset selection required for Test 5.

### 6.3.1 Discretization of Continuous Attributes

Ant-Miner generates rule sets from data sets that have only nominal attributes. Three of the data sets used have a mixture of both nominal and continuous attributes and therefore a discretization process is necessary for the continuous attributes. In [Parpinelli *et al*, 2002a] the discretization was carried out using a method called *C4.5-Disc* ([Kohavi & Sahami, 1996]).

*C4.5-Disc* is an entropy-based method that applies the decision-tree algorithm *C4.5* to obtain a discretization of the continuous attributes. Each continuous attribute is taken separately with the conclusion attribute to form a reduced data set. *C4.5* is applied to this reduced data set and the nodes of the induced decision-tree are used as threshold values for a discretization of that continuous attribute.

*C4.5* applies pruning before outputting the final decision tree. In their original work, referenced above, Kohavi and Sahami changed the default pruning confidence value from 25% to 1% in order to produce fewer discretization intervals. Discussion with Dr Parpinelli confirmed that in their work they used a pruning confidence value of 1% and this same value was used for discretization of continuous features during this project. Both in [Parpinelli *et al*, 2002a] and in this project the discretization of continuous features was carried out before the final nominal-only data set was split into ten folds for a ten-fold cross-validation.



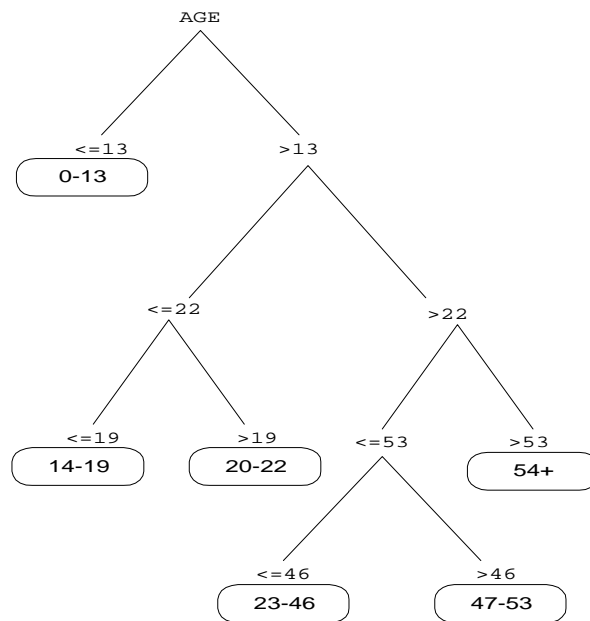


Figure 6.2 Sample discretization resulting from *C4.5-Disc*

Sometimes, *C4.5* can determine no association between the values of the continuous attribute and the associated conclusion values. In this case it outputs a decision tree with just one leaf, i.e. only one discretization interval is produced. All values of this continuous attribute are therefore changed to the same value in the data set before being inputted into the implemented system for a ten-fold cross-validation.

### 6.3.2 Attribute Subset Selection

Test 5 – Use of Reduced Data Sets - was carried out on data sets with a reduced number of attributes.

The program used for attribute reduction was developed by Richard Jensen, University of Edinburgh, and was based on the work in [Jensen & Shen, 2002b]. It is fuzzy-rough set approach that works on both continuous and nominal-valued attributes to find a subset of the original attributes called a *reduct*. All other attributes may be removed from the data set with minimal information loss.

A sample output from the program follows, showing the *reducts* produced in the cases for the Dermatology data set.

```
eddie[Richard_FeatureSelect] java rsar Dermatology_C
<== RSAR with bailing ==>
35 attributes in dataset
366 object in dataset

Choosing {21} ==> 0.24863388
Choosing {21, 26} ==> 0.44535518
Choosing {4, 21, 26} ==> 0.5464481
Choosing {4, 14, 21, 26} ==> 0.6912568
Choosing {3, 4, 14, 21, 26} ==> 0.7568306
Choosing {3, 4, 14, 21, 26, 30} ==> 0.8770492
Choosing {3, 4, 14, 15, 21, 26, 30} ==> 0.92076504
Choosing {3, 4, 14, 15, 21, 26, 30, 31} ==> 0.97540987
Choosing {3, 4, 14, 15, 21, 26, 30, 31, 33} ==> 0.9918033
Choosing {3, 4, 14, 15, 17, 21, 26, 30, 31, 33} ==> 1.0
QuickReduct is {3, 4, 14, 15, 17, 21, 26, 30, 31, 33, 34}
```

Figure 6.3 Reduct for Dermatology data set

The table below shows the difference between the original and the reduced number of attributes. The numbers do not include the conclusion attribute. This table will be discussed further when the results of Test 5 – Use of Reduced Data Sets, are analysed.

<b>Dataset</b>	<b>Original # Attributes</b>	<b>Reduced # Attributes</b>
Ljubljana Breast Cancer	9	8
Wisconsin Breast Cancer	9	5
Tic-Tac-Toe	9	8
Dermatology	34	11
Hepatitis	19	11
Cleveland Heart Disease	13	10

Table 6.2 Original vs Reduced Number of Attributes in Data Sets



## Chapter 7

### Results and Analyses

As mentioned in Section 6.1 Experimental Methodology, in order to obtain reliable performance estimates ten ten-fold cross-validations were carried out to produce each of the statistics in the tables below for this project.

As some of the tests required changes to a parameter setting for each of the data sets (e.g. `noAnts` value was changed in Test 2 – Increasing Population Size, and the `q-value` was changed during Test 3 – Changing the Transition Rule) this resulted in over 1000 ten-fold cross-validations. The results obtained are summarised and discussed below. Possible follow-up investigations suggested by the results are also mentioned where appropriate.

Parpinelli *et al* used two sets of evaluation criteria for comparison with other rule induction algorithms (CN2 and C4.5). Those same criteria are used here for comparison between Ant-Miner and this implementation, and the modifications made to this implementation. The first criterion is predictive accuracy. The second criterion comprises rule set size (the number of rules in a rule set), and individual rule size (the number of terms in a rule).

Unless otherwise stated, the tests were run on similar machines with the following specification:

Manufacturer: Sun (Sun Microsystems)  
 System Model: Blade 100  
 Main Memory: 128MB  
 ROM Version: OBP 4.0.45  
 No of CPUs: 1  
 CPU Type: 502 MHz SUNW, UltraSPARC-IIc

## 7.1 Test 1 – Comparison with Ant-Miner

The purpose of this test was to make a basic comparison between the results produced by the implemented system and those reproduced from [Parpinelli *et al*, 2002a] for the Ant-Miner algorithm. With the exceptions noted further below, the parameter settings used for this test are as defined in [Parpinelli *et al*, 2002a] and listed in Appendix A User-Defined System Parameters.

Table 7.1 and Figure 7.1 below compare the predictive accuracy and associated standard deviation.

Dataset	Original Ant-Miner		This Implementation	
	Predictive Accuracy (%)	Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
Ljubljana Breast Cancer	75.42	10.99	72.53	7.55
Wisconsin Breast Cancer	96.04	2.80	92.22	3.35
Tic-Tac-Toe	73.04	7.60	74.20	4.43
Dermatology *	86.55	6.13	89.84	6.19
Hepatitis *	90.00	9.35	84.33	10.93
Cleveland Heart Disease *	59.67	7.52	56.00	8.48

Table 7.1 Comparing Predictive Accuracy with Ant-Miner

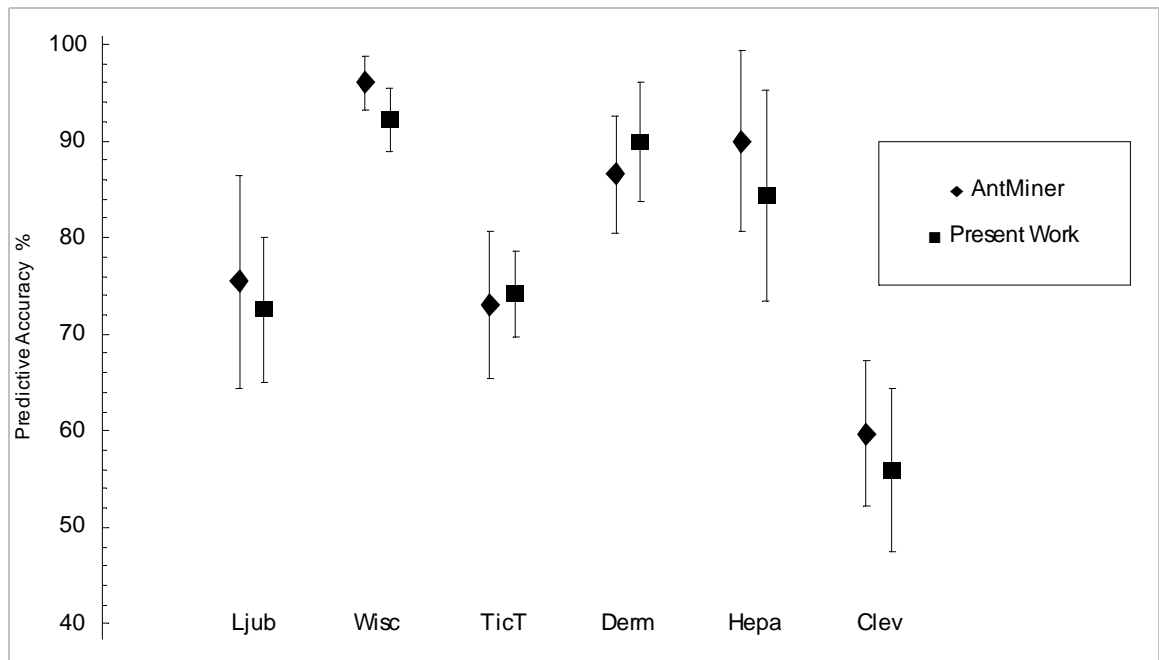


Figure 7.1 Comparing Predictive Accuracy with Ant-Miner

On the whole this implementation produces rule sets with slightly less generalisation power than Ant-Miner, the exceptions being the Dermatology data set where an average predictive accuracy of 89.84% was achieved by this implementation as opposed to Ant-Miner's 86.55%, and the Tic-Tac-Toe data set with 74.01% predictive accuracy as opposed to 73.04%.

It should be remembered at this stage that although great care has been taken to make this implementation as faithful as possible to Ant-Miner there are known differences, and possibly as yet unknown differences arising from the physical implementation. The known differences have been detailed in Section 5.3 Modifications of Original Ant-Miner. The modification discussed in Section 5.3.1 Stopping Criteria for AntRuns, may have influenced the predictive accuracy and therefore may partly account for the overall lower predictive accuracy obtained.

Apart from the number of ten-fold cross-validations run to obtain the estimates in the above table, another factor that may account for the discrepancy in predictive accuracy is the data set folds. Since the folds created for these tests were randomly created it is extremely

improbable that they were the same folds used by Parpinelli *et al* when conducting their own experiments. The differences in the folds used will result in some corresponding discrepancy in the predictive accuracy.

One modification to a parameter setting for practicality purposes seems to have had no detrimental effect on the predictive accuracy. As discussed in Section 5.3.3 Pheromone Levels, the pheromone levels of some terms for some of the data sets (marked with an \* in table above) became exceptionally small, causing difficulties in the calculation of the term probabilities. To temporarily resolve this issue the number of iterations were reduced from 3000.

For the Dermatology data set the maximum number of iterations was set at 1000, and yet the resulting predictive accuracy achieved was better than that obtained by Ant-Miner.

For the Hepatitis data set the figures in the above table were obtained with a maximum number of iterations set at 1000 – resulting in an average predictive accuracy of 84.33%. An additional estimate for predictive accuracy was obtained after another ten ten-fold cross-validations, this time with the maximum number of iterations set at 1500 – 81.28%.

In the case of the Cleveland Heart Disease data set the above estimate of 56.00% was obtained with a maximum number of iterations of 1500. The slightly lower estimate of 55.94% was obtained after running another ten ten-fold cross-validations with a maximum number of iterations of 2000.

There is no strong case arising from the preceding three paragraphs for believing that the greater the number of iterations (and therefore the more information from pheromone updates that the later ants obtain), the *lower* the predictive accuracy.

However, they do show that more information from pheromone updates need *not* result in better rules and rule sets. What may be happening when the number of iterations is too great is that at a later stage ants get caught up in local minima; by this time the pheromone levels of some terms may have been reinforced so much that it makes it virtually impossible for

later ants to select seemingly 'less useful' terms and therefore accidentally find a path that takes it to a global maximum. The same few rules get generated over and over with little likelihood that a new improved rule gets created. (An analysis of what happens when an AntRun actually tries to generate 3000 iterations was provided in Section 5.3.3 Pheromone Levels, page 45.)

Table 7.2 and Figure 7.2 below compare the rule sets and rules.

Dataset	Original Ant-Miner			This Implementation		
	No. Rules in Ruleset	Standard Deviation (+/-)	No. Terms in Rule	Average No. Rules in Ruleset	Average Standard Deviation (+/-)	Average No. Terms in Rule
Ljubljana Breast Cancer	7.20	0.60	1.36	8.23	0.90	1.07
Wisconsin Breast Cancer	6.20	0.75	1.97	11.51	0.86	1.01
Tic-Tac-Toe	8.50	1.86	1.18	8.12	1.52	1.26
Dermatology	7.00	0.00	11.57	7.40	0.54	2.83
Hepatitis	3.40	0.49	2.41	3.98	0.06	1.51
Cleveland Heart Disease	9.50	0.92	1.71	14.82	0.97	1.84

Table 7.2 Comparing Rule Sets and Rules with Ant-Miner

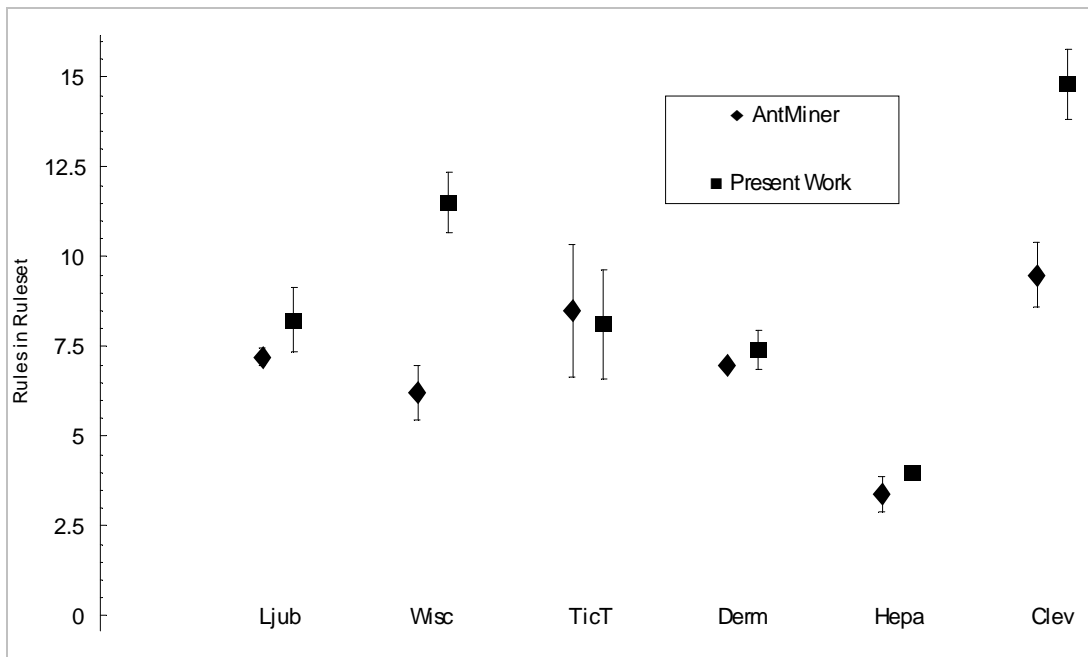


Figure 7.2 Comparing Rule Sets with Ant-Miner

Both the table and graph above indicate that in two cases in particular the number of rules generated by this implementation is greater than the number of rules generated by Ant-Miner – the Wisconsin Breast Cancer data set with 11.51 rules on average vs Ant-Miner's 6.20, and the Cleveland Heart Disease data set with 14.82 rules vs 9.50.

