SIGNIFICANT GENETIC ALGORITHM IN FEATURE SELECTION, RULE DISCOVERY AND THE USE OF STATISTICAL TOOLS IN OPTIMIZATION

THESIS

Submitted by
V.N.RAJAVARMAN

For the award of the Degree
of
DOCTOR OF PHILOSOPHY

Guided by
Dr. S.P.RAJAGOPALAN
Professor Emeritus

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr. M.G.R. EDUCATIONAL AND RESEARCH INSTITUTE
(University Declared U/S 3 of the UGC Act 1956)
Chennai - 600 095

July - 2010

BONAFIDE CERTIFICATE

Certified that this thesis titled **"SIGNIFICANT GENETIC ALGORITHM IN FEATURE SELECTION, RULE DISCOVERY AND THE USE OF STATISTICAL TOOLS IN OPTIMIZATION"** is the bonafide work of **Mr.V.N.RAJAVARMAN,** who carried out the research under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. S.P.RAJAGOPALAN
Professor Emeritus
School of Computer Science & Engineering
Dr.M.G.R Educational & Research Institute
(University decl.u/s.3of UGC Act 1956)
Maduravoyal, Chennai – 600 095.

DECLARATION

I declare that the thesis entitled "**SIGNIFICANT GENETIC ALGORITHM IN FEATURE SELECTION, RULE DISCOVERY AND THE USE OF STATISTICAL TOOLS IN OPTIMIZATION"** submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me under the Faculty of Engineering and Technology in the Department of Computer Science and Engineering, under the guidance and supervision of **Dr. S. P. RAJAGOPALAN** during the period from August 2005 to July 2010 and that any part of the thesis has not formed the basis for the award of any degree, diploma, associate ship, fellowship or other titles in this University or any other University or other Institution of higher learning.

Signature of the Candidate
V.N.RAJAVARMAN
Research Scholar
Department of Computer Science and Engineering

Abstract

Genetic algorithm is one of the commonly used approaches on data mining. In this thesis, a genetic algorithm approach has been applied for classification problems. Binary coding has been adopted in which an individual in a population consists of a fixed number of rules that stand for a solution candidate. The evaluation function considers four important factors which are error rate, entropy measure, rule consistency and hole ratio, respectively. Adaptive asymmetric mutation has been applied by the self-adaptation of mutation inversion probability from 1-0 (0-1). The generated rules are not disjoint but can overlap. The final conclusion for prediction has been based on the voting of rules and the classifier gives all rules equal weight for their votes. Based on three databases, the scholar has compared his approach with several other traditional data mining techniques including decision trees, neural networks and naive bayes learning. The results show that his approach outperformed others on both the prediction accuracy and the standard deviation.

It has been discovered that genetic features and environmental factors were involved in multifactorial diseases. To exploit the massive data obtained from the experiments conducted at the General Hospital, Chennai, data mining tools were required and the researcher has proposed a 2-Phase approach using a specific genetic algorithm. This heuristic approach had been chosen as the number of features to consider was large (up to 3654 for biological data under our study). Collected data indicated for pairs of affected individuals of a same family their similarity at given points (locus) of their chromosomes. This was represented in a matrix where each locus was represented by a column and each pairs of individuals considered by a row. The objective was first to isolate the most relevant associations of features, and then to class individuals that had the considered disease according to these associations. For the first phase, the feature selection problem, a genetic algorithm (GA) has been used. To deal with this very specific problem, some advanced mechanisms had been introduced in the genetic algorithm such as sharing, random immigrant, dedicated genetic operators and a

particular distance operator had been defined. Then, the second phase, a clustering based on the features selected during the previous phase, will use the clustering algorithm K-means.

An approach to GA efficiency improvement has been developed in this thesis. This approach is based on using statistical methods of inference in order to: (1) identify GA factors significantly affecting the efficiency of Genetic Algorithms, and (2) to find the optimal values for such factors. It will be shown that after substitution of the optimal values obtained using the developed approach; the search speed of Genetic Algorithms substantially improves.

ACKNOWLEDGEMENTS

I am greatly indebted to a number of people who, through their support, comments, criticism and guidance, have kept me enthused and helped me transform the objective to reality.

I sincerely submit my respectful gratitude to our **Chancellor Thiru. A.C. Shanmugam** B.A., B.L., Dr. M.G.R. Educational & Research Institute University, Maduravoyal, Chennai for the facilities provided for my Ph.D. thesis.

I am equally indebted to our **Vice President Er. A.C.S. Arun Kumar**, for his generous and continuous patronage throughout my work.

I convey my gratitude to our **Advisor-Administration Prof. R.M.Vasagam** for his valuable guidance.

I express my gratitude to our **Advisor-Academics Dr. P.T. Manoharan**, for his valuable advice.

I express my profound sense of gratitude with pleasure and pride to our **Vice Chancellor Dr. M.K. Padmanabhan** for his valuable guidance.

I convey my special thanks to our **Registrar Dr. S. Dinakaran,** and **Additional Registrar Dr. C. B. Palanivelu** for their valuable encouragement during my whole work.

I am grateful to our **Controller of Examination Dr. P. Sivakesan and Dean E&T Dr.Uma Rajaram** for their valuable directions for my work.

I would like to express my gratitude to our **Dean–Research Dr. P. Aravindan** for his valuable suggestions throughout the study.

It has been Privilege and pleasure to work under my **Research Guide Dr. S.P. Rajagopalan, Professor Emeritus, School of Computer Science and Engineering**, Dr.M.G.R. University, whose help, advice, supervision, and crucial contribution, stimulating suggestions and encouragement which made me a backbone of this research, helped me in all the time of research and writing of this thesis.

I express my profound sense of gratitude with pleasure and pride to our Expert Member **Dr. Rama** for her inspiring and excellent motivation.

I express my sincere thanks to **Mr. Winston F. Jesudas, Former HOD-IT** for his valuable encouragement during my whole work.

I express my sincere thanks to **Dr. Cyril Raj**, **HOD – CSE** and **Dr. T. Bhuvaneswari**, **HOD - MCA** for their valuable directions for my work.