This may well be due to the rule pruning procedure implemented and discussed in Section 5.3.4 Rule Pruning Procedure. It is not known whether a rule outputted by Ant-Miner's pruning procedure should continue to satisfy the condition that it covers a minimum number of training instances as specified by the `minInstCovered` parameter. The rule pruning procedure implemented for the current project does *not* have to maintain this condition and this may be why on average the system generates rule sets with a larger number of rules. If this is the case, then it may partly account for the discrepancy in predictive accuracy – the rules that cover a smaller number of instances may be prone to over-fitting of the training data.

However, it may also be seen from comparing columns four and seven in the above table (No of Terms in Rule), that in four out of the six data sets this implementation produces shorter rules. The exceptions are the Tic-Tac-Toe and Cleveland Heart Disease data sets. It may be argued that the shorter the rule the more comprehensible it is.

It is worth noting that Parpinelli *et al* have revised some of their results published in [Parpinelli *et al*, 2002a] and reproduced in the above tables. These revisions have been made in [Parpinelli *et al*, 2002b], an extended version of the previous paper that is to appear in a special issue of the *IEEE Transactions of Evolutionary Computation*, 2002. A brief comparison between this implementation's performance and the revised Ant-Miner figures is in Appendix D Comparison with Revised Ant-Miner Results.

Within the limitations already discussed in this section, this test indicates that this project's implementation achieves a performance comparable with that attained by the original Ant-Miner algorithm. In order to better judge the later modifications made to the implementation and to the data sets, the results of Tests 2 to 5 are compared with this implementation's

results from Test 1 and listed in Table 7.1 and Table 7.2 above (and not with the original Ant-Miner estimates).

## 7.2 Test 2 – Increasing Population Size

This was an exploratory test to see whether a change in population size from Ant-Miner's single ant has an effect on the predictive accuracy. Lack of time prevented the processing and analysis of rule-set and rule sizes. However, this data is available and an initial assessment of the results indicate little or no impact on these factors.

The following parameter settings were changed: `maxNoIterations` and `noAnts`. As `noAnts` was increased from 1 to 10, 20, 30, 40 and 50, `maxNoIterations` was reduced from 3000 to 300, 150, 100, 75 and 60 respectively. This made the maximum total number of ants created equal to Ant-Miner's original 3000, but reduced the number of times a pheromone update was effected (equal to the `maxNoIterations`). The aim was to observe the pure effect of a change in population size for an iteration, as far as possible, and see whether such an increase with fewer but more discriminatory pheromone updates produced comparable results for predictive accuracy.

Note that since there were fewer iterations the pheromone levels for the 'problematic' data sets from the previous test – Dermatology, Hepatitis and Cleveland Heart Disease – remained within the range permitted by Java `doubles`. That is, the settings for these three data sets are identical to those of the other data sets in this test.

The following two tables show the results obtained for all of the data sets, and compare them with the result obtained by this implementation with an ant population size of 1. The best predictive accuracy achieved is shown in bold and italics.

	Ljubljana Breast Cancer		Wisconsin Breast Cancer		Tic-Tac-Toe	
	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
Pop1*	72.53	7.55	<b>92.22</b>	3.35	<b>74.20</b>	4.43
Pop10	74.63	8.78	91.59	3.22	72.54	5.57
Pop20	74.47	8.93	91.48	3.50	72.37	5.51
Pop30	74.23	8.73	91.98	3.65	72.30	5.68
Pop40	<b>74.83</b>	9.13	91.58	3.77	72.40	5.33
Pop50	74.79	8.97	91.55	3.63	71.87	5.97

(a) – Ljubljana Breast Cancer, Wisconsin Breast Cancer, Tic-Tac-Toe Data Sets

	Dermatology		Hepatitis		Cleveland Heart Disease	
	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
Pop1*	89.84	6.19	84.33	10.93	56.00	8.48
Pop10	95.38	3.35	85.65	10.12	56.79	8.63
Pop20	95.12	3.70	85.20	10.65	56.50	8.47
Pop30	95.00	3.91	85.88	9.73	<b>56.84</b>	8.80
Pop40	<b>95.65</b>	3.35	85.60	10.23	56.53	8.28
Pop50	95.52	3.45	<b>86.30</b>	10.27	56.43	8.27

(b) – Dermatology, Hepatitis, Cleveland Heart Disease Data Sets

Table 7.3 (a) and (b) - Comparing Predictive Accuracy with Increase in Population Size

\* Note that Pop1 means a population size of one ant and these figures are taken from the results obtained for Test 1, Table 7.1.

In two of the data sets – Wisconsin Breast Cancer and Tic-Tac-Toe – the best average predictive accuracy was obtained with Ant-Miner's original single ant. In the other four data sets improvement was seen when the population size was increased, particularly in the case of the Dermatology data set where the average predictive accuracy of 89.94% using a single ant increased to 95.65% when a population size of 40 ants was used.

It is suggested that the setting of this parameter – population size – may need to be determined experimentally and depends at least partly on the data set. A quick and incomplete survey of the rules created *within* an iteration indicates that for the Wisconsin and Tic-Tac-Toe data sets in particular the ants were creating identical rules during the early

iterations. Choosing the 'best' rule from those created within an iteration became a superficial and pointless task. This suggests that the additional iterations, and hence pheromone updates, were essential to these data sets so that later ants could create different rules that might also have an improved quality.

It would be interesting to try forcing the ants within an iteration to explore more and therefore to create different rules from one another. This may be accomplished by the implementation of a local pheromone update rule, as described in [Dorigo & Gambardella, 1997]. During an iteration, when one ant selects a term, that term's pheromone level would be *decreased*, thereby making it more likely that the other ants in the same iteration select different terms and hence create different rules. More detailed investigation is required, however.

One thing to note is that when the population size is increased so is the time taken. The table below gives the averages for elapsed time over ten ten-fold cross-validation for each of the various population sizes. As far as possible the tests were run on similar machines with comparable speed and memory and when no other jobs were being processed, but these figures can only be suggestive at this stage.

DataSet	Elapsed Time (mins)					
	Pop1	Pop10	Pop20	Pop30	Pop40	Pop50
Ljubljana Breast Cancer	0.0	1.0	2.0	2.9	3.9	-
Wisconsin Breast Cancer	2.5	6.4	12.7	18.9	25.2	-
Tic-Tac-Toe	4.8	5.8	9.6	12.2	15.2	-
Dermatology	123.0	524.3	507.1	501.5	489.5	-
Hepatitis	2.9	7.5	9.0	9.6	10.6	10.5
Cleveland Heart Disease	2.3	7.8	14.5	21.1	26.6	-

Table 7.4 Comparing Elapsed Time with Increase in Population Size

Note: 0 minutes means less than 1 minute; the column for time for a population size 50 is left mainly blank as these experiments were run on a slower machine than the machines used for population size 1, 10, 20, 30 and 40, with the exception of the Hepatitis data set.

### 7.3 Test 3 – Changing the Transition Rule

This was an exploratory test to see whether a change in the transition rule from random proportional selection to pseudo proportional selection has an effect on the predictive accuracy. Lack of time prevented the processing and analysis of rule-set and rule sizes. However, this data is available and an initial assessment of the results indicates that these factors have little or no impact.

The default setting for Ant-Miner's random proportional transition rule was over-ridden for this test. Instead, a pseudo-random proportional transition rule was used and this required the setting of a parameter – termed the  $q$ -value – which fixes the balance between exploitation and exploration for an ant as it is building a rule, as explained in Section 4.2. All other parameter settings remained the same as for Test 1, including the reduced number of iterations for the data sets Dermatology, Hepatitis and Cleveland Heart Disease.

Eleven different settings for the  $q$ -value were tried, ranging from 0.0, 0.1, 0.2, ... , to 1.0. A setting of  $q=0.0$  means no exploitation of learnt information and is, effectively, the same as using a random proportional transition rule. A setting of  $q=1.0$  turns the implementation into a purely-deterministic algorithm where the term with the highest probability is *always* chosen. Values in between combine different proportions of exploration of the problem space with an exploitation of information learnt through pheromone updates and the term heuristic values.

The following table, parts (a) and (b), shows the results obtained for the different  $q$ -value settings. The best predictive accuracy achieved is shown in bold and italics. Note that for  $q=0.0$ , the results are the same as those obtained in Test 1.

Exploration vs Exploitation	Ljubljana Breast Cancer		Wisconsin Breast Cancer		Tic-Tac-Toe	
	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
q=0.0	72.53	7.55	92.22	3.35	74.20	4.43
q=0.1	73.08	7.61	92.87	3.66	74.15	4.54
q=0.2	<b>73.30</b>	8.72	91.86	4.31	73.92	4.55
q=0.3	73.02	8.38	92.36	4.05	74.12	4.39
q=0.4	73.28	8.51	92.12	4.34	74.64	3.98
q=0.5	72.20	8.51	92.12	4.52	74.55	4.06
q=0.6	72.66	9.16	92.47	3.94	<b>75.06</b>	4.44
q=0.7	72.37	8.81	92.78	3.68	74.44	3.98
q=0.8	71.74	9.26	92.65	4.49	74.88	4.14
q=0.9	71.67	9.21	93.41	4.08	74.50	3.46
q=1.0	70.74	8.96	<b>94.28</b>	3.45	73.49	4.11

(a) – Ljubljana Breast Cancer, Wisconsin Breast Cancer, Tic-Tac-Toe Data Sets

Exploration vs Exploitation	Dermatology		Hepatitis		Cleveland Heart Disease	
	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
q=0.0	89.84	6.19	84.33	10.93	56.00	8.48
q=0.1	90.46	7.06	82.91	10.68	56.79	8.03
q=0.2	91.06	6.01	82.96	11.71	55.74	8.41
q=0.3	90.80	5.93	83.43	10.61	56.63	8.34
q=0.4	91.19	6.19	84.25	10.82	56.42	8.29
q=0.5	<b>91.59</b>	5.67	84.20	10.71	56.01	8.77
q=0.6	90.34	6.45	83.83	11.24	56.59	7.93
q=0.7	88.84	6.27	83.83	10.73	56.71	8.38
q=0.8	87.45	6.16	<b>85.54</b>	11.06	<b>57.08</b>	7.79
q=0.9	85.06	4.80	85.31	10.68	56.40	7.81
q=1.0	77.98	4.36	83.20	14.20	54.99	7.40

(b) – Dermatology, Hepatitis, Cleveland Heart Disease Data Sets

Table 7.5 (a) and (b) - Comparing Predictive Accuracy with Increase in Q-Value

The figures indicate that at least some exploitation of learnt information improves the predictive accuracy for all data sets.

It is particularly interesting to note that the deterministic algorithm achieved the best predictive accuracy of 94.28% for the Wisconsin Breast Cancer data set. Yet, for the Dermatology data set the deterministic algorithm achieved only 77.98% predictive accuracy compared to the best possible of 91.59% when a combination of both exploration and

exploitation is used. It is suggested that the value of this parameter needs to be determined empirically and depends at least partly on the data set itself.

The assumption that the data sets are a random and therefore representative sample of the underlying population may not be an appropriate one here (not enough information is available), and therefore may be a clue as to why the apparently contradictory results for the Wisconsin and Dermatology data sets were obtained. An interesting avenue of future research may be to determine which characteristics of a data set (e.g. proportion of population covered, correlation between attributes) make Ant-Miner and related stochastic algorithms achieve better or worse predictive accuracy than deterministic rule induction algorithms such as C4.5 and CN2.

Another beneficial side-effect of increasing the amount of exploitation by ants during rule building is a reduction in the time taken for an experiment. Table 7.6 below gives averages of elapsed time taken over ten ten-fold cross-validations, and the graph following illustrates that the overall trend for most of the data sets is a reduction in elapsed time. The figures come with the same proviso as for those in Table 7.4 Comparing Elapsed Time with Increase in Population Size, i.e. they can only be suggestive at this stage.

Data Set	Elapsed Time (mins)										
	q=0.0	q=0.1	q=0.2	q=0.3	q=0.4	q=0.5	q=0.6	q=0.7	q=0.8	q=0.9	q=1.0
Ljubljana Breast Cancer	0.30	0.50	0.30	0.30	0.20	0.10	0.00	0.00	0.00	0.00	0.00
Wisconsin Breast Cancer	2.80	1.70	1.60	1.20	0.90	1.50	0.90	1.80	1.00	1.00	0.00
Tic-Tac-Toe	4.90	5.80	4.70	4.00	3.80	3.20	2.50	2.30	1.50	1.10	1.00
Dermatology *	124.30	122.40	117.00	126.90	127.90	144.30	148.30	160.20	160.90	149.60	123.00
Hepatitis	2.60	2.60	2.30	2.40	2.10	2.00	1.60	1.50	0.90	0.30	0.00
Cleveland Heart Disease	4.20	3.40	2.90	2.80	2.90	2.50	2.30	1.00	0.80	0.40	0.00

Table 7.6 Comparing Elapsed Time with Increase in q-Value

\* The figures for the Dermatology data set (not shown in the figure below) represent an anomaly that requires further investigation.