I express my sincere thanks to my beloved Parents, Wife, Children and all of my family members, relatives, and friends for their valuable encouragement during my whole work.

My sincere thanks to all the staff of Dr. M.G.R. Educational & Research Institute, University and others who have at some time or other helped me directly or indirectly during this work.

Above all, the profound blessings of The Almighty made me complete this work successfully.

Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

List of Figures

List of Tables

LIST OF ABBREVIATIONS

RE	Rule extraction
CE	Co-evolution and Cultural evolution
GA	Genetic algorithms
GP	Genetic programming
ES	Evolutionary strategy
EP	Evolutionary programming
EA	Evolutionary algorithms
PSOs	Particle swarm optimizers
MAs	Memetic algorithms
EC	Evolutionary computation
RI	Rule induction
ID3	Induction Decision Trees
IG	Information gain
GR	Gain ratio
SSOCF	Subset Size-Oriented Common Feature Crossover Operator
SF	Significant features
RI	Random Immigrant
BMI	Body Mass Index
OE	Optimization in engineering
OR	Operation Research

Chapter 1

Introduction

1.1 Problem statement and overview

Many real world problems have a requirement for solutions with high accuracy and the same be provided in the shortest possible time. Examples of such real world problems include air traffic control systems, fraud detection, medical analysis systems, production and process control systems, voice identification, speech recognition, image analysis, and many more applications. Solutions which are accurate and have been optimized quickly are therefore desirable.

Currently, two competitive paradigms have been devised that solve optimization problems, namely local optimization and global optimization. The main problem with local optimization algorithms is that the algorithm gets caught in a local optimum in the search space of the associated problem. Additionally, local optimization algorithms inherently have high computational complexity and a large dependency on problem specific information. Global optimization algorithms, on the other hand, provide approximate solutions quickly.

Examples of local search algorithms include hill climbing [20,37,61],decision trees [70,54], neural networks trained using back propagation [43], and the conjugate gradient method [74]. Examples of global search algorithms are landscape approximation [49], tabu search methods [55,1], the leapfrog algorithm [4], simulated annealing , rough sets , neural networks not trained using the back propagation method, Bayesian networks [25] and evolutionary algorithms [6,25].

One specific type of optimization problem is the process of knowledge discovery, or rule extraction (RE), from databases. The Objective of the process is to extract a minimal set of simple rules that classifies (covers) with the best possible accuracy examples in the given database (dataset). Several global search algorithms in the evolutionary algorithms (EA) paradigm, which effectively classify large datasets without

performing an exhaustic search [78,4]. Examples of EA implementations used for optimization and classification problems are genetic algorithms (GA) [36,37,44], geometric programming (GP) [70], evolutionary strategy (ES) [19], evolutionary programming (EP) [20], Co-evolution and Cultural evolution (CE) [6]. This thesis focuses on Genetic Algorithm for rule discovery & feature selections in Data mining, Adaptive GA for learning classification rules and significant GA factors optimization using statistical tools.

1.2 Motivation and Objective

Genetic Algorithms (GAs) are different from more normal optimization and search procedures in the following four ways.

- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GA use payoff(Objective function) information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.

Also GA is to be one of the commonly used approaches on data mining. Hence, Gas have been chosen to use in this humble work.

The main objective of this thesis is to study the effectiveness and efficiency of combining a Genetic Algorithm in Feature selection, Rule discovery and the use of Statistical tools in optimization, also the target application area is knowledge discovery. In addition to the main objective, the following sub-objectives can be identified.

- To improve the efficiency of the fitness function of GA for finding genetic features and environmental factors which were involved in multifactorial diseases.
- To an approach to GA efficiency improvement is based on using statistical methods.

- To provide the representation of the chromosomes, reasonable operators as well as the fitness function measure how well the discovered rules work.

1.3 Thesis Outline

Chapter 2 introduces the concepts of optimization and classification. The theory of evolutionary computation is reviewed with regard to the theory of evolution as provided by Charles Darwin [21]. The general frame work of an evolutionary algorithm is then given and evolutionary operators reviewed. The process of rule extraction in the context of knowledge discovery is then discussed.

Chapter 3 provides an overview of Genetic Algorithm in the context of representation of methods and selection. Also the strengths and limitations of Gas were discussed in detail.

Chapter 4 explains how to apply a genetic algorithm approach for classification problem and based on three databases compared this approach with several other traditional data mining techniques including decision trees, neural networks and naïve bays learning.

Chapter 5 describes how the genetic features and environmental factors discovered which were involved in multifactorial diseases using a specific GA.

Chapter 6 presents the representation of chromosomes, reasonable operators as well as the fitness function that measure how well the discovered rules work.

Chapter 7 Illustrate an approach to GA efficiency improvement that has been developed. This approach is based on using statistical methods to identify the GA factors significantly affecting the efficiency of GA and to find optimal values for such factors.

Chapter 8 summarizes results and conclusions from the all experimental results given in chapter 4,5 and 6 additionally.

Chapter 2

Background & Literature Study

This chapter starts by reviewing definitions related to optimization and classification. An overview of the evolutionary computation paradigm is then presented in order to introduce evolutionary algorithms. The classical evolutionary algorithm is then reviewed with the evolutionary operators and strategies discussed. Finally the process of knowledge discovery, data representation and rule extraction is defined and discussed.

2.1 Optimisation and Classification

The term optimisation refers to both the maximisation and minimisation of tasks. A task is optimally maximised (or minimised) if the determined values of a set of parameters of the task under certain constraints satisfy some measure of optimality. For example, the optimisation of a combinatorial problem can be described as the process of continual improvement (refinement) of the set of possible solution encodings of a problem until the best (most accurate) solution, possibly the action solution, is found. The set of all possible solution encodings of a problem until the best (most accurate) solution, possibly the actual solution, is found. The set of all possible solution encodings defines a search space. The complexity of the problem is defined by the size of the search space. The process of optimisation is a repetitive process that can continue indefinitely, unless some termination condition (measurement of optimality) is specified. The termination condition must stop the algorithm when one of two conditions occurs. Either the solution produced by the algorithm is a close approximation of the actual solution, or a predefined maximum number of allowable generations are reached. Another variation of the termination condition is to terminate the algorithm if no improvement is detected in the current solution for a specified number of generations. Therefore, optimisation algorithms

can guarantee termination in an acceptable amount of computation time if a proper termination condition is specified. Hence, optimisation algorithms are useful for finding the best possible solution to a problem that cannot otherwise be solved by conventional methods and algorithms [79,61,44,11,45,18,76,67,52].

Classification algorithms that perform classification use data from a given database and its classification classes to build a set of pattern descriptions. If the pattern descriptions are structured, they can be used to classify the unknown data records from the database. Discriminating and characterization rules, which are discussed in detail in section 2.4.3 regarding rule extraction, are examples of structured pattern descriptions produced by the process of classification. The goal of the classification algorithm is to produce pattern descriptions, and are thus sometimes referred to as classifiers. Classifiers, however, imply more than just the algorithm. They are really the product of the classification process that is defined as one in which classification algorithms are used in a manner known as supervised learning. With supervised learning, the data from the database is divided into two sets, namely, training and test datasets. The training data are used to build the pattern descriptions with the help of classification classes, while the test data are used to validate pattern descriptions. If no classification classes are provided in the database records, the purpose is rather to identify the existence of clusters with common properties in the data. These clusters are then recognized as classification classes and are used in a process known as unsupervised learning. The classification process is discussed further in section 2.4.3 regarding rule extraction.

For complex real world problems, it is sometimes more important to find a quick, approximate solution than to spend time producing an accurate one. From the speed-versus-accuracy trade-off, one may be satisfied by sacrificing accuracy for a faster approximate solution. Complex real world problems are usually also described by non-linear models. This means that non-linear optimisation or classification problems need to be solved. Either conventional methods cannot find a solution, or they do not solve the non-linear problems within the required time-frame.

2.1.1 Search Paradigms

Two distinct types of search paradigms are used to solve non-linear problems, i.e local and global search. Local search algorithms search for optimal solutions near existing solutions, while global search algorithms explore new areas of the search space. Both paradigms have their own advantages and disadvantages. These are discussed in the following paragraphs.

Global search algorithms such as evolutionary algorithms (EA), are probabilistic search techniques inspired by principles of natural evolution. A few other examples of global search techniques include tabu search, asynchronous parallel genetic optimisation strategy, particle swarm optimizers (PSOs) [76,10], and landscape approximation [49].

Global search algorithms implement operators to cover the largest possible portion of search space S_p for problem P in order to locate the global optimum. The latter, in the case of a minimization problem, is defined as the smallest value that can be obtained from the objective function, $m_p(y,x)$, and is viewed as the minimum 'price' or 'cost' of solution y given instance x for the problem P. Therefore, in mathematical terms, y^* is the global minimum of m_p if $m_p(y^*,x) \leq m_p(y,x) \nabla x \in [x_{min,} x_{max}]$.

Local search algorithms, on the other hand, usually employ a directed deterministic search which covers only a portion of search space S_p. For example, the conjugate gradient method uses information about the search space to guide the direction of the search. Local search algorithms find a local minimum, y^*, where $m_p(y^*,x)=0$ and $\exists$ $x \in [x_{min},x_{max}]$ such that $m_p(y,x) \leq m_p(y^*,x)$.

Generally, the use of a local search algorithm is discouraged when the local search space has many optima [20], or where the search space is very rough and convoluted. Figure 2.1 below illustrates such a rough and convoluted search space. A global search algorithm, in contrast, is more efficient in locating a solution for convoluted spaces. A global search algorithm is faster than a local search algorithm when searching through such rough and convoluted search spaces. However, global search is less effective in refining the solution.[79,31,7,49,25]