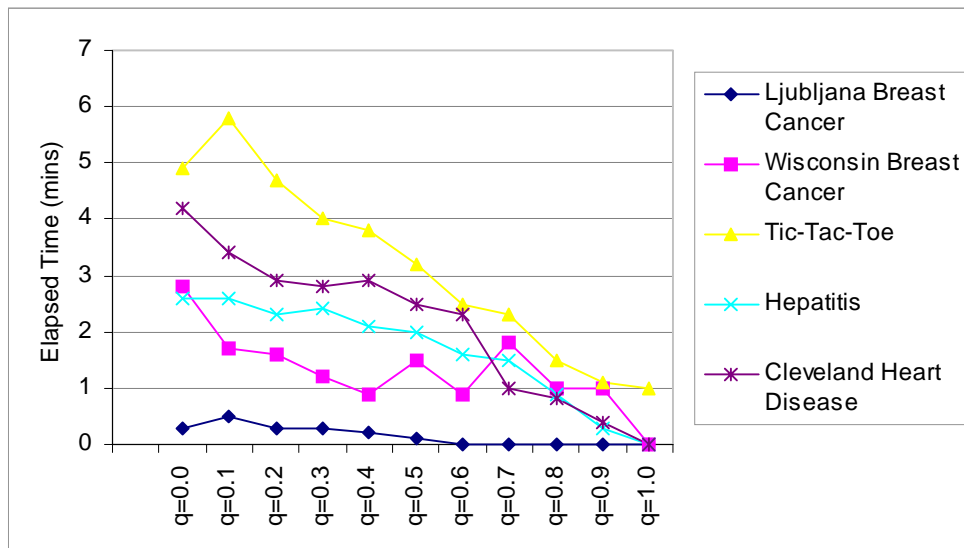


Figure 7.3 Comparing Elapsed Time with Increase in Q-Value

Remember that a term is generally chosen probabilistically. However, in order for it to be added to the rule antecedent being built by an ant, the addition of that term must not cause the number of instances in the training set that are covered by the resultant rule antecedent to fall below a pre-determined threshold – `minNoInstCovered`. The supposition for now is that choosing a term with a higher probability – i.e. a term that has been found in higher-quality rules that cover at least the minimum number of instances required – is less likely to cause the resultant rule antecedent to cover fewer training instances than is required, which would therefore necessitate the selection of other terms instead. If fewer redundant selections are made in constructing a rule, then less time is required. This, however, demands further investigation and confirmation.

Other interesting possibilities that may be followed up from this test and the previous one include:

- making the  $q$ -value dynamic. This would require more exploration at the beginning of an AntRun, that is during the creation of the first few rules, but more exploitation towards the end. This possibility was raised in [Bonabeau *et al*, 1999] in the context of discussing the Ant Colony System and its application to the travelling salesman problem, but as far as is known has not yet been implemented;

- combining an increase in population size with an increase in exploitation. Test 2 – Increasing Population Size indicated an increase in predictive accuracy in some of the data sets but also an increase in the time taken. Changing the q-value achieved some increase in predictive accuracy for all of the data sets, with a decrease in the time taken.

## 7.4 Test 4 – Handling Missing Attribute Values

This was the result of an extension to the implemented system to adapt it to deal with data sets that contained missing attribute values. Dr Parpinelli confirmed recently that the current version of Ant-Miner also handles instances with missing values.

Missing values within an instance were ignored by the program but the other terms contributed towards the calculation of their frequencies, and hence the entropies and heuristic values. This was done to utilize as much information as was available within a data set.

During training and classification, instances with missing values are treated in a similar manner to complete instances. A rule antecedent matches an instance with a missing value if the value for each attribute in the rule antecedent matches the value for that same attribute in the instance. Naturally, the rule conclusion and the class label of the instance must also match. If, however, a rule contains an attribute value but that same attribute has a missing value in the instance, then that rule cannot cover that instance.

For example, where ATT is attribute and VAL is value:

Rule A:	IF ATT1=VAL1	THEN	CLASS=A
Rule B:	IF ATT2=VAL2 AND ATT3=VAL1	THEN	CLASS=A
Instance (i):	ATT1=VAL1 ATT2=VAL3 ATT3=VAL3		CLASS=A
Instance (ii):	ATT1=VAL1 ATT2=VAL2 ATT3=?		CLASS=A
Instance (iii):	ATT1=? ATT2=VAL2 ATT3=VAL3		CLASS=A

Rule A covers instances (i) and (ii) but not (iii), while Rule B covers only instance (iii).

Remember these are ordered rule sets that are generated, and in practice if the criterion for the minimum number of instances covered by a rule is greater than one (to avoid over-fitting), and there are no duplicate instances in the training sets, then the rules themselves will be shorter than the instances.

Table 6.1 Data Set Properties, shows that the Hepatitis data set had the greatest proportion of missing values which resulted in 70 of the 155 instances containing at least one missing attribute value. As stated in Section 6.1 Experimental Methodology, with the exception of this test, instances with missing values were removed from the folds created for a ten-fold cross-validation. This meant that for the Hepatitis data set Tests 1 to 3, and Test 5 were run on only 55% of the original data set. Therefore, the inclusion of instances with missing values was likely to have the greatest impact on this data set, which is why this test was run only on the Hepatitis data set.

The folds for the Hepatitis data set were as similar as they could be to the folds that were used for Tests 1 to 3, and Test 5; instances with missing attribute values were only removed after the folds were created, and a copy of the original folds containing the missing values was saved for this later test.

Below are the results after running ten ten-fold cross-validations on the folds containing missing values, compared to the results obtained from Test 1.

	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average No. Rules in Rule Set	Average Standard Deviation (+/-)	Average No. Terms in Rule Set	Average Standard Deviation (+/-)	Average No. Terms in Rule
Folds INclude missing values	81.1648	7.5031	5.6200	0.4973	10.3100	1.2746	1.8345
Folds EXclude missing values	84.3302	10.9272	3.9800	0.0632	6.0200	1.1811	1.5125

Table 7.7 Comparing Results for the Hepatitis Data Set with Missing Attribute Values

The predictive accuracy achieved on the data set containing missing values is lower than that achieved when instances with missing values were removed. Initially it was expected that the extra information extracted from the data set would lead to better rule sets and improved

accuracy. However, a brief analysis of the distribution of the missing values provides a clue to the decreased predictive accuracy.

Attribute Id	# Missing Values	% Instances w. Missing Values
1	0	0.00
2	0	0.00
3	0	0.00
4	1	0.65
5	0	0.00
6	1	0.65
7	1	0.65
8	1	0.65
9	10	6.45
10	11	7.10
11	5	3.23
12	5	3.23
13	5	3.23
14	5	3.23
15	6	3.87
16	29	18.71
17	4	2.58
18	16	10.32
19	67	43.23
20	0	0.00

Table 7.8 Distribution of Missing Values for the Hepatitis Data Set

Table 7.8 highlights that approximately 75% of the attributes have a missing value in at least one instance of the data set. The attribute value for attribute 19, in particular, is missing in over 40% of the instances, with attribute 16 following with 18%. The terms of such attributes may be considered disadvantaged, therefore, when their relative frequencies and entropies are calculated. Their heuristic values will be low which will make it less likely for them to be selected for inclusion in a rule. This is supported by the fact that in the hundred rule sets generated by the ten ten-fold cross-validations for the Hepatitis data set run without missing values, eighty-three of those rule sets contained at least one rule with a term belonging to attribute 19. Not one term from this attribute appeared in the hundred rule sets generated from the ten ten-fold cross-validations run including the missing values.

The conclusion reached thus far is that instances with missing values should be included only if such values are approximately equally distributed between the attributes (and perhaps also between the elements of the domain of an attribute), especially if the heuristic is dependent on term entropy, and hence on relative term frequencies.

## 7.5 Test 5 – Use of Reduced Data Sets

Section 6.3.2 describes how attribute subsets were selected. The parameter settings for this test are as for Test 1, the only difference being that the data sets used have had certain attributes removed. The folds of the data sets for this test otherwise remain the same.

Table 7.9 below compares the number of attributes and terms in the data sets before and after subset selection. The number of attributes excludes the conclusion, and the number of terms excludes the class labels. The table also compares the average elapsed time for a ten-fold cross-validation on the data sets before and after attribute subset selection.

As may be expected with a reduced number of terms in the problem space the overall trend is a definite reduction in time. In particular note that the time for the Dermatology data set was reduced from over two hours to under a minute.

Data Set	UnReduced Data Sets			Reduced Data Sets		
	No. of Attributes	No. of Terms	Time Taken (mins)	No. of Attributes	No. of Terms	Time Taken (mins)
Ljubljana Breast Cancer	9	56	0.00	8	52	0.00
Wisconsin Breast Cancer	9	90	2.50	5	40	0.00
Tic-Tac-Toe	9	27	4.80	8	24	4.10
Dermatology	34	126	123.00	11	42	0.00
Hepatitis	19	37	2.90	11	20	0.00
Cleveland Heart Disease	13	39	0.60	10	34	0.60

Table 7.9 Number of Attributes and Terms Reduced by Feature Selection Procedure

The following table and graph compare the predictive accuracy achieved by the rule sets generated from the reduced data sets, with the predictive accuracy of the rule sets generated from the original data sets.

Data Set	UnReduced Data Sets		Reduced Data Sets	
	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
Ljubljana Breast Cancer	72.53	7.55	75.50	9.17
Wisconsin Breast Cancer	92.22	3.35	92.62	2.60
Tic-Tac-Toe	74.20	4.43	73.65	4.44
Dermatology	89.84	6.19	92.19	7.11
Hepatitis	84.33	10.93	84.89	13.71
Cleveland Heart Disease	56.00	8.48	55.95	7.97

Table 7.10 Predictive Accuracy of Rule Sets Generated from Reduced Data Sets

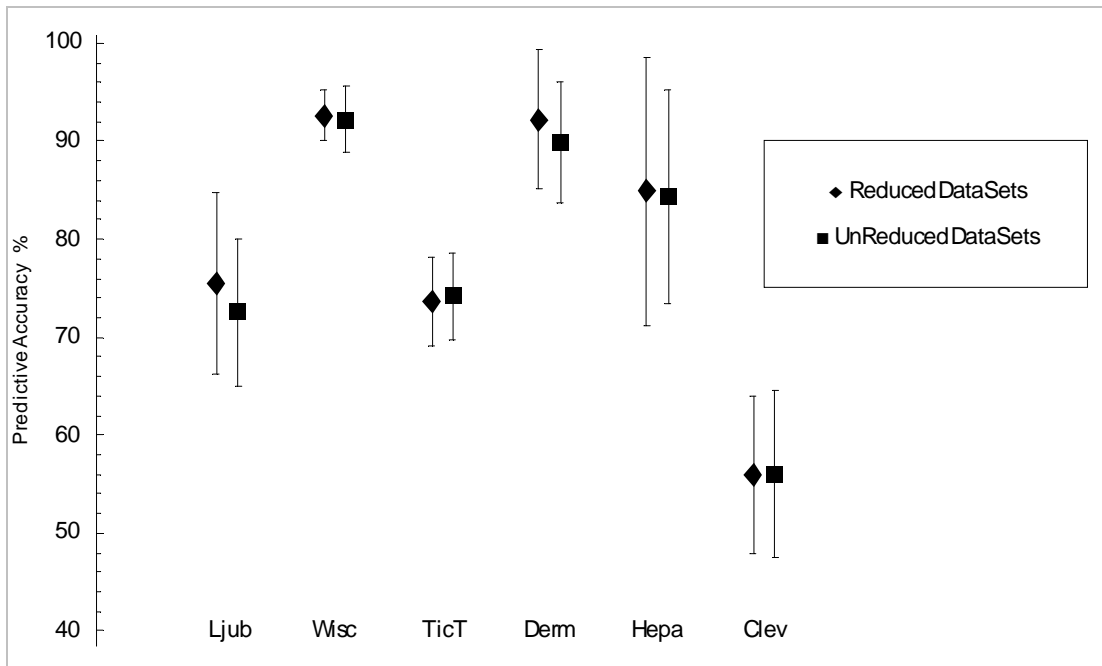


Figure 7.4 Predictive Accuracy of Rule Sets Generated from Reduced Data Sets

The graph clearly illustrates that the beneficial reduction in time is achieved through minimal or no cost at all to the predictive accuracy of the generated rule sets. In four of the data sets the predictive accuracy is improved while in the other two it decreases slightly. However, the

standard deviations in three of the data sets have also increased and these push the expected best results even higher.

The following table and graph indicate similar results for the size of rule sets and rules, i.e. minimal negative impact. The average number of rules for four of the data sets has increased slightly yet the overall individual rule size for all the data sets has decreased.

Data Set	UnReduced Data Sets			Reduced Data Sets		
	Average No. Rules in Ruleset	Average Standard Deviation (+/-)	Average No. Terms in Rule	Average No. Rules in Ruleset	Average Standard Deviation (+/-)	Average No. Terms in Rule
Ljubljana Breast Cancer	8.23	0.90	1.07	8.43	0.99	0.98
Wisconsin Breast Cancer	11.51	0.86	1.01	11.33	0.84	0.91
Tic-Tac-Toe	8.12	1.52	1.26	7.53	1.78	1.15
Dermatology	7.40	0.54	2.83	7.90	0.51	2.77
Hepatitis	3.98	0.06	1.51	4.36	0.49	1.42
Cleveland Heart Disease	14.82	0.97	1.84	15.11	1.07	1.75

Table 7.11 Size of Rule Sets and Rules Generated by Reduced Data Sets

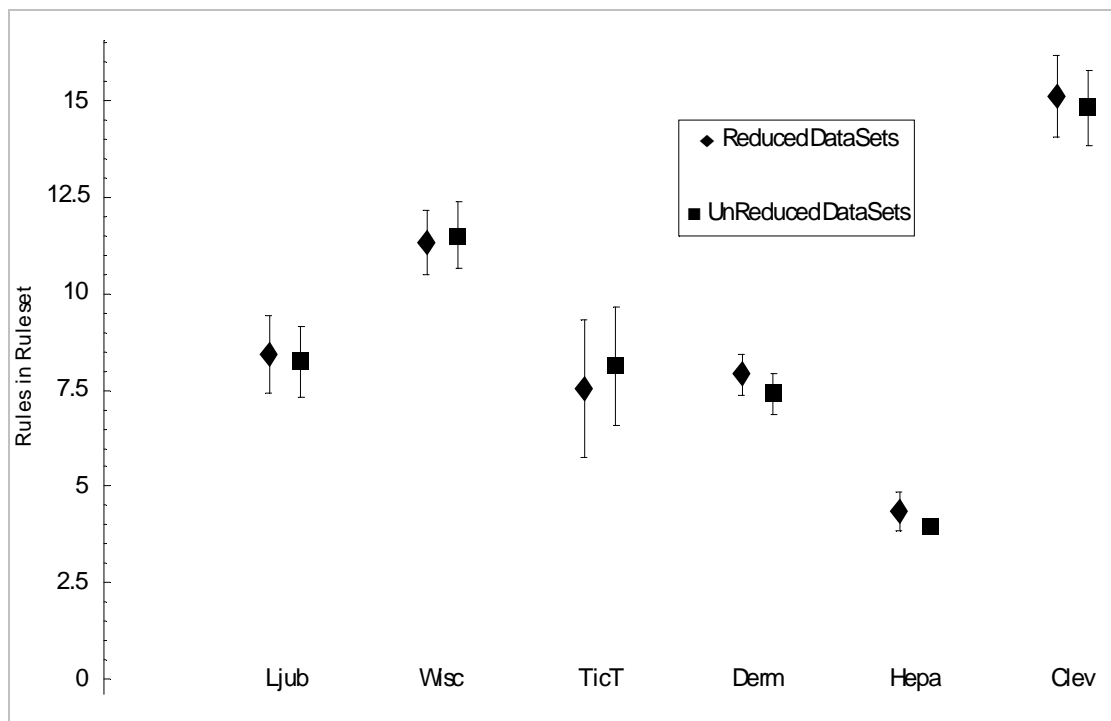


Figure 7.5 Size of Rule Sets Generated by Reduced Data Sets

Another benefit attained from a reduction in the number of attributes is that the pheromone levels of the problematic data sets mentioned in Test 1 – Dermatology, Hepatitis and Cleveland Heart Disease – remained within Java's `double` range. This may be as expected – with the elimination of attributes that have little influence on the class label, and therefore the elimination of insignificant terms whose pheromone levels are likely to get very small, it should be possible to run a greater number of iterations. This was tested and confirmed for the Dermatology and Cleveland data sets. Some terms for the Hepatitis data set were still producing very low pheromone levels.

In the case of Dermatology additional tests on the reduced data set were run at 2000 and 3000 iterations, attaining a predictive accuracy of 92.36% ( $\sigma = 7.48$ ) and 93.34% ( $\sigma = 6.14$ ) respectively.

For the Cleveland Heart Disease data set additional tests were also run at 2000 and 3000 iterations, attaining a predictive accuracy of 56.81% ( $\sigma = 8.09$ ) and 55.91% ( $\sigma = 8.04$ ) respectively.

## 7.6 Test 6 – Use of 'Tiger' Data Set

This was a quick test run on a subset of the 'Tiger' data set being used in [Pan, 2002]. The aim of that project is to utilise rule induction techniques to predict failure of specific parts of gas turbines. The data set represents data acquired from gas turbines in operation over a two-day period. It has 5 attributes – one continuous and the other four nominal – excluding the conclusion attribute. The data set has 4406 instances, both *C4.5* and the system implemented for this project were applied to it.

*C4.5* produced the following tree, with 35 nodes and 22 leaves, i.e. rules. The rules have an average of 3.64 terms each and the predictive accuracy achieved over a ten-fold cross validation was 97.21%:

```

A68A0TE858 <= 142.2306
|
| COMB_8_8_1 = 0: 0 (2460.0/13.0)
| COMB_8_8_1 = 1: 5 (3.0/1.0)
| COMB_8_8_1 = 2: 0 (4.0)
| COMB_8_8_1 = 3: 0 (4.0/2.0)
| COMB_8_8_1 = 4
| | A68A0TE858 <= 136.4147: 0 (2.0/1.0)
| | A68A0TE858 > 136.4147: 2 (2.0/1.0)
| COMB_8_8_1 = 5
| | A68A0TE858 <= 136.5049: 2 (3.0/2.0)
| | A68A0TE858 > 136.5049: 5 (4.0/1.0)
|
|
A68A0TE858 > 142.2306
|
| A68A0TE858 <= 182.2567: 5 (1860.0/38.0)
| A68A0TE858 > 182.2567
| | COMB_8_9_1 = 0
| | | A68A0TE858 <= 415.6815
| | | | A68A0TE858 <= 207.7871
| | | | | A68A0TE858 <= 206.193: 1 (2.0/1.0)
| | | | | A68A0TE858 > 206.193: 5 (6.0/2.0)
| | | | A68A0TE858 > 207.7871: 1 (5.0/3.0)
| | | A68A0TE858 > 415.6815: 0 (4.0)
| | COMB_8_9_1 = 1: 5 (9.0/4.0)
| | COMB_8_9_1 = 2: 0 (4.0/2.0)
| | COMB_8_9_1 = 3
| | | A68A0TE858 <= 207.6847: 0 (2.0/1.0)
| | | A68A0TE858 > 207.6847: 4 (2.0/1.0)
| | COMB_8_9_1 = 4: 0 (4.0/1.0)
| | COMB_8_9_1 = 5
| | | A68A0TE858 <= 207.0266
| | | | A68A0TE858 <= 206.193: 0 (2.0/1.0)
| | | | A68A0TE858 > 206.193: 1 (4.0/2.0)
| | | A68A0TE858 > 207.0266
| | | | A68A0TE858 <= 208.2356: 5 (16.0/3.0)
| | | | A68A0TE858 > 208.2356: 0 (4.0/2.0)
|
|

```

Part of the output resulting from running the data set on the implemented system is shown on the following page. It specifies the parameter settings and shows one of the rule sets generated in the ten-fold cross-validation as an example. It is apparent from the previous tree and the rule set below that different rules are actually generated. For instance, an attribute that does not appear at all in the tree above – COMB\_20\_1\_1 – appears in the rule set below and also in at least one rule of the other nine rule sets generated under this ten-fold cross-validation.

The average predictive accuracy achieved compares well with *C4.5*'s accuracy, at 97.62%, the average number of rules in a rule set was 13.9 and the average number of terms in a rule was 1.39. This again confirms Parpinelli's *et al* claim that Ant-Miner achieves comparable accuracy with simpler rule sets and rules.

## Chapter 7 Results and Analyses

PARAMETER SETTINGS:  
No Ants per Iteration: 1  
Max no iterations: 500  
Min no instances covered by rule: 10  
Max no instances left uncovered: 10  
Max no rules allowed to converge: 10  
Transition rule: random proportional selection

TRAINING DATA DETAILS:  
No attributes inc predictor: 6  
No class labels: 15  
No terms in problem space: 28

Seed for this run: 1030448299968

\*\*\*\*\*  
k = 6

Total no instances in training data: 3965  
No instances in training data with missing values: 0

Final rule set has 14 rules:

```
IF A68AOTE858=142.23057-182.25674
    THEN CLASS=5          No instances covered: 1654
IF A68AOTE858<=142.23057
    COMB_8_8_1=0
    THEN CLASS=0          No instances covered: 2186
IF A68AOTE858=142.23057-182.25674
    COMB_8_9_1=5
    THEN CLASS=0          No instances covered: 31
IF COMB_8_8_1=0
    COMB_8_9_1=5
    THEN CLASS=5          No instances covered: 19
IF COMB_8_8_1=0
    COMB_8_9_1=5
    THEN CLASS=1          No instances covered: 6
IF COMB_8_9_1=0
    COMB_20__1=0
    THEN CLASS=1          No instances covered: 5
IF A68AOTE858=182.25674-415.6815
    THEN CLASS=0          No instances covered: 16
IF A68AOTE858<=142.23056
    THEN CLASS=0          No instances covered: 7
IF COMB_8_9_1=5
    THEN CLASS=2          No instances covered: 6
IF A68AOTE858=182.25674-415.6815
    THEN CLASS=5          No instances covered: 11
IF A68AOTE858<=142.23056
    THEN CLASS=5          No instances covered: 5
IF COMB_8_9_1=5
    THEN CLASS=4          No instances covered: 5
IF COMB_8_8_1=0
    THEN CLASS=3          No instances covered: 5
IF      THEN CLASS=4          No instances covered: 9
```

AntRun	NoIterations
1	18
2	32
3	15
4	15
5	23
6	17
7	16
8	22
9	19

10	24
11	13
12	12
13	10

Total no instances in test set: 441  
No instances in test data with missing values: 0  
Predictive accuracy: 97.73242630385488%

Some points are worth noting at this stage. Initially the data set had duplicate instances. When the continuous attribute was discretized this resulted in even more duplicate instances which would have helped to obtain an optimistic predictive accuracy on the cross-validation; if there are duplicate instances then it is possible some of the instances in the test set were also in the training set. Furthermore, the discretization process resulted in inconsistent instances, which resulted in apparently contradictory rules in the rule sets generated.

In conclusion, this was an informative experiment to conduct. It highlighted potential issues when dealing with large real-world data sets, such as the above-mentioned duplicate and inconsistent instances arising from discretization of continuous attributes. The authors of Ant-Miner, Parpinelli *et al*, have indicated that one of their future research directions will be adapting their algorithm to deal directly with continuous attributes.



## Chapter 8

### Conclusions and Future Work

The aim of this project was to investigate the application of swarm intelligence to rule induction. The specific objectives were to investigate the role of exploration and provide a more directed search during rule building by ants. With this in mind, the next section summarises the conclusions drawn from the tests results discussed in Chapter 7, while the following section suggests avenues for future work.

#### 8.1 Summary and Conclusions

The system implemented re-affirms Parpinelli's *et al* claim that ant colony optimisation is a viable approach to the rule induction problem. The predictive accuracy achieved by this approach compares favourably with that achieved by more well-established algorithms such as *C4.5* and *CN2*, and the rule sets and rules generated are considerably smaller and simpler.

The changes in population size and transition rule effected in tests 2 and 3 indicate that an improvement in predictive accuracy is possible. The results suggest that by combining the two changes it may be possible to counteract the detrimental effect of increase in time due to one change (population increase), by the additional benefit of a reduction in time due to the other change (different transition rule). More tests are necessary to confirm the significance

of the improvement in accuracy achieved by these changes, individually and combined. Other variations on these tests that may confirm the benefit of fine-tuning the balance between exploration and exploitation have also been mentioned when discussing these test results.

Tests 4 and 6 proved informative in highlighting some of the issues to be aware of when dealing with real-world data sets. Test 5 confirms a different way of improving or maintaining accuracy and reducing performance time.

Time had not initially been considered to be a relevant evaluation criteria. However, the results obtained due to the modifications made to the basic algorithm suggested that it should be taken into account. With hindsight, it would have been useful to prepare for the collation of more accurate and reliable estimates of this performance factor from the beginning, but the estimates actually obtained at least suggest trends that may be confirmed at a later stage.

## **8.2 Future Work**

Several extensions to the current work have already been mentioned in the preceding section, in Chapter 7 when discussing the results, and in Chapter 4 when initially investigating the scope of this project. These include trying Ant-Miner with different heuristic functions, pheromone update strategies, fitness functions and more transition rules. These changes should be relatively easy to effect and investigate.

Other more demanding work might involve obtaining a thorough understanding of what factors might render this stochastic approach a better or worse alternative to deterministic rule induction algorithms. These factors are likely to include specific features of the data sets but may also include particular combinations of changes made to previously mentioned features such as the transition rule and the pheromone update strategy.

On a more specific note, work may be directed along the following lines. This project has concentrated on applying ant algorithms to the search for *crisp* rules in *modest-sized discrete* data sets. Considering each of the italicised words in turn:

- the generation of fuzzy rules would be a useful direction as many real-world application data sets are best described by fuzzy rules. This might also improve the comprehensibility of the rules themselves. The work of [Casillas *et al*, 2000] is a start but it focuses on using ant algorithms to assign class labels to previously determined fuzzy rule antecedents, as opposed to actually creating fuzzy rules by ants;
- the data sets used in this project are real-world but modest-sized. Test 5 confirms the benefits of attribute subset selection before the generation of rule sets. From an implementation aspect, this could become a standard pre-processing step and it might be interesting to also look into related methods such as instance subset selection. Recent work in this area uses an evolutionary approach to generate a smaller subset of instances of the data set that can cover the domain knowledge as well as the original data set, [Endou & Zhao, 2002]. Many evolutionary computation approaches also lend themselves particularly well to parallel processing and this may prove useful in dealing with very large real-world data sets.
- Parpinelli *et al* have already stated the potential use of adapting Ant-Miner to deal directly with continuous attributes. This would eliminate the possible loss of valuable information in the discretization process, and other potential issues arising such as redundant or inconsistent instances.

As a final suggestion, the generation of *unordered* rule sets may be considered. Ant-Miner as it currently is generates ordered rule sets that are relatively compact and have short individual rules. However, ordered rule sets, or lists, come with an inherent deterrent to comprehensibility – the meaning of a single rule in the list is dependent on all the previous rules and the further down the list you get the more difficult it is to make sense of the rule. Good starting points for this work may already exist. For instance, Ant-Miner clearly follows the same 'separate-and-conquer' strategy as *CN2*, which has been adapted to generate unordered rule sets, [Clark & Boswell, 1991].



## Bibliography

- [Agresti, 1990] Agresti, A, *Categorical Data Analysis*. NY, John Wiley & Sons, 1990.
- [Aha *et al*, 1991] Aha D, Kibler D, Albert M, "Instance-Based Learning Algorithms". In: *Machine Learning*, 6(1), pages 37-66, 1991.
- [Baker & Jain, 1976] Baker E, Jain AK, "On Feature Ordering in Practice and Some Finite Sample Effects". In: *Proceedings of the Third International Joint Conference on Pattern Recognition*, pages 45-49, CA, 1976.
- [Bay, 1995] Bay JS, "Design of the Army-Ant Cooperative Lifting Robot". In: *IEEE Robotics and Automation Mag.*, 2, pages 36-43, 1995.
- [Ben-Bassat, 1982] Ben-Bassat, M, "Use of Distance Measures, Information Measures and Error Bounds on Feature Evaluation". In: Krishnaiah, Kanal, editors, *Handbook of Statistics 2*, pages 773-791, 1982.
- [Berry & Linoff, 1997] Berry MJA, Linoff G, *Data Mining Techniques for Marketing, Sales, and Customer Support*. John Wiley & Sons, 1997.
- [Bonabeau *et al*, 1999] Bonabeau E, Dorigo M, Theraulez G, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, NY, 1999.
- [Breiman *et al*, 1984] Breiman L, Friedman JH, Olshen RA, Stone CJ, *Classification and Regression Trees*. CA, Wadsworth, 1984.
- [Buchanan & Wilkins, 1993] Buchanan BG, Wilkins DC, editors, *Readings in Knowledge Acquisition and Learning*. Morgan Kaufmann, CA, 1993.
- [Casillas *et al*, 2000] Casillas, Cordon, Herrera "Learning Fuzzy Rules using Ant Colony Optimization Algorithms". In: .....
- [Chen *et al*, 2001] Chen S-M, Lee S-H, Lee C-H, "A new Method for Generating Fuzzy Rules from Numerical Data for Handling Classification Problems". In: *Applied Artificial Intelligence*, 15, pages 645-664, 2001.
- [Clark & Boswell, 1991] Clark P, Boswell R, "Rule Induction with CN2: Some Recent Improvements". In: Yves Kodratoff editor, *Proceedings of the Fifth European Conference on Machine Learning*, pages 151-163. Berlin, Springer-Verlag, 1991.