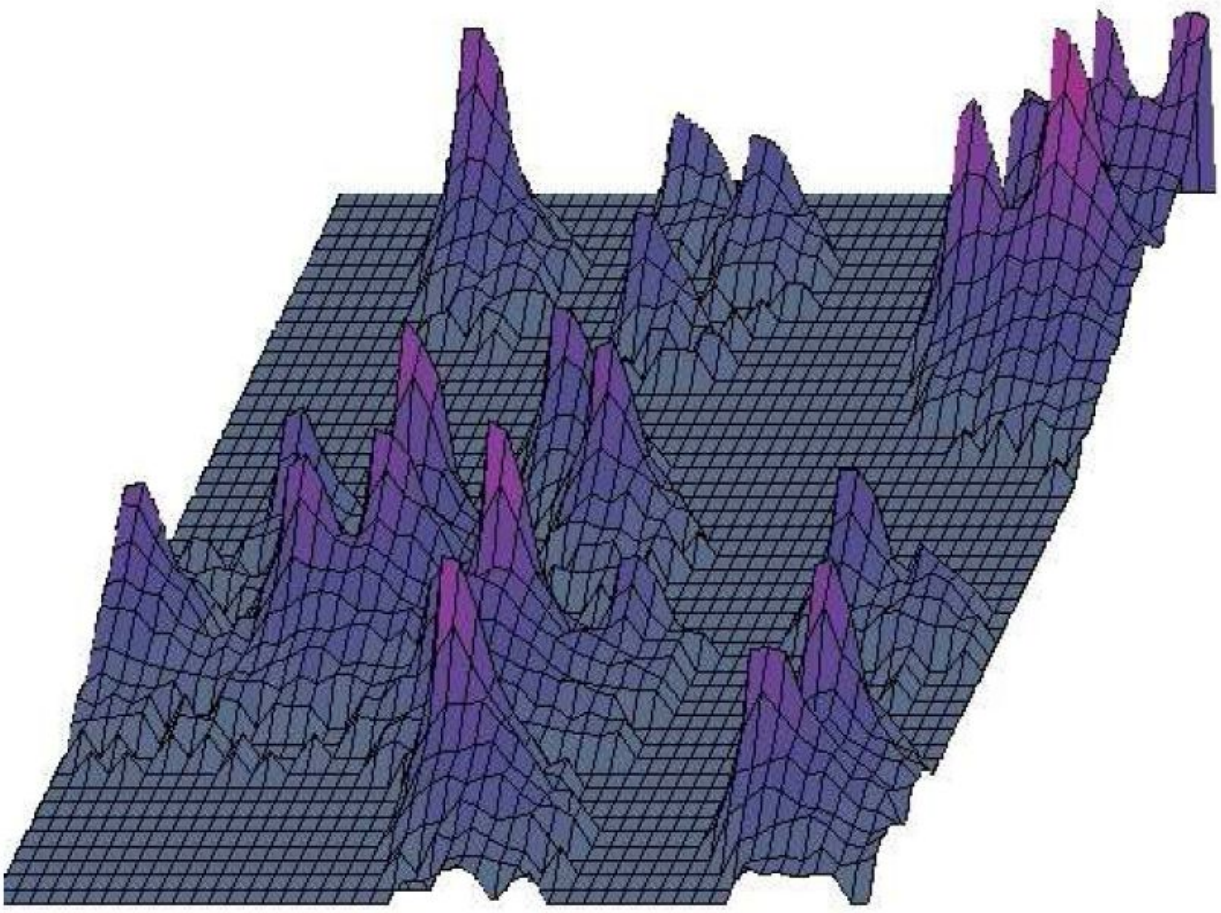

Figure 2.1 : An example of a rough and convoluted search space

In recent years, a new paradigm of search algorithms for optimisation has been developed, namely memetic searches. Memetic algorithms (MAs) are generally acknowledged as being one of the more successful approaches for combinatorial optimisation. MAs in particular are very successful in obtaining more accurate solutions, rather than the approximate solutions produced by other EAs for NP-hard optimisation problems [35,61,65,54]. NP-hard problems are defined as those problems for which a

global solution cannot be found in polynomial time, where a measure of accuracy can be calculated for possible solutions. The success of MAs is partly due to the fact that a MA is able to obtain and identify meta-heuristics concerning a specific problem. MAs are considered to be a synergy of different search approaches being incorporated. As a more general description, one can think of MA as a special combination of a global search algorithm with some other localised optimising strategy. In essence, MAs can be classified as being two competitive optimising strategies (local and global searches) that are in co-operative fashion as a single algorithm.

To distinguish the MA from other optimisation methods and algorithms, and to describe MAs better, the term meme was introduced by R.Dawkins in the last chapter of his book [23]. He defines memes as ideas, information, clothing fashions, ways of building bridges and arches, catch-phrases, music and imagination. Just as genes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation. MAs use memes that are defined as a unit of information adapted during an evolutionary process.

MAs combine global search algorithms, specifically evolutionary algorithms that make use of evolutionary operators, with local search algorithms. The EAs can quickly determine regions of interest in the search space, while local search algorithms refine the solution in a localised area of the search space. A MA is therefore more efficient and effective in finding and refining a solution than a single global search algorithm. Alternate names for MAs such as hybrid genetic algorithms, hybrid genetic programs [31,77], genetic local search algorithms and knowledge-augmented genetic algorithms [11,19], all have their origin in the notion that a MA can be seen as an EA. This thesis, however, views MAs as population-based evolutionary search methods for combinatorial optimisation problems. Note that MAs use as much knowledge as possible combinatorial optimisation problem to produce a solution. In some respects MAs are similar to EAs, that simulate the process of biological evolution. However, unlike EAs that use genes, MAs use memes that are typically adapted, and transmitted to the next generation. In [61], Moscato and Norman state that memetic evolution can be mimicked by combining

EAs with local refinement strategies such as a local neighbourhood search simulated annealing or any of the other local search heuristics explained. It also provides an in-depth theoretical discussion regarding local search and MAs.

EAs play an important role in this thesis which holds that an EA is transformed into a MA. Both EAs and MAs fall into a subset of the evolutionary computation (EC) paradigm. Therefore, it is necessary to define and review that paradigm which is done in the following Section.

2.2 Evolutionary Computation

The theory of evolution was presented by Charles Darwin in 1858 at a meeting of the Linnean Society of London and subsequently published [21]. At the same meeting, Alfred Russel Wallace also presented his independently developed theory of evolution and therefore shares the credit with Darwin. In essence, the theory of evolution states that the goal of all species is their survival, which is based on the underlying principal which is that of optimisation.

In nature evolution is driven by four key processes, namely, reproduction, mutation, competition and natural selection [51]. Individuals in a population of a species carry the genetic traits of the species. If the species is to survive and be more successful, then the genetic traits have constantly to change and adapt in order that the species survives the constantly changing environment that it occupies. Note that the environment of the species changes due to natural causes that occur on earth.

Reproduction facilitates change the adaptation of genetic traits of the successful organisms in a species. Reproduction is the process whereby one or more successful parent organisms can reproduce either by cloning or combining their genetic materials. Note that in nature the reproduction model is very successful whereby two parents pass on their combined genetic traits to their offspring.

Mutation, alternatively, occurs when reproduction produces new organisms as offspring, and there is an error in the transfer of genetic material from the parent

organisms to their young ones. Mutation also occurs when environmental conditions force mutation of the genetic material within an individual organism. Therefore the process of mutation can be either harmful or beneficial to the resultant organisms. An offspring's genetic traits (characteristics) are, therefore, in part inherited from parents through the process of recombination, and in part as the result of genes that are mutated in the process of reproduction.

Competition and natural selection go hand in hand because, in nature, organisms are compelled to compete for food, living space (shelter) and the need to reproduce. Competition for natural resources and timely reproduction usually leads to the more successful organisms' survival and procreation, while less successful organisms may well face extinction. Therefore, the phrase 'natural selection' is used to describe the survival of the 'fittest'. Note that the genetic traits of an unsuccessful organism are not necessarily removed immediately, but rather they do not 'survive' into the next generation. If the organism is removed immediately, the operation is then referred to as 'culling'. In nature successful organisms may also survive for multiple generations, a process referred to as 'elitism'.

Evolutionary computation is inspired by Darwin's theory of evolution, and was introduced in the 1960s by Rechenberg, John Holland, in 1975, introduced genetic algorithms [36]. In 1992, genetic programming was derived from genetic algorithms by John Koza, in order to evolve programs to perform certain tasks. Today, the different evolutionary algorithms that were developed in the EC paradigm include genetic algorithms (GA), genetic programming (GP) [13,69,46,24,50], evolutionary programming (EP) [6,7,37], evolutionary strategies (ES) [5,45], co-evolution and cultural evolution (CE) [25].

As a principle, EC algorithms emulate nature by implementing the four key evolutionary processes given above as operators which affect a population of individuals. The mathematically encoded solution to a problem is known as the genotype of an individual in an EC algorithm. Therefore, a population of individuals in the problem space encodes a set of possible solutions. In nature the equivalent of a mathematical

representation of the individual is the genetic traits (genes) of a successful organism. Therefore, the genotype carries genetic information encoded in an individual. The set of related properties that a population of individuals may exhibit in a specific environment is known as the phenotype of a population. In other words, the phenotype is the behaviour exhibited by the genotype of a population in a certain environment. Interactions between, and dependencies of the phenotype and genotype of a population can be quite complex. Note that in natural evolution, the mapping between phenotype and genotype is a non-linear function (not one-to-one mapping), and represents interaction between genes and the environment. Unexpected variations in phenotypes can be caused by a random change in a fragment of information in the genotype, a process known as pleiotropy. A specific phenotype trait is produced by the interaction of several pieces of information in the genotype and is known as polygeny. Therefore, a specific phenotype trait can be changed only if all relevant pieces of genotypic information that individuals possess are changed.

Evolutionary computation is used not only to solve optimisation problems, but also has been found to be useful in a variety of applications. These relate to problems that overlap somewhat, but all were previously thought of as being computationally intractable. Application categories in which EC has successfully been used [26,46,55,18,76,4,38] include:

- simulation
- classification and identification
- design and planning
- process control systems and
- image feature extraction.

The participation of EC algorithms in the referred categories vary depending on the application and the particular EC algorithm used. EC have proved to be very successful in providing solutions for the problems of the above mentioned categories, in spite of such problems being both complex and time consuming to solve. However, one of the drawbacks of many EC methods, specifically the global search variety, is that even

though they may be relatively efficient in finding a solution, it seldom has high accuracy. Therefore, most global search methods do not have a good 'exploiting capability'. The inaccuracy of the global solution found is an inherent drawback specifically built into most global search methods. It is required that the implemented global search algorithm must not return solutions at specific local optimum points before the full search space has been explored.

The following Section reviews broadly and discusses the strategies and operations shared by all EAs. Another important aspect regarding an EA is the representation of the genotype of individuals in its population. The representation of the genotype is reviewed in detail in the final Section below regarding rule extraction, where it is appropriately discussed since its influence is paramount on the rule extraction process.

2.3 Evolutionary Algorithms

This Section provides a review of EAs, their operations, strategies, and general architecture. First, the benefits of using EAs are listed. Then the general architecture for a classical EA is discussed, followed by the algorithm framework. The latter is used to review the initialisation of the population, the EA operators and the issues relating to EAs. Several strategies that were devised for some of the operators are also discussed. Finally, the issues that all EAs face are listed and discussed.

2.3.1 Benefits and General Architecture

The benefits of using EAs include the following:

- The evolution concept is easy both to understand and to represent in an algorithm.
- The implementation of an EA can be kept separate from another application that utilises it, as it is modular.

- Multiple solutions can be produced for a multi-objective problem, and multiple fitness functions can be used simultaneously [19,5,18].
- The EA algorithm can be distributed easily over several processors, because some of the operations in an EA are inherently parallel.
- Some of the previous or alternate solutions can easily be exploited by using EA parameter settings.
- As knowledge about a problem domain is gained, many ways become apparent that can be used to manipulate the speed and accuracy of an EA-based application.
- 'Noisy' data are handled while reasonable solutions can still be delivered by EAs.
- An EA always produces an answer.
- The answers become more accurate with time.

The different benefits of EAs will become even more apparent over the following paragraphs as the architecture, operators and different strategies are discussed. All EAs have the same general architecture:

1. Initialise the population
2. While not converged
    a) Evaluate the fitness of each individual
    b) Perform evolutionary operations such as reproduction and mutation
    c) Select the new population
3. Return the best individual as the solution found.

The different EA paradigms that were mentioned in the previous Section, e.g. GA, GP, EP, ES and CE can be distinguished by the population size relationship between parent and offspring populations, representation of the individuals, as well as the operator strategies that are used. Note that because GP was derived from GA, GP is treated as a specialised GA with a different representation for the individuals in its population. This thesis concentrates on a specific implementation of a GP, and for that reason, the

evolutionary procedure outlined above is discussed wit the GP paradigm in mind. In order to define properly and elaborate on the evolutionary process, some mathematical notation should be explained.

The search space S_p was defined in Section 2.1.1 as being the set of all possible solutions to problem P. Assume first that a population of η individuals at time t is given by $P(t) = (x_1(t), x_2(t)\ldots\ldots x_\eta(t))$. Every $x_\eta \in S_p$ represents a possible solution for problem P in the search space. Every individual solution x_η has a relative measure of accuracy in solving P. Let the fitness function, $f(x_\eta)$ which is also known as the evaluation function, decode the genotype representation of x_η and assigns it a fitness measure. Therefore the fitness of the whole population in S_p at time t can be expressed as $F(t) = (f(x_1(t)),f(x_2(t)),\ldots..,f(x_\eta(t)))$. Generally, solutions are normalized and compared with one another in how well each possible solution solves the particular problem at hand. Normalisation of the solutions will be explained in detail when the population selection process is discussed in Section 2.3.7.