## Bibliography

- [Clark & Niblett, 1989] Clark P, Niblett T, "The CN2 Induction Algorithm". In: *Machine Learning Journal*, 3 (4), pages 261-283. The Netherlands, Kluwer, 1989.
- [Cordon *et al*, 1999] Cordon O, del Jesus MJ, Herrera F, "Evolutionary Approaches to the Learning of Fuzzy Rule-Based Classification Systems". In: Jain LC, editor, *Engineering and Information Systems and their Applications*, pages 107-160. CRC Press, 1999.
- [Craven & Shavlik, 1993] Craven MW, Shavlik JW, "Using Sampling and Queries to Extract Rules from Trained Neural Networks". In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1319-1324. France, Morgan Kaufmann, 1993.
- [Dorigo *et al*, 1996] Dorigo M, Maziello V, Colomi A, "The Ant System: Optimization by a Colony of Cooperating Ants". In: *IEEE Transactions on Systems, Man and Cybernetics B*, 26(1), pages 29-41, 1996.
- [Dorigo & Gambardella, 1997] Dorigo M, Gambardella LM, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". In: *IEEE Transactions on Evolutionary Computation*, 1, pages 53-66, 1997.
- [Ducatelle & Levine, 2001] Ducatelle F, Levine J, "Ant Colony Optimisation for Bin Packing and Cutting Stock Problems", UK Workshop on Computational Intelligence, The University of Edinburgh, 2001.
- [Eibe & Witten, 1998] Eibe F, Witten IH, "Generating Accurate Rule Sets Without Global Optimization". In: *Proceedings of the 15<sup>th</sup> International Conference on Machine Learning*, pages 144-157, Morgan Kaufmann, 1998.
- [Eibe, 2000] Eibe F, *Pruning Decision Trees and Lists*. PhD Thesis, University of Waikato, 2000.
- [Endou & Zhao, 2002] Endou T, Zhao Qiangfu, "Generation of Comprehensible Decision Trees Through Evolution of Training Data". In: *Proceedings of the IEEE World Congress on Computational Intelligence*, 2002.
- [Fayyad *et al*, 1996] Fayyad UM, Piatetsky-Shapiro G, Smyth P, "From Data Mining to Knowledge Discovery: An Overview". In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R, editors, *Advances in Knowledge Discovery & Data Mining*, pages 1-34. MA, MIT Press, 1996.
- [Flake, 1998] Flake GW, *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MA, MIT Press, 1998.
- [Freitas, 1999] Freitas AA, "A Genetic Algorithm for Generalized Rule Induction". In Roy R *et al*, editor, *Advances in Soft Computing – Engineering Design and Manufacturing*, pages 340-353. Springer-Verlag, 1999.
- [Freitas, 2001] Freitas AA, "Understanding the Crucial Role of Attribute Interaction in Data Mining". In: *Artificial Intelligence Review*, 16(3), pages 177-199, 2001.

- [Freitas, 2002] Freitas, AA, "A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discover". To appear in: Ghosh A, Tsutsui S, editors, *Advances in Evolutionary Computation*, Springer-Verlag, 2002.
- [Gambardella & Dorigo, 1995] Gambardella LM, Dorigo M, "Ant-Q: A Reinforcement Learning Approach to the Travelling Salesman Problem". In: *Proceedings of the Twelfth International Conference on Machine Learning*, pages 252-260. CA, Morgan Kaufmann, 1995.
- [Hand, 1997] Hand DJ, *Construction and Assessment of Classification Rules*. John Wiley & Sons, 1997.
- [Hunter & Klein, 1993] Hunter L, Klein T, "Finding Relevant Biomolecular Features". In: *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1993.
- [Janikow, 1998] Janikow CZ, "Fuzzy Decision Trees; Issue and Methods". In: *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics* vol.28 no.1, 1998.
- [Jensen & Shen, 2002a] Jensen R, Shen Qiang, "Applying Swarm Intelligence to Rule Induction". Project proposal, Division of Informatics, The University of Edinburgh, 2002.
- [Jensen & Shen, 2002b] Jensen R, Shen Qiang, "Fuzzy-Rough Sets for Descriptive Dimensionality Reduction". In: .....????, 2001?
- [Jordan, 1999] Jordan MI, editor, *Learning in Graphical Models*. MA, MIT Press, 1999.
- [Kasabov, 1996] Kasabov NK, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. MA, MIT Press, 1996.
- [Kearns, 1996] Kearns M, "A Bound on the Error of Cross-validation Using the Approximation and Estimation Rates, with Consequence for the Training-Test Split". In: Touretzky DS, Mozer MC, Hasselmo ME, editors, *Advances in Neural Information Processing Systems*, 8, pages 183-189, MIT Press, MA, 1996.
- [Kohavi & Sahami, 1996] Kohavi R, Sahami M, "Error-Based and Entropy-Based Discretization of Continuous Features". In: *Proceedings of the 2<sup>nd</sup> International conference Knowledge Discovery and Data Mining*, pages 114-119. AAAI Press, CA, 1996.
- [Kohavi, 1995] Kohavi R, "An Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14<sup>th</sup> International Joint Conference on Artificial Intelligence*, pages 1137-1143, Morgan Kauffman, CA, 1995.
- [Kwedlo & Kretowski, 1998] Kwedlo W, Kretowski M, "Discovery of Decision Rules from Databases: An Evolutionary Approach". *European Symposium on Principles of Mining and Knowledge Discovery (PDD-98)*. Lecture Notes in Artificial Intelligence 1510, pages 371-378. Springer-Verlag, 1998.

## Bibliography

- [Liu & Kwok, 2000] Liu JJ, Kwok J T-Y, *An Extended Genetic Rule Induction Algorithm*, 2000?
- [Maniezzo et al, 1994] Maniezzo V, Colorni A, Dorigo M, *The Ant System applied to the Quadratic Assignment Problem*. Technical Report IRIDIA/94-28, Universite Libre de Bruxelles, Belgium, 1994.
- [Monmarche, 1999] Monmarch N, "On Data Clustering with Artificial Ants". In: Freitas AA, editor, *Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop*, pages 23-26. AAAI Press, 1999.
- [Mendes et al, 2001] Mendes RRF, Voznika FB, Freitas AA, Nievola JC, "Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution". In: *Proceedings of the Fifth European conference on Knowledge Discovery and Data Mining (PKDD-2001)*. Lecture Notes in Artificial Intelligence 2168, pages 314-325. Springer-Verlag, 2001.
- [Michalski, 1969] Michalski RS, "On the Quasi-Minimal Solution of the Covering Problem". In: *Proceedings of the Fifth International Symposium on Information Processing, A3 (Switching Circuits)*, pages 125-128. Bled, Yugoslavia.
- [Mingers, 1989] Mingers J, "An Empirical Comparison of Selection Measures for Decision Tree Induction". In: *Machine Learning*, 3, pages 319-342, 1989.
- [Nozaki et al, 1997] Nozaki K, Ishibuchi H, Tanaka H, "A Simple but Powerful Heuristic Method for Generating Fuzzy Rules from Numerical Data". In: *Fuzzy Sets and Systems*, 86, pages 251-270, 1997.
- [Pan, 2002] Pan Y, *Gas Turbine Combustion Area Condition Prediction using Induced Decision Trees*. MSc Artificial Intelligence dissertation, The University of Edinburgh, 2002.
- [Parpinelli et al, 2001] Parpinelli RS, Lopes HS, Freitas AA, "An Ant Colony Based System for Data Mining: Applications to Medical Data". In *GECCO Proceedings.....*, 2001.
- [Parpinelli et al, 2002a] Parpinelli RS, Lopes HS, Freitas AA, "An Ant Colony Algorithm for Classification Rule Discovery". In: Abbas HA, Sarker RA, Newton CS, editors, *Data Mining: A Heuristic Approach*, pages 190-208, Idea Group Publishing, London, 2002.
- [Parpinelli et al, 2002b] Parpinelli RS, Lopes HS, Freitas AA, "Data Mining with an Ant Colony Optimization Algorithm". To appear in *IEEE Transactions on Evolutionary Computation*, special issue on Ant Colony Algorithms, 2002
- [Pazzani & Kibler, 1992] Pazzani M, Kibler D, "The Utility of Knowledge in Inductive Learning". In: *Machine Learning*, 9(1), 1992.
- [Quinlan, 1987] Quinlan JR, "Generating Production Rules from Decision Trees". In: McDermott J, editor, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 304-307. Morgan Kaufmann, 1987

- [Quinlan, 1992] Quinlan JR, *C4.5: Programs for Machine Learning*. CA, Morgan Kaufmann, 1992.
- [Rivest, 1987] Rivest RL, "Learning Decision Lists". In: *Machine Learning*, 2(3), pages 229-246, 1987.
- [Rumelhart & McClelland, 1986] Rumelhart DE, McClelland JL, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol.1: Foundations*. MA, MIT Press, 1986.
- [Stülze & Hoos, 2000] Stülze T, Hoos H, "Max-Min Ant System". In: *Future Generation Computer Systems*, 16(8), pages 889-914, 2000.
- [Umano *et al*, 1994] Umano M, Okomoto H, Hatono I, Tamura H, Kawachi F, Umedzu S, Kinoshita J, "Fuzzy Decision Trees by Fuzzy ID3 Algorithm and its Application to Diagnosis System". In: *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 2113-2118, 1994.
- [Wang & Mendel, 1992] Wang LX, Mendel JM, "Generating Fuzzy Rules by Learning from Examples". In: *IEEE Transactions on Systems, Man and Cybernetics*, 22(6), pages 1414-1427, 1992.

## Appendix A User-Defined System Parameters

Parameter Name	Parameter Description	Default setting
alpha	<p>Heuristic value exponent – this setting, together with the one below – beta, defines the relative importance of the information provided by the heuristic value vs. the information provided by the pheromone level during the calculation of a term probability.</p> <p>[Parpinelli et al, 2002a] gave both types of information equal importance and did not experiment with changing either of the settings. This implementation makes it possible to conduct such experimentation, though none was done during this project.</p>	1
beta	Pheromone value exponent – see alpha.	1
maxNoIterations	<p>Maximum number of iterations that may be generated by an AntRun.</p> <p>An AntRun may halt before this condition is satisfied if the number of identical rules produced by iterations in succession reaches another parameter setting value – maxRulesConverge (see below).</p> <p>See Section 5.1 Conceptual Overview of the System, and Figure 5.3 Level II view of the system - creation of ants and rules, for a more detailed discussion.</p>	3000

Appendix A User-Defined System Parameters

Parameter Name	Parameter Description	Default setting
noAnts	<p>Number of ants created during an iteration. A value of 1 makes this implementation equivalent to the Ant-Miner algorithm, which may be considered as running a maximum of 3000 iterations each with a population of one ant.</p> <p>Section 5.3.2 Ant Population Size, and Figure 5.5 Creation of ants and rules in the original Ant-Miner algorithm, provide a more detailed discussion.</p>	1
maxInstUncovered	<p>Maximum number of training set instances that may be left uncovered.</p> <p>This parameter setting determines when the generation of AntRuns should halt - when the number of instances remaining in the training set reaches this setting value or is below it, then no more rules are created (i.e. no more AntRuns), and the default rule to cover the remaining instances is then created.</p> <p>See Section 5.1 Conceptual Overview of the System, and Figure 5.2 Level I view of the system – a run of a k-fold cross-validation, for more detail.</p>	10
minInstPerRule	<p>Minimum number of training set instances that must be covered by the rule created by an ant.</p>	10
maxRulesConverge	<p>Maximum number of identical rules created in successive iterations before an AntRun is halted. See <code>maxNoIterations</code> above.</p>	10
transitionRule	<p>Determines which transition rule should be used by an ant in determining the next term to be added to the rule antecedent being built.</p> <p>If the pseudo-random proportional rule is chosen (instead of Ant-Miner's random proportional transition rule), then a <i>q-value</i> setting must also be given – this determines the balance between exploration and exploitation.</p>	random proportional

## Appendix B Sample Outputs from Implemented

### System

#### Analysis of Tic-Tac-Toe Dataset

The following is the output resulting from one ten-fold cross-validation test on the Tic-Tac-Toe data. The data set encodes the 958 valid possible board configurations for the end of a tic-tac-toe game, where the player X starts first and the conclusion indicates whether X has won the game or not, i.e. `Win_for_X=yes` or `Win_for_X=no` respectively.

The rule antecedent is made of terms that indicate whether various squares on the board have been taken over by Player X or by Player O. An example of a generated rule is:

```
IF BotLeftSq=X and BotMiddleSq=X and BotRightSq=X THEN Win_for_X=yes.
```

That is, if Player X has taken over the bottom left, bottom middle and bottom right squares of the board, then he has won the game.

Appendix B Sample Outputs from Implemented System

Tue Sep 03 13:54:32 BST 2002

tic-tac-toe - 10-fold cross validation

PARAMETER SETTINGS:

No Ants per Iteration: 1  
Max no iterations: 3000  
Min no instances covered by rule: 10  
Max no instances left uncovered: 10  
Max no rules allowed to converge: 10  
Transition rule: random proportional selection

TRAINING DATA DETAILS:

No attributes inc predictor: 10  
No class labels: 2  
No terms in problem space: 27

Seed for this run: 1031049872382

\*\*\*\*\*

k = 1

Total no instances in training data: 862  
No instances in training data with missing values: 0

Final rule set has 9 rules:

IF MidMiddleSq=X		
THEN Win_For_X=Yes	No instances covered:	325
IF TopLeftSq=0		
THEN Win_For_X=No	No instances covered:	135
IF TopRightSq=X		
THEN Win_For_X=Yes	No instances covered:	139
IF BotLeftSq=0		
THEN Win_For_X=No	No instances covered:	83
IF TopRightSq=X		
THEN Win_For_X=No	No instances covered:	34
IF BotLeftSq=X		
BotMiddleSq=X		
BotRightSq=X		
THEN Win_For_X=Yes	No instances covered:	44
IF TopRightSq=0		
MidRightSq=0		
BotRightSq=0		
THEN Win_For_X=No	No instances covered:	28
IF TopLeftSq=X		
MidLeftSq=X		
BotLeftSq=X		
THEN Win_For_X=Yes	No instances covered:	45
IF	THEN Win_For_X=No	No instances covered: 29

AntRun NoIterations

1	13
2	40
3	28
4	21
5	24
6	24

Appendix B Sample Outputs from Implemented System

7 23  
8 22  
9 571

Total no instances in test set: 96  
No instances in test data with missing values: 0  
Predictive accuracy: 81.25%

\*\*\*\*\*  
k = 2

Total no instances in training data: 862  
No instances in training data with missing values: 0

Final rule set has 10 rules:

IF MidMiddleSq=X		
THEN Win_For_X=Yes	No instances covered:	324
IF BotLeftSq=0		
THEN Win_For_X=No	No instances covered:	134
IF MidMiddleSq=B		
THEN Win_For_X=Yes	No instances covered:	102
IF TopLeftSq=0		
THEN Win_For_X=No	No instances covered:	84
IF BotRightSq=X		
THEN Win_For_X=Yes	No instances covered:	79
IF MidRightSq=0		
THEN Win_For_X=No	No instances covered:	56
IF TopMiddleSq=0		
BotMiddleSq=0		
THEN Win_For_X=No	No instances covered:	25
IF MidLeftSq=X		
THEN Win_For_X=Yes	No instances covered:	32
IF TopMiddleSq=X		
TopRightSq=X		
THEN Win_For_X=Yes	No instances covered:	21
IF	THEN Win_For_X=No	No instances covered: 5

AntRun	NoIterations
1	17
2	54
3	61
4	74
5	20
6	33
7	24
8	23
9	10

Total no instances in test set: 96  
No instances in test data with missing values: 0  
Predictive accuracy: 75.0%

\*\*\*\*\*  
k = 3

Total no instances in training data: 862

Appendix B Sample Outputs from Implemented System

No instances in training data with missing values: 0

Final rule set has 9 rules:

IF MidMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 329		
IF TopLeftSq=0	THEN Win_For_X=No	No instances covered: 138		
IF BotLeftSq=X	THEN Win_For_X=Yes	No instances covered: 133		
IF TopRightSq=0	THEN Win_For_X=No	No instances covered: 79		
IF BotMiddleSq=0	THEN Win_For_X=No	No instances covered: 55		
IF MidLeftSq=0	MidMiddleSq=0	MidRightSq=0	THEN Win_For_X=No	No instances covered: 25
IF TopMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 60		
IF MidRightSq=X	BotRightSq=X	THEN Win_For_X=Yes	No instances covered: 38	
IF	THEN Win_For_X=No	No instances covered: 5		

AntRun NoIterations

1	10
2	93
3	190
4	25
5	86
6	17
7	33
8	10

Total no instances in test set: 96  
No instances in test data with missing values: 0  
Predictive accuracy: 77.08333333333333%

\*\*\*\*\*  
k = 4

Total no instances in training data: 862  
No instances in training data with missing values: 0

Final rule set has 8 rules:

IF MidMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 328		
IF TopRightSq=0	THEN Win_For_X=No	No instances covered: 135		
IF BotRightSq=X	THEN Win_For_X=Yes	No instances covered: 139		
IF TopLeftSq=0	THEN Win_For_X=No	No instances covered: 79		
IF TopLeftSq=X	MidLeftSq=X	BotLeftSq=X	THEN Win_For_X=Yes	No instances covered: 49

*Appendix B Sample Outputs from Implemented System*

```

IF TopMiddleSq=0
    THEN Win_For_X=No           No instances covered: 36
IF TopLeftSq=X
    TopMiddleSq=X
    TopRightSq=X
    THEN Win_For_X=Yes         No instances covered: 50
IF      THEN Win_For_X=No         No instances covered: 46

```

```

AntRun  NoIterations
1        17
2        26
3       2746
4        13
5        35
6        28
7        29
8       3000

```

Total no instances in test set: 96  
 No instances in test data with missing values: 0  
 Predictive accuracy: 73.95833333333333%

\*\*\*\*\*  
 k = 5

Total no instances in training data: 862  
 No instances in training data with missing values: 0

Final rule set has 10 rules:

```

IF MidMiddleSq=X
    THEN Win_For_X=Yes         No instances covered: 332
IF MidMiddleSq=B
    THEN Win_For_X=Yes         No instances covered: 103
IF TopLeftSq=0
    THEN Win_For_X=No         No instances covered: 130
IF TopRightSq=X
    MidMiddleSq=0
    THEN Win_For_X=Yes         No instances covered: 71
IF BotLeftSq=0
    THEN Win_For_X=No         No instances covered: 82
IF MidRightSq=0
    THEN Win_For_X=No         No instances covered: 58
IF MidMiddleSq=0
    BotMiddleSq=X
    THEN Win_For_X=Yes         No instances covered: 34
IF TopMiddleSq=0
    BotMiddleSq=0
    THEN Win_For_X=No         No instances covered: 25
IF MidLeftSq=X
    THEN Win_For_X=Yes         No instances covered: 22
IF      THEN Win_For_X=No         No instances covered: 5

```

```

AntRun  NoIterations
1        11
2       3000
3       130
4        23

```

Appendix B Sample Outputs from Implemented System

```

5         13
6         43
7         17
8         10
9         10
    
```

```

Total no instances in test set: 96
No instances in test data with missing values: 0
Predictive accuracy: 69.79166666666667%
    
```

```

*****
k = 6
    
```

```

Total no instances in training data: 862
No instances in training data with missing values: 0
    
```

Final rule set has 9 rules:

```

IF MidMiddleSq=X
    THEN Win_For_X=Yes           No instances covered: 337
IF BotLeftSq=X
    THEN Win_For_X=Yes           No instances covered: 134
IF TopRightSq=0
    THEN Win_For_X=No           No instances covered: 129
IF TopLeftSq=0
    THEN Win_For_X=No           No instances covered: 79
IF BotLeftSq=X
    THEN Win_For_X=No           No instances covered: 31
IF TopRightSq=X
   MidRightSq=X
   BotRightSq=X
    THEN Win_For_X=Yes           No instances covered: 49
IF BotLeftSq=0
   BotMiddleSq=0
   BotRightSq=0
    THEN Win_For_X=No           No instances covered: 27
IF TopLeftSq=X
   TopMiddleSq=X
   TopRightSq=X
    THEN Win_For_X=Yes           No instances covered: 49
IF                               THEN Win_For_X=No           No instances covered: 27
    
```

```

AntRun  NoIterations
1         16
2        3000
3         29
4         42
5         48
6         22
7         34
8         10
9        3000
    
```

```

Total no instances in test set: 96
No instances in test data with missing values: 0
Predictive accuracy: 70.83333333333333%
    
```

*Appendix B Sample Outputs from Implemented System*

\*\*\*\*\*  
k = 7

Total no instances in training data: 862  
No instances in training data with missing values: 0

Final rule set has 5 rules:

IF MidMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 330
IF BotRightSq=0	THEN Win_For_X=No	No instances covered: 131
IF BotLeftSq=X	THEN Win_For_X=Yes	No instances covered: 138
IF TopRightSq=X	THEN Win_For_X=Yes	No instances covered: 99
IF	THEN Win_For_X=No	No instances covered: 164

AntRun	NoIterations
1	22
2	1049
3	88
4	30
5	3000

Total no instances in test set: 96  
No instances in test data with missing values: 0  
Predictive accuracy: 75.0%

\*\*\*\*\*  
k = 8

Total no instances in training data: 862  
No instances in training data with missing values: 0

Final rule set has 9 rules:

IF MidMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 330		
IF TopRightSq=0	THEN Win_For_X=No	No instances covered: 129		
IF TopLeftSq=X	THEN Win_For_X=Yes	No instances covered: 140		
IF BotRightSq=0	THEN Win_For_X=No	No instances covered: 76		
IF TopLeftSq=X	THEN Win_For_X=No	No instances covered: 31		
IF BotLeftSq=X	BotMiddleSq=X	BotRightSq=X	THEN Win_For_X=Yes	No instances covered: 52
IF TopLeftSq=0	MidLeftSq=0	BotLeftSq=0	THEN Win_For_X=No	No instances covered: 27
IF TopRightSq=X	MidRightSq=X	BotRightSq=X	THEN Win_For_X=Yes	No instances covered: 50

Appendix B Sample Outputs from Implemented System

IF THEN Win\_For\_X=No No instances covered: 27

AntRun	NoIterations
1	23
2	79
3	3000
4	24
5	55
6	25
7	32
8	19
9	293

Total no instances in test set: 96  
 No instances in test data with missing values: 0  
 Predictive accuracy: 77.08333333333333%

\*\*\*\*\*  
 k = 9

Total no instances in training data: 862  
 No instances in training data with missing values: 0

Final rule set has 8 rules:

IF MidMiddleSq=0	THEN Win_For_X=No	No instances covered: 179
IF BotLeftSq=X	THEN Win_For_X=Yes	No instances covered: 266
IF BotRightSq=X	THEN Win_For_X=Yes	No instances covered: 151
IF TopMiddleSq=X	THEN Win_For_X=Yes	No instances covered: 104
IF TopMiddleSq=X	THEN Win_For_X=No	No instances covered: 51
IF BotMiddleSq=X	THEN Win_For_X=No	No instances covered: 43
IF MidLeftSq=X	MidMiddleSq=X	MidRightSq=X
	THEN Win_For_X=Yes	No instances covered: 38
IF	THEN Win_For_X=No	No instances covered: 30

AntRun	NoIterations
1	19
2	18
3	25
4	76
5	305
6	38
7	20
8	3000

Total no instances in test set: 96  
 No instances in test data with missing values: 0  
 Predictive accuracy: 62.5%

*Appendix B Sample Outputs from Implemented System*

\*\*\*\*\*  
k = 10

Total no instances in training data: 864  
No instances in training data with missing values: 0

Final rule set has 8 rules:

```

IF MidMiddleSq=X
    THEN Win_For_X=Yes           No instances covered: 330
IF TopLeftSq=0
    THEN Win_For_X=No           No instances covered: 133
IF BotLeftSq=X
    THEN Win_For_X=Yes           No instances covered: 138
IF TopRightSq=0
    THEN Win_For_X=No           No instances covered: 81
IF TopRightSq=X
    MidRightSq=X
    BotRightSq=X
    THEN Win_For_X=Yes           No instances covered: 50
IF TopMiddleSq=0
    THEN Win_For_X=No           No instances covered: 33
IF TopLeftSq=X
    TopMiddleSq=X
    TopRightSq=X
    THEN Win_For_X=Yes           No instances covered: 50
IF
    THEN Win_For_X=No           No instances covered: 49
  
```

AntRun	NoIterations
1	18
2	58
3	134
4	18
5	43
6	27
7	26
8	3000

Total no instances in test set: 94  
No instances in test data with missing values: 0  
Predictive accuracy: 74.46808510638297%

\*\*\*\*\*  
\*\*\*\*\*

k	NoAntRuns	NoItns	NoAnts	PredAcc	NoTerms	NoRules
1	9	766	766	81.2500	14	9
2	9	316	316	75.0000	11	10
3	8	464	464	77.0833	11	9
4	8	5894	5894	73.9583	11	8
5	9	3257	3257	69.7917	12	10
6	9	6201	6201	70.8333	14	9
7	5	4189	4189	75.0000	4	5
8	9	3550	3550	77.0833	14	9
9	8	3501	3501	62.5000	9	8
10	8	3324	3324	74.4681	11	8

	Average	StdDeviation
PredAcc	73.6968	5.0866
Rules	8.5000	1.4337
Terms	11.1000	2.9981

Time taken: 5 mins

## **Analysis of Ljubljana Dataset**

The next output is from Test 1 – Comparison with Ant-Miner on the Ljubljana data set. Note that this sample output provides the basis for the summary statistics shown in the first row of the first table in Appendix C Sample Detailed Test Results.

*Appendix B Sample Outputs from Implemented System*

Fri Sep 06 15:21:22 BST 2002

LBCancer - 10-fold cross validation

PARAMETER SETTINGS:

No Ants per Iteration: 1  
Max no iterations: 3000  
Min no instances covered by rule: 10  
Max no instances left uncovered: 10  
Max no rules allowed to converge: 10  
Transition rule: random proportional selection

TRAINING DATA DETAILS:

No attributes inc predictor: 10  
No class labels: 2  
No terms in problem space: 56

Seed for this run: 1029507256679

\*\*\*\*\*  
k = 1

Total no instances in training data: 249  
No instances in training data with missing values: 0

Final rule set has 8 rules:

IF Inv-nodes=0		
THEN Recurrence=no		No instances covered: 114
IF Breast-quad=left-upper		
THEN Recurrence=no		No instances covered: 28
IF Menopause=premenopausal		
Irradiation=no		
THEN Recurrence=yes		No instances covered: 30
IF Menopause=>=40		
THEN Recurrence=yes		No instances covered: 28
IF Irradiation=no		
THEN Recurrence=no		No instances covered: 22
IF Age=40-49		
Node-caps=no		
THEN Recurrence=no		No instances covered: 8
IF Menopause=premenopausal		
Breast=left		
THEN Recurrence=yes		No instances covered: 8
IF	THEN Recurrence=no	No instances covered: 11

AntRun	NoIterations
1	19
2	11
3	41
4	27
5	19
6	26
7	19
8	10

Total no instances in test set: 28  
No instances in test data with missing values: 0

Appendix B Sample Outputs from Implemented System

Predictive accuracy: 71.42857142857143%

\*\*\*\*\*  
k = 2

Total no instances in training data: 248  
No instances in training data with missing values: 0

Final rule set has 8 rules:

IF Inv-nodes=0	THEN Recurrence=no	No instances covered: 115
IF Node-caps=no	Irradiation=no	
	THEN Recurrence=yes	No instances covered: 35
IF Deg-malig=2	THEN Recurrence=no	No instances covered: 39
IF Tumour-size=30-34	THEN Recurrence=yes	No instances covered: 12
IF Node-caps=yes	THEN Recurrence=yes	No instances covered: 17
IF Irradiation=no	THEN Recurrence=no	No instances covered: 12
IF Age=40-49	THEN Recurrence=no	No instances covered: 7
IF	THEN Recurrence=yes	No instances covered: 11

AntRun	NoIterations
1	14
2	33
3	1365
4	40
5	23
6	14
7	29
8	404

Total no instances in test set: 29  
No instances in test data with missing values: 0  
Predictive accuracy: 72.41379310344827%

\*\*\*\*\*  
k = 3

Total no instances in training data: 249  
No instances in training data with missing values: 0

Final rule set has 9 rules:

IF Inv-nodes=0	THEN Recurrence=no	No instances covered: 121
IF Deg-malig=3	THEN Recurrence=yes	No instances covered: 40
IF Inv-nodes=0	THEN Recurrence=yes	No instances covered: 11
IF Deg-malig=3	THEN Recurrence=no	No instances covered: 16
IF Menopause=>=40		

*Appendix B Sample Outputs from Implemented System*

```

                THEN Recurrence=no           No instances covered: 18
IF Menopause=premenopausal
  Node-caps=no
                THEN Recurrence=no           No instances covered: 17
IF Node-caps=no
                THEN Recurrence=yes          No instances covered: 13
IF Menopause=premenopausal
                THEN Recurrence=no           No instances covered: 7
IF                THEN Recurrence=yes       No instances covered: 6

```

```

AntRun  NoIterations
1       17
2       37
3       22
4       21
5       25
6       22
7       12
8       10

```

Total no instances in test set: 28  
 No instances in test data with missing values: 0  
 Predictive accuracy: 67.85714285714286%

```

*****
k = 4

```

Total no instances in training data: 248  
 No instances in training data with missing values: 0

```

Final rule set has 9 rules:
IF Inv-nodes=0
  THEN Recurrence=no           No instances covered: 116
IF Deg-malig=3
  THEN Recurrence=yes          No instances covered: 40
IF Inv-nodes=0
  THEN Recurrence=yes          No instances covered: 14
IF Age=50-59
  THEN Recurrence=no           No instances covered: 19
IF Breast=right
  Irradiation=yes
  THEN Recurrence=no           No instances covered: 13
IF Node-caps=no
  Breast=left
  THEN Recurrence=no           No instances covered: 15
IF Irradiation=yes
  THEN Recurrence=yes          No instances covered: 11
IF Menopause=premenopausal
  THEN Recurrence=yes          No instances covered: 9
IF                THEN Recurrence=no       No instances covered: 11