The objective function is defined next in order to compare the solutions that are produced. The objective function is described as both the implicit and user-specified constraints under which solutions are optimized. The fitness function, which is a function applied to the chromosomes of an individual, is used to determine the fitness of an individual, and depends on the objective function. The fitness function can also depend on the specific representation of the individual used by the specific algorithm [30]. Note that it is only the fitness function which links the EA and the problem it is solving. No other process in the EA links it with the problem being solved. The reproductive operators use the fitness function to compare individuals with one another and are therefore dependent on it. For example, the selection operator used the fitness function, since the fitness function decodes the genotype representation of x_η (an individual solution) and assigns to it a fitness measure. Hence, the choice of fitness function along with the genotype representation is very important when implementing an EA.

The pseudo-algorithm (framework) representing the classical EA in more detail is provided by, and adapted from several authors [26,57,65,46,30,3,4,56], as given in Figure 2.2. The population at time t, P(t), and the fitness of the population at time t, F(t), have already been defined above. Parameter η denotes the size of the initial population, while Υ denotes the size of the offspring after applying the recombination and mutation operations. Usually, $\eta = \Upsilon$. $P'(t)$ and $P''(t)$ denote the resultant populations after the respective recombination and mutation operations had been performed. The recombination, mutation and selection operators are all applied with a certain probability, respectively denoted by φ_r,φ_m and φ_s. The selection probability parameter φ_s also controls, by means of some metric which is discussed in Section 2.3.7, the selection operator when the population $P''(t)$ is reduced to the initial η population size.

The steps of the EA framework as given in Figure 2.2 are reviewed and discussed in the following Sections. An overview of the initialisation of the population is provided first, and followed by a review of the genotype representation. The evolutionary operators used by EAs are then discussed, followed by an overview of common issues found in EAs. Note that the termination condition was previously discussed as part of Section 2.1 and is therefore not repeated here.

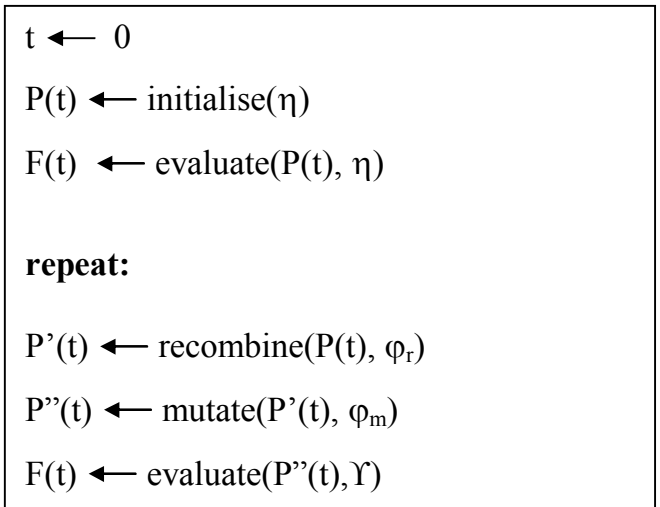

Figure 2.2: Pseudo-algorithm (framework) describing the general architecture of classical EAs.

2.3.2 Initial Population

The initial selection of the starting population must ensure diversity of the population over the search space in order to enhance the search capability of the algorithm used. A uniform random selection is the preferred method whereby the initial population is selected, since the probability is better that the spread in terms of fitness values of individuals are more uniformly distributed. A diverse selection will ensure that the fitness of the initial generation is significantly lower than the fitness of the optimal solution. The task of the evolutionary process is therefore, to improve the fitness of the initially stochastically selection of individuals.

2.3.3 Representation of the Genotype

The representation of the genotype is important because the strategies and evolutionary operators of an EA depend greatly on representation of the individuals. Traditionally, EAs such as GAs implement the genotype representation as a binary

encoding, in other words an encoded bit string [26,77,59,19]. The main problem with the binary encoding scheme is that there is a loss in accuracy when a mapping occurs between a binary string of limited length, and real-valued data that the binary string represents. This loss of accuracy occurs since binary encoding functions are complex, and if finer granularity is required in the mapping, then a more complex encoding scheme (longer binary string length) is required. If a binary string of variable length is used, then the evolutionary operators and strategies engaged need to be altered to accommodate the encoding.

The binary encoding scheme is not the only representation scheme used. Some GAs use the real values in data directly and indirectly in their genotype representations [27,41]. Another encoding scheme also employed is the permutation encoding scheme as used in [22], for example, the travelling salesman problem. Tree-based representations are used in the GP paradigm [68,13,69,46,24,50,53].

2.3.4 Evolutionary Operators

The evolutionary operators, namely recombination (also known as crossover), mutation and selection which include elitism, implement a pseudo-random walk through the search space. It is a pseudo-random walk because the recombination, selection and mutation operators are applied with a certain probability, which makes the operators non-deterministic. Note that the pseudo-random walk is directed, because recombination, mutation and selection try to maximize the quality of the possible solutions. The evolutionary operators that were mentioned are provided and discussed in the following Sections.

2.3.5 Recombination

The goal of the recombination operation is to enable the EA to look for solutions near existing solutions. Recombination achieves this goal by combining (pairing) genetic material of two or more randomly chosen parent individuals to produce offspring

individuals. The recombination of two or more individuals involves the exchange of certain randomly chosen parts of parents to create a new individual, which is added to the original population. The parts that are randomly chosen are the points in the parent individuals where the exchange will occur, and are referred to as the swap points. The recombination probability parameter, φ_r, is often applied with a high probability to ensure that crossover occurs frequently. The latter is so in order to induce convergence in the EA. Several types of crossover strategies have been devised for EAs:

- One-point crossover,
- Two-point crossover,
- Uniform crossover, and
- Arithmetic crossover.

The choice of crossover strategy depends on the specific EA algorithm that is implemented, as well as the representation of individuals of the EA. As an example, to illustrate the different crossover strategies for EAs, assume that a GA with a bit string representation is used.

With one-point crossover, a single swap point in the bit string is selected, and all the bits after the swap point are swapped out with the respective bits of the associated parent. Figure 2.3 illustrates one-point crossover.