```

```

AntRun  NoIterations
1       21
2       72
3       30
4       32

```

Appendix B Sample Outputs from Implemented System

5 25  
6 23  
7 19  
8 10  
9 10

Total no instances in test set: 29  
No instances in test data with missing values: 0  
Predictive accuracy: 68.96551724137932%

\*\*\*\*\*  
k = 5

Total no instances in training data: 249  
No instances in training data with missing values: 0

Final rule set has 8 rules:

IF Inv-nodes=0	THEN Recurrence=no	No instances covered: 113
IF Deg-malig=3	THEN Recurrence=yes	No instances covered: 40
IF Inv-nodes=0	THEN Recurrence=yes	No instances covered: 14
IF Breast-quad=left-upper	THEN Recurrence=no	No instances covered: 28
IF Breast=left	THEN Recurrence=no	No instances covered: 21
IF Menopause=premenopausal Breast=right	THEN Recurrence=no	No instances covered: 12
IF Menopause=premenopausal	THEN Recurrence=yes	No instances covered: 13
IF	THEN Recurrence=yes	No instances covered: 8

AntRun	NoIterations
1	19
2	18
3	29
4	57
5	2539
6	23
7	17

Total no instances in test set: 28  
No instances in test data with missing values: 0  
Predictive accuracy: 75.0%

\*\*\*\*\*  
k = 6

Total no instances in training data: 249  
No instances in training data with missing values: 0

Final rule set has 8 rules:

IF Inv-nodes=0	THEN Recurrence=no	No instances covered: 109
----------------	--------------------	---------------------------

Appendix B Sample Outputs from Implemented System

```
IF Node-caps=no
  Irradiation=no
    THEN Recurrence=yes      No instances covered: 33
IF Node-caps=no
  THEN Recurrence=no        No instances covered: 41
IF Deg-malig=3
  THEN Recurrence=yes      No instances covered: 28
IF Node-caps=yes
  Breast=right
    THEN Recurrence=no      No instances covered: 15
IF Irradiation=yes
  THEN Recurrence=yes      No instances covered: 11
IF Breast=left
  THEN Recurrence=no      No instances covered: 9
IF      THEN Recurrence=yes No instances covered: 3
```

```
AntRun  NoIterations
1       17
2       42
3       18
4       25
5       36
6       15
7       11
```

Total no instances in test set: 28  
No instances in test data with missing values: 0  
Predictive accuracy: 89.28571428571429%

\*\*\*\*\*  
k = 7

Total no instances in training data: 249  
No instances in training data with missing values: 0

Final rule set has 8 rules:

```
IF Inv-nodes=0
  THEN Recurrence=no      No instances covered: 116
IF Node-caps=no
  Irradiation=no
    THEN Recurrence=yes    No instances covered: 33
IF Node-caps=no
  THEN Recurrence=no      No instances covered: 36
IF Deg-malig=3
  THEN Recurrence=yes    No instances covered: 28
IF Node-caps=yes
  Breast=right
    THEN Recurrence=no    No instances covered: 14
IF Irradiation=yes
  THEN Recurrence=yes    No instances covered: 10
IF Breast=left
  THEN Recurrence=no     No instances covered: 9
IF      THEN Recurrence=yes No instances covered: 3
```

```
AntRun  NoIterations
1       14
2       51
```

Appendix B Sample Outputs from Implemented System

3 128  
4 21  
5 24  
6 34  
7 13

Total no instances in test set: 28  
No instances in test data with missing values: 0  
Predictive accuracy: 75.0%

\*\*\*\*\*  
k = 8

Total no instances in training data: 251  
No instances in training data with missing values: 0

Final rule set has 9 rules:

IF Inv-nodes=0  
    THEN Recurrence=no           No instances covered: 111  
IF Deg-malig=3  
    THEN Recurrence=yes        No instances covered: 42  
IF Inv-nodes=0  
    THEN Recurrence=yes        No instances covered: 13  
IF Age=50-59  
    THEN Recurrence=no        No instances covered: 22  
IF Breast=right  
    Irradiation=yes  
    THEN Recurrence=no        No instances covered: 18  
IF Node-caps=no  
    Breast=left  
    THEN Recurrence=no        No instances covered: 15  
IF Irradiation=yes  
    THEN Recurrence=yes        No instances covered: 11  
IF Menopause=premenopausal  
    THEN Recurrence=yes        No instances covered: 8  
IF        THEN Recurrence=no    No instances covered: 11

AntRun   NoIterations  
1        10  
2        50  
3        28  
4        52  
5        56  
6        20  
7        17  
8        10  
9        587

Total no instances in test set: 26  
No instances in test data with missing values: 0  
Predictive accuracy: 84.61538461538461%

\*\*\*\*\*  
k = 9

Total no instances in training data: 249

*Appendix B Sample Outputs from Implemented System*

No instances in training data with missing values: 0

Final rule set has 8 rules:

```
IF Inv-nodes=0
    THEN Recurrence=no          No instances covered: 112
IF Menopause=premenopausal
    Irradiation=no
    THEN Recurrence=yes        No instances covered: 31
IF Deg-malig=2
    THEN Recurrence=no          No instances covered: 40
IF Menopause=>=40
    THEN Recurrence=yes        No instances covered: 26
IF Irradiation=no
    THEN Recurrence=no          No instances covered: 17
IF Breast=left
    THEN Recurrence=yes        No instances covered: 9
IF Deg-malig=3
    THEN Recurrence=no          No instances covered: 7
IF      THEN Recurrence=yes    No instances covered: 7
```

AntRun NoIterations

```
1      22
2      36
3     2101
4      18
5      10
6      20
7      10
```

Total no instances in test set: 28

No instances in test data with missing values: 0

Predictive accuracy: 78.57142857142857%

\*\*\*\*\*

k = 10

Total no instances in training data: 252

No instances in training data with missing values: 0

Final rule set has 8 rules:

```
IF Inv-nodes=0
    THEN Recurrence=no          No instances covered: 116
IF Deg-malig=3
    THEN Recurrence=yes        No instances covered: 40
IF Inv-nodes=0
    THEN Recurrence=yes        No instances covered: 12
IF Breast-quad=left-upper
    THEN Recurrence=no          No instances covered: 28
IF Breast=left
    THEN Recurrence=no          No instances covered: 22
IF Menopause=premenopausal
    Irradiation=no
    THEN Recurrence=yes        No instances covered: 9
IF Menopause=premenopausal
    Breast=right
    THEN Recurrence=no          No instances covered: 12
```

Appendix B Sample Outputs from Implemented System

IF THEN Recurrence=yes No instances covered: 13

AntRun	NoIterations
1	15
2	45
3	22
4	99
5	23
6	37
7	13
8	10

Total no instances in test set: 25  
No instances in test data with missing values: 0  
Predictive accuracy: 72.0%

\*\*\*\*\*  
\*\*\*\*\*

k	NoAlgs	NoItns	NoAnts	PredAcc	NoTerms	NoRules
1	8	172	172	71.4286	10	8
2	8	1922	1922	72.4138	8	8
3	8	166	166	67.8571	9	9
4	9	242	242	68.9655	10	9
5	7	2702	2702	75.0000	8	8
6	7	164	164	89.2857	9	8
7	7	285	285	75.0000	9	8
8	9	830	830	84.6154	10	9
9	7	2217	2217	78.5714	8	8
10	8	264	264	72.0000	9	8

	Average	StdDeviation
PredAcc	75.5138	6.8519
Rules	8.3000	0.4830
Terms	9.0000	0.8165

Time taken: 0 mins

## Appendix C Sample Detailed Test Results

As mentioned in Section **6.1 Experimental Methodology**, the summary tables in **Chapter 7 Results and Analyses** contain figures based on *ten* ten-fold cross-validation results. This appendix contains the summary results from all the ten ten-fold cross-validation runs for **Test 3 – Changing the Transition Rule** on the Ljubljana Breast Cancer data set.

This test required the changing of a parameter that sets the balance between exploration and exploitation, called the *q-value*. Eleven different values for this parameter were tried ranging from  $q=0.0$  (i.e. no exploitation of previously learnt information) to  $q=1.0$  (i.e. no exploration, turning the algorithm into a purely deterministic one). Ten ten-fold cross-validations were done for each of the *q-value* settings..

Each table following represents the results obtained for one of the *q-value* settings. Note that each row in these tables is a summary one ten-fold cross-validation run – an example of which may be seen in **Appendix B Sample Outputs from Implemented System**. The second sample of outputs in Appendix B shows specifically how the figures in the first row of the first table below were obtained.

The first column of a table numbers each of the ten ten-fold cross-validations from one to ten. The second column gives the predictive accuracy obtained by averaging the ten results for a ten-fold cross-validation, while third is the resulting standard deviation from the ten individual predictive accuracy figures.

The fourth column is the average number of rules in a rule set with the fifth the associated standard deviation. The sixth column is the average number of terms in a rule set with the seventh being the associated standard deviation. Column eight is column seven divided by column six and gives an approximate number of terms for a rule.

The last column gives an indication of the time taken rounded to the nearest minute for a ten-fold cross-validation.

The last row in a table gives an average of the various columns. It is these figures that appear in the summary tables of **Chapter 7 Results and Analyses**.

Appendix C Sample Detailed Test Results

**q = 0.0**

Exp	AvePred	StdDvn	AveRules	StdDvn	AveTerms	StdDvn	Ratio	Time
1	75.5138	6.8519	8.30	0.4830	9.00	0.8165	1.08	0.0
2	70.7826	6.5369	8.40	0.6992	8.90	0.8756	1.06	0.0
3	72.9162	6.2566	8.30	1.0593	8.50	1.1785	1.02	0.0
4	69.6560	6.7365	8.40	1.0750	8.80	1.3166	1.05	0.0
5	72.1834	9.8307	8.40	0.9661	9.10	1.4491	1.08	0.0
6	73.2579	7.7575	7.90	1.1972	8.30	1.4944	1.05	0.0
7	73.6028	6.6262	7.90	0.8756	8.90	2.0248	1.13	0.0
8	72.5834	8.2639	8.10	0.8756	8.50	1.2693	1.05	0.0
9	71.5395	8.3923	8.60	0.9661	9.20	1.4757	1.07	0.0
10	73.2579	8.2759	8.00	0.8165	8.80	1.4757	1.10	0.0
<i>Average</i>	<i>72.5294</i>	<i>7.5528</i>	<i>8.23</i>	<i>0.9014</i>	<i>8.80</i>	<i>1.3376</i>	<i>1.07</i>	<i>0.0</i>

**q = 0.1**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	71.4235	8.5068	8.00	0.9428	8.70	1.8288	1.09	1.0
2	73.9874	9.0435	8.30	0.8233	8.90	1.5951	1.07	1.0
3	73.2703	6.4571	8.00	0.9428	8.30	1.4944	1.04	0.0
4	73.8924	7.1822	8.40	0.8433	8.90	1.1005	1.06	1.0
5	74.7812	6.9373	8.60	1.1738	9.30	1.4181	1.08	0.0
6	73.5874	8.4676	7.80	0.7888	8.30	0.8233	1.06	1.0
7	73.2520	8.1215	7.80	0.6325	8.10	1.6633	1.04	1.0
8	72.2171	4.8068	8.50	0.9718	8.90	1.2867	1.05	0.0
9	72.1683	7.5718	7.90	0.5676	8.60	1.1738	1.09	0.0
10	72.2263	9.0456	8.00	0.9428	8.90	2.1833	1.11	0.0
<i>Average</i>	<i>73.0806</i>	<i>7.6140</i>	<i>8.13</i>	<i>0.8629</i>	<i>8.69</i>	<i>1.4567</i>	<i>1.07</i>	<i>0.5</i>

Appendix C Sample Detailed Test Results

**q = 0.2**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	75.0617	9.0688	7.80	0.7888	8.30	1.2517	1.06	0.0
2	72.9529	9.0197	8.10	0.8756	8.30	1.1595	1.02	0.0
3	72.5560	10.0369	8.50	0.9718	8.20	1.0328	0.96	0.0
4	75.3392	8.3575	7.90	0.8756	7.90	0.9944	1.00	1.0
5	71.4538	9.8905	8.40	0.8433	9.10	1.8529	1.08	0.0
6	74.3294	8.4972	7.70	0.9487	7.80	1.1353	1.01	0.0
7	72.0887	7.9682	8.10	1.2867	8.50	1.6499	1.05	0.0
8	74.3263	8.7745	7.70	1.0593	7.80	1.2293	1.01	1.0
9	71.3649	8.6928	7.60	0.6992	8.20	1.8738	1.08	0.0
10	73.5751	6.8629	7.70	0.6749	7.90	0.9944	1.03	1.0
<i>Average</i>	<i>73.3048</i>	<i>8.7169</i>	<i>7.95</i>	<i>0.9024</i>	<i>8.20</i>	<i>1.3174</i>	<i>1.03</i>	<i>0.3</i>

**q = 0.3**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	71.7834	10.0116	7.70	0.6749	8.10	1.1005	1.05	0.0
2	74.6894	5.9648	8.30	0.8233	9.10	1.4491	1.10	0.0
3	73.2977	8.6253	8.10	0.9944	8.70	1.3375	1.07	0.0
4	73.9815	8.8705	8.00	1.0541	8.20	2.0440	1.03	1.0
5	71.5092	7.3132	7.30	1.1595	7.40	1.7127	1.01	1.0
6	73.6613	11.5122	8.60	1.1738	8.80	1.3984	1.02	0.0
7	72.5529	8.5548	7.70	1.1595	8.10	2.0790	1.05	0.0
8	71.1579	8.5127	7.60	0.5164	7.70	1.1595	1.01	0.0
9	74.3047	6.6070	8.30	1.1595	8.90	1.6633	1.07	1.0
10	73.2151	7.7931	8.00	0.8165	8.70	1.6364	1.09	0.0
<i>Average</i>	<i>73.0153</i>	<i>8.3765</i>	<i>7.96</i>	<i>0.9532</i>	<i>8.37</i>	<i>1.5580</i>	<i>1.05</i>	<i>0.3</i>

Appendix C Sample Detailed Test Results

**q = 0.4**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	75.0160	7.1583	7.90	1.2867	8.80	1.6193	1.11	1.0
2	72.2417	6.5923	7.90	0.5676	8.40	0.8433	1.06	0.0
3	73.6395	8.4641	8.20	1.0328	8.60	1.4298	1.05	0.0
4	71.0969	5.8074	8.10	0.9944	8.00	1.1547	0.99	0.0
5	71.7560	9.9306	8.10	0.8756	9.00	1.6997	1.11	0.0
6	73.2120	9.6757	7.80	0.4216	8.40	0.9661	1.08	0.0
7	75.7577	9.1462	8.60	1.0750	8.80	1.3166	1.02	0.0
8	71.8512	10.8390	7.60	0.5164	7.70	0.9487	1.01	0.0
9	75.4280	7.7125	7.90	0.9944	8.00	1.2472	1.01	0.0
10	72.8182	9.7258	7.90	0.7379	8.50	1.5811	1.08	1.0
<i>Average</i>	<i>73.2817</i>	<i>8.5052</i>	<i>8.00</i>	<i>0.8502</i>	<i>8.42</i>	<i>1.2806</i>	<i>1.05</i>	<i>0.2</i>