10110 **010**	10110100
01001 **1001**	01001010
(a)	(b)

Figure 2.3: One-point crossover with parents in (a) and offspring in (b).

With two-point crossover two swap points are selected randomly, and the bits between the swap points are exchanged. Two-point crossover will, on average, select a smaller segment to swap than one-point crossover. Therefore, two-point crossover is more conservative than one-point crossover when changing the individuals.

Uniform crossover works completely differently in that the bits that are swapped are selected at random by sampling from a uniform distribution. The selection of bits occurs with a certain probability, which can also be specified to the algorithm as an additional input parameter. Once a bit which is to be swapped has been selected, the corresponding bits in the parents are swapped out.

Arithmetic crossover is specifically used in EAs where the representation of the individual is real-valued. Assume two parent individuals, $x_k(t)$ and $x_1(t)$, exist at time t. Let d be a uniform random variate in the range (0,1), therefore $d \sim U(0,1)$. Parents $x_k(t+1)$ and $x_1(t+1)$ are mutated at time t + 1 with the following respective functions:

$$x_k(t+1)=(d)x_k(t)+(1.0-d)x_1(t)$$

and

$$x_1(t+1)=(d)x_1(t)+(1.0-d)x_k(t)$$

Note that more than two parents can be chosen for recombination as in, for example, differential evolutions where more than two parent individuals are used when performing recombination.

2.3.6 Mutation

Mutation is used to randomly change the genetic material of an individual. The purpose of mutation is to introduce new genetic material into the population, thereby increasing the diversity of the population. Therefore, the mutation operation enables the EA to look at completely new areas of the search space. Hence, a larger part of the search space is searched. The mutation probability parameter, φ_m, is usually set high, but decreases as the fitness of the individuals in the EA improves. In fact it is recommended that $\varphi_m \alpha F(t)$, because initially the EA needs to explore more unexplored regions of the search space for possible solutions. As the EA evolves solutions over the generations, and the population fitness improves, the probability to mutate must be reduced in order to reduce the chance that highly fit individuals are mutated. Several mutation strategies

specific to GP have been devised. Other known mutation strategies include bit flip mutation, and promote and demote mutation.

2.3.7 Selection

The function of the selection operator is to reduce the size of population P(t+1) back to the original number of individuals, η, after the recombination and mutation operators have been applied. Remember that the recombination and mutation operators add Υ offspring to the original population. However, to simulate Darwin's evolution theory effectively, every new generation must be equal to the initial generation in population size. Therefore, Υ individuals must be selected to be removed from the $\eta + \Upsilon$ set of individuals. Note however that the selection operation is directed, meaning that preference is given to certain individuals with higher fitness values, rather than individuals with lower fitness values. In fact, a selection strategy, namely elitism, has been devised that ensures that a certain percentage of the population, say the top 20% in terms of their fitness value, survives to the next generation by copying directly that subpopulation to P(t+1). Note that selection strategies are used to select individuals from the population after recombination, mutation, elitism and selection of the new population operations have been performed. The value of the fitness function $f(x_1)$ for individual x_1 is compared with the fitness values of other individuals in the population when deciding whether an individual is selected or not. Therefore, the fitness function is the selection criterion used during the selection process. Several selection strategies have been devised in order to allow for a higher probability for selecting individuals with higher (or lower) fitness values [10,15,26,46,48,58,60,68,76], namely:

- Roulette Wheel Selection
- Rank Selection
- Steady State Selection and
- Tournament Selection

The selection operators listed above are discussed in the following paragraphs.

Roulette Wheel Selection

With roulette wheel selection, each individual is assigned a slice of the wheel in proportion to the fitness value of the individual. Therefore, the fitter an individual is, the larger the slice of the wheel. The wheel is simulated by normalising the fitness values of the population of individuals. Normalisation of fitness values of the individuals implies that all the fitness values must add up to 1. The roulette wheel strategy is best explained by the following example. Assume there is a set of four individuals (x_1,x_2,x_3,x_4) in the population of possible solutions. Assume that the normalized fitness of the set is (0.1,0.2,0.3,0.4), which add up to 1. For individual x_1 a range from 0 (exclusive) to 0.1 (inclusive) is assigned, i.e. $x_1 \in (0.0, 0.1)$. For individual x_2 a range from 0.1 (exclusive) to 0.3 (inclusive) is assigned, i.e. $x_2 \in (0.1, 0.3)$. The high-end normalized range value of x_2 is calculated by adding the normalized values of x_1 and x_2. The range values of x_3 and x_4 are determined and assigned in similar fashion, and is therefore $x_3 \in (0.3, 0.6)$, $x_4 \in (0.6, 1.0)$. Clearly x_4 has the largest proportion of the range (0.0, 1.0), which is the equivalent of the roulette wheel, while x_1 has the smallest proportion. Hence, x_4 has a better probability of being selected than x_1. A random number is then generated from a uniform distribution U(0.0, 1.0). Assume the random number is 0.5, and then the selected individual is x_3, because 0.5 falls within the range represented by x_3. As a result of performing roulette selection, the individual with the best fitness value has a better chance of being selected.

Rank Selection

Rank selection is performed by first ranking the population of individuals according to the fitness. Using the example in the roulette wheel strategy, x_1, x_2 ,x_3, and x_4 would be ranked (4,3,2,1). A random selection from the set {1, 2, 3, 4} is made to choose the individual. Note that rank selection is based on the ranking of the individuals, and not their fitness value. Hence, the rank selection method focuses rather on the rank while ignoring absolute fitness differences between the individuals. As a result of performing rank selection, highly fit individuals cannot dominate, when a fitness-based selection is performed.

Steady State Selection

With steady state selection a few individuals with high fitness values in every generation are selected for creating new offspring individuals. The new offspring individuals replace the parent individuals only if the offspring is more fit than the parents. The whole population is then selected for the new generation.