**q = 0.5**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	69.3235	8.9406	7.90	0.7379	8.20	1.4757	1.04	0.0
2	75.0711	7.7794	8.70	1.3375	9.10	1.5239	1.05	0.0
3	72.9098	8.8366	7.90	1.1005	8.50	1.2693	1.08	0.0
4	71.7894	7.4922	8.30	1.2517	8.50	1.7795	1.02	1.0
5	72.5283	9.1876	7.80	0.7888	8.40	1.8379	1.08	0.0
6	72.1131	10.5343	7.50	0.8498	7.50	1.0801	1.00	0.0
7	71.8417	9.9563	8.20	0.7888	8.00	0.9428	0.98	0.0
8	72.8579	7.0181	8.00	1.0541	8.50	1.7159	1.06	0.0
9	72.8456	6.6833	7.80	0.9189	8.20	1.2293	1.05	0.0
10	70.6722	8.6521	7.80	0.4216	8.40	0.5164	1.08	0.0
<i>Average</i>	<i>72.1953</i>	<i>8.5080</i>	<i>7.99</i>	<i>0.9250</i>	<i>8.33</i>	<i>1.3371</i>	<i>1.04</i>	<i>0.1</i>

**q = 0.6**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	72.8608	8.8478	7.80	0.7888	8.00	1.4142	1.03	0.0
2	71.8017	8.1733	8.30	1.4944	8.60	1.6465	1.04	0.0
3	71.0969	9.8022	7.80	0.7888	8.20	1.6193	1.05	0.0
4	72.9834	8.4777	8.20	1.3166	8.50	1.8409	1.04	0.0
5	72.8608	9.7519	7.70	0.9487	8.20	1.8738	1.06	0.0
6	72.8608	8.8478	7.90	1.1972	7.90	1.4491	1.00	0.0
7	72.1560	9.1705	7.60	0.6992	8.00	1.4907	1.05	0.0
8	73.9874	9.0435	8.10	0.9944	8.70	1.0593	1.07	0.0
9	72.4210	9.3206	7.80	0.7888	8.20	1.6193	1.05	0.0
10	73.6120	10.1833	7.90	0.9944	8.20	1.5492	1.04	0.0
<i>Average</i>	<i>72.6641</i>	<i>9.1619</i>	<i>7.91</i>	<i>1.0011</i>	<i>8.25</i>	<i>1.5562</i>	<i>1.04</i>	<i>0.0</i>

**q = 0.7**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	73.5322	9.9712	7.70	0.6749	8.20	1.3166	1.06	0.0
2	69.6377	9.4900	8.00	0.6667	8.30	1.2517	1.04	0.0
3	71.7711	10.0545	7.60	0.5164	7.90	0.9944	1.04	0.0
4	72.8056	6.8115	8.40	0.9661	8.70	1.4944	1.04	0.0
5	72.1834	9.2568	7.50	0.8498	7.90	1.5951	1.05	0.0
6	73.2274	7.5497	7.90	0.7379	8.30	1.1595	1.05	0.0
7	72.5254	7.2355	7.90	1.1005	8.30	1.4944	1.05	0.0
8	73.2549	6.8691	7.80	0.6325	8.10	0.9944	1.04	0.0
9	71.4812	10.0372	7.70	0.6749	7.80	1.2293	1.01	0.0
10	73.2672	10.8056	8.30	1.4181	8.40	2.1705	1.01	0.0
<i>Average</i>	<i>72.3686</i>	<i>8.8081</i>	<i>7.88</i>	<i>0.8238</i>	<i>8.19</i>	<i>1.3700</i>	<i>1.04</i>	<i>0.0</i>

Appendix C Sample Detailed Test Results

**q = 0.8**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	71.1243	9.2952	8.00	0.9428	8.30	1.7029	1.04	0.0
2	71.4938	10.0521	7.20	1.0328	7.50	1.5811	1.04	0.0
3	71.7036	9.5329	8.10	0.9944	8.10	1.1005	1.00	0.0
4	71.4692	8.6592	7.90	1.2867	8.00	1.4907	1.01	0.0
5	71.1366	9.6886	7.50	0.7071	7.90	1.2867	1.05	0.0
6	72.8426	10.0305	7.90	1.1005	8.20	1.5492	1.04	0.0
7	74.6465	6.2533	8.30	0.8233	8.20	1.0328	0.99	0.0
8	70.3520	9.8997	7.60	0.6992	7.80	1.1353	1.03	0.0
9	71.1366	9.6886	7.70	0.8233	8.00	1.3333	1.04	0.0
10	71.4815	9.5264	7.70	0.6749	8.10	1.5951	1.05	0.0
<i>Average</i>	<i>71.7387</i>	<i>9.2626</i>	<i>7.79</i>	<i>0.9085</i>	<i>8.01</i>	<i>1.3808</i>	<i>1.03</i>	<i>0.0</i>

**q = 0.9**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	71.8111	9.6644	7.40	0.8433	7.90	1.9120	1.07	0.0
2	71.4417	9.0472	7.70	0.8233	8.00	1.5635	1.04	0.0
3	71.8386	10.0340	7.40	0.8433	7.60	1.1738	1.03	0.0
4	71.8263	9.2048	7.70	0.9487	8.10	1.7920	1.05	0.0
5	71.4692	8.9806	7.20	0.6325	7.30	1.1595	1.01	0.0
6	71.0969	7.8907	8.00	1.0541	8.40	1.5776	1.05	0.0
7	72.5160	9.2250	7.50	0.9718	7.50	1.5092	1.00	0.0
8	71.4815	9.3867	7.20	0.6325	7.30	1.0593	1.01	0.0
9	70.7397	8.9600	7.30	0.8233	7.30	1.1595	1.00	0.0
10	72.5131	9.6592	7.50	0.5270	7.80	1.0328	1.04	0.0
<i>Average</i>	<i>71.6734</i>	<i>9.2053</i>	<i>7.49</i>	<i>0.8100</i>	<i>7.72</i>	<i>1.3939</i>	<i>1.03</i>	<i>0.0</i>

Appendix C Sample Detailed Test Results

**q = 1.0**

Exp	AvePred	StdDvn	AvNRules	StdDvn	AvNTerms	StdDvn	Ratio	Time
1	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
2	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
3	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
4	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
5	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
6	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
7	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
8	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
9	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
10	70.7379	8.9600	7.10	0.5676	7.00	0.8165	0.99	0.0
<i>Average</i>	<i>70.7379</i>	<i>8.9600</i>	<i>7.1000</i>	<i>0.5676</i>	<i>7.0000</i>	<i>0.8165</i>	<i>0.9859</i>	<i>0.0000</i>

## Appendix D Comparison with Revised Ant-Miner

### Results

Parpinelli *et al* have revised some of their figures published in [Parpinelli et al, 2002a] and reproduced in Section 7.1 for comparison with this implementation's results.

The revised estimates are found in [Parpinelli et al, 2002b], to appear in a special issue of the *IEEE Transactions on Evolutionary Computation*, 2002. These estimates have been reproduced here for a brief comparison with this project's implementation results.

The following table and graph compare the predictive accuracy.

Dataset	Revised Ant-Miner		This Implementation	
	Predictive Accuracy (%)	Standard Deviation (+/-)	Average Predictive Accuracy (%)	Average Standard Deviation (+/-)
Ljubljana Breast Cancer	75.28	2.24	72.53	7.55
Wisconsin Breast Cancer	96.04	0.93	92.22	3.35
Tic-Tac-Toe	73.04	2.53	74.20	4.43
Dermatology	94.29	1.20	89.84	6.19
Hepatitis	90.00	3.11	84.33	10.93
Cleveland Heart Disease	59.67	2.50	56.00	8.48

Table D 1 Comparing Predictive Accuracy with Revised Ant-Miner Estimates

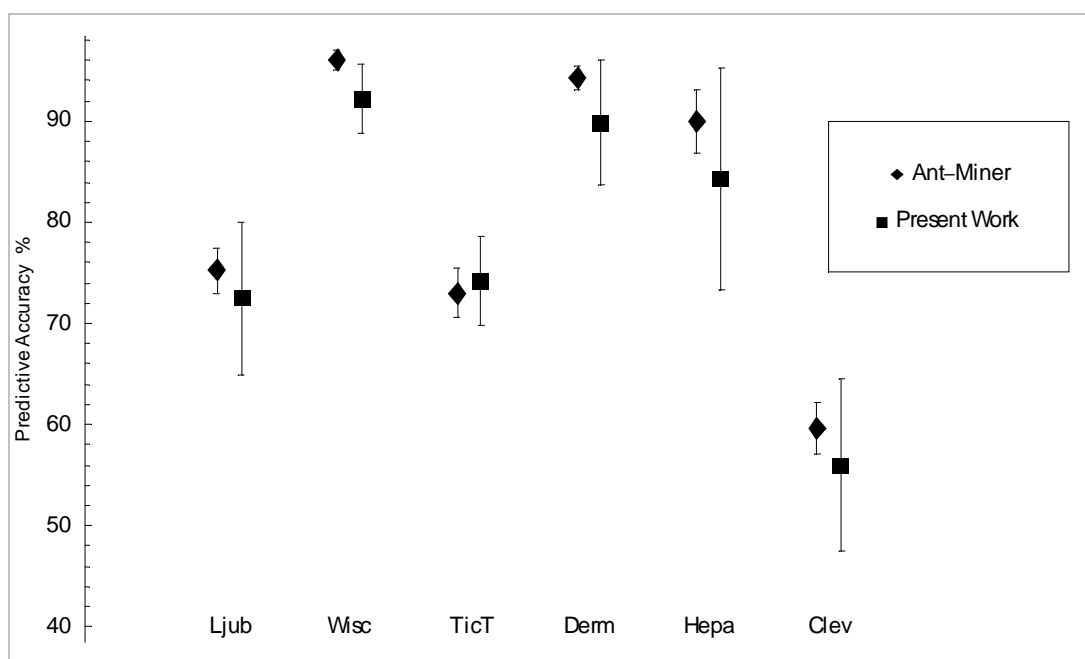


Figure D.1 Comparing Predictive Accuracy with Revised Ant-Miner Estimates

There are two points worth taking note of:

- Parpinelli *et al* have considerably improved the predictive accuracy for the Dermatology data set – 94.29% as opposed to their original 86.55%;
- The revised standard deviations are also considerably smaller than in Table 7.1, with the revised figures occasionally being as much as four times smaller than the original estimates.

Keeping in mind that only one ten-fold cross-validation has been run for the revised Ant-Miner estimates, on average Ant-Miner's predictive accuracy is higher than this implementation's, with the exception of the Tic-Tac-Toe data set. However, the best predictive accuracy produced by this implementation – taking into account the revised standard deviations – is now greater than Ant-Miner's for all the data sets with the exception of the Wisconsin Breast Cancer data set.

There has been no time to discuss these revised figures with Dr Parpinelli, no explanation is provided for the revisions made.

Appendix D Comparison with Revised Ant-Miner Results

Dataset	Revised Ant-Miner			This Implementation		
	No. Rules in Ruleset	Standard Deviation (+/-)	No. Terms in a Rule	Average No. Rules in Ruleset	Average Standard Deviation (+/-)	Average No. Terms in a Rule
Ljubljana Breast Cancer	7.10	0.31	1.28	8.23	0.90	1.07
Wisconsin Breast Cancer	6.20	0.25	1.97	11.51	0.86	1.01
Tic-Tac-Toe	8.50	0.62	1.18	8.12	1.52	1.26
Dermatology	7.30	0.15	3.16	7.40	0.54	2.83
Hepatitis	3.40	0.16	2.41	3.98	0.06	1.51
Cleveland Heart Disease	9.50	0.92	1.71	14.82	0.97	1.84

Table D 2 Comparing Rule Sets and Rules with Revised Ant-Miner Estimates

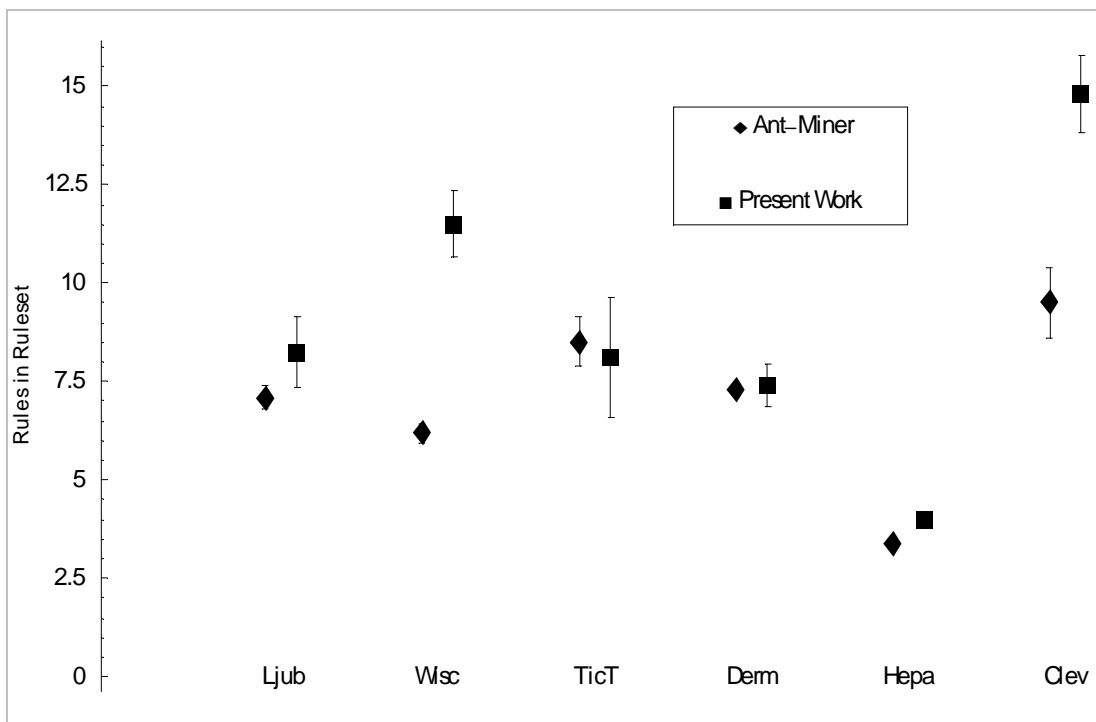


Figure D 2 Comparing Rule Sets with Revised Ant-Miner Estimates

Note that:

- The above table, when compared with Table 7.2, shows a considerable revision to the average number of terms in a rule for the Dermatology data set – 3.16 terms vs the original of 11.57 terms;
- The standard deviations for the rule set size have also decreased, by as much as three times for the Wisconsin Breast Cancer, Tic-Tac-Toe and Hepatitis data sets.

These revisions have minimal impact when compared with this implementation's results on rule set and rule sizes; the same conclusions drawn for Test 1 – Comparison with Ant-Miner still hold.