Tournament Selection

Several variations of the tournament selection strategy exist [68,76,10]. Only two popular variations are provided here. The first is to divide the population into groups by adding at random an individual to a group. After the population is divided into several groups, select the best individuals in terms of fitness values from each group. Note that each individual cannot appear in more than one group. Thereby, individuals within the groups compete with one another.

The second variation uses a given value n to compare n individuals with one another. Randomly select n individuals as the subpopulation to be used. Individual x_n of the sub-population of n individuals is compared with the rest of those in the sub-population. The individual x_n scores a point if it has a higher fitness value than another individual in the sub-population. Hence, the individuals in the sub-population can be

ranked according to the points they score when compared to other members in the sub-population. If n is set to a low value, the ranking of individual x_n is not necessarily unique, since it then depends on other individuals in the sub-population to which it was compared. Hence, the selection pressure is low, and therefore approaches random choice. If n = ϒ, then the sub-population will be equal in size to the population, and an unique rank is possible for each individual, as is the case with the rank selection strategy. The result of choosing a small value for n is that a truly random selection of individuals is achieved, while a larger value for n will force a selection according to rank.

2.3.8 Issues in EA

Some common issues that require careful consideration when selecting a specific EA implementation exist in all EAs. Often the hardest and most important choice in producing a good solution is the fitness function that will be used [46,4,56,12]. However, the representation of the genotype of the individuals in the population usually influences the choice of fitness function, as well as strategies implemented by the evolutionary operators. Careful consideration must be given to the input parameters, which include the population size and operator probability parameters which were discussed. It is important to consider the population size because if the number of individuals is too few, then a proper exploration of the search space is unlikely. A large population size can conceivably cause problems if the hardware that the EA is executed on does not have enough resources. Furthermore, a large population would induce more evolutionary operations which increase computational complexity unnecessarily, while a smaller population can find a solution of equal accuracy with less associated computational difficulty. Unfortunately, the systems that EAs run on are all limited in resources in some way or the other. Those limitations often include the processing power and memory capabilities of the system.

The importance of the selection of a correct representation for the encoding scheme was discussed in detail in Sections 2.3.1 and 2.3.3 above. Another very important

consideration involves the recombination rate and mutation rate. The recombination rate controls the rate that the algorithm will converge. If the algorithm converges too fast, then the probability that the search space is not properly explored increases. If the recombination rate is selected at a low level, the algorithm will require more generations before converging to a solution of sufficient accuracy. The mutation rate controls the pace with which new regions of the search space is explored. If the mutation rate is selected too high, the algorithm may not converge at all to a solution. If the mutation rate is selected low, very few new possible solutions are introduced. Hence, the search capability of the algorithm is reduced.

The selection strategy and the policy for the deletion of individuals from the population are further issues that are partly decided by the specific EA implementation in use. The choice of termination criteria that is commonly used by most of the EAs was discussed above as part of Section 2.1.

Another issue that EAs have to consider is that the solutions produced must not over-fit the problem data. With over-fitting it is meant that the solutions do not generalise the data well. Over-fitting of the data is discussed in more detail in the following Section. The choice of the specific EA implementation is also influenced, but not determined, by the ease with which the problem data are mapped to the genotype representation of the individuals of the specific implementation. If the mapping is incorrect, the solutions produced will be sub-optimal and less useful in terms of solving the problem.

The representation of the individuals is especially important when rules are extracted from solutions produced by EAs. Hence, representation of the individuals is discussed in detail in the next Section. There also are reviewed concepts associated with the process of knowledge discovery.

2.4 Rule Extraction

In order to understand clearly the process of rule extraction, a motivation for performing it is required. Such a motivation for performing knowledge discovery is given in the following Section. Knowledge discovery is then described in detail and the subsequent process of performing rule extraction is given. Finally, an overview is then provided of current methods by which one can perform knowledge discovery.

2.4.1 Motivation for Knowledge Discovery

Knowledge discovery has become important in recent times because most corporations have acquired and are storing vast quantities of data that is subsequently analysed in order to remain competitive. Computation systems and methods for analysing databases have improved considerably in recent times. The direct result is a reduction in the cost associated with automated knowledge discovery. For example, a company can be more competitive by identifying trends and constraints within the market in which it competes by storing market associated data, and subsequently analysing it. However, the stored data has to be analysed and classified within a reasonable time frame in order to be useful in a competitive environment. A further requirement is that the trends and constraints of the market must be determined with sufficient accuracy in order to be useful. The costs involved in the production of a product or delivery of a service can be part of the constraints. The constraints associated with products and services are not static. They change over time. Consider the mobile phone market which is constantly changing with the introduction of new technologies (trends) and new mobile phone models, with the retirement of older types. The technology used in a mobile phone, and the materials used to construct it have an ever changing cost component. Mobile phone manufactures can use information regarding the costs involved when producing their products. If the cost is known, then competitive pricing of the item can be determined optimally. Therefore, rules are required that describe the trends and constraints (such as costs) associated with a product or service. The rules can be used to determine pricing of

products and services, and can give important guidance when strategic business decisions are made.

A set of rules is the preferred format of the output when solving a classification problem, because humans can understand and interpret rules. For example, the rule 'IF cost to produce mobile phone A is high AND profit margin is low AND demand for mobile phone A is low THEN stop production of mobile phone A', is clear and unambiguous. The example rule helps to determine when it is no longer profitable to produce mobile phone A, as when combination of the production cost, and the demand for mobile phone A is at a certain threshold. Hence, a rule concerning the constraints and trends associated with a concept is more useful to humans than the raw data in the database regarding it.

The constraints and the trends associated with a product or service, in other words the dataset that describes the product or service, are dynamic. Therefore, the rules describing the data must also be dynamic. Traditionally, such rules are extracted by humans who perform the analysis and classification of the data describing products or services. However, due to the sheer volume of information available and the number of variables to be taken into account, the task of analysing and classifying data has become complex and time-consuming at best. Once a set of rules has been determined that sufficiently describes the dataset, it is very difficult to allow for any subsequent addition of further relevant attributes without affecting any or all of the existing rules. Such an introduction of additional relevant attributes would increase the complexity of the problem, and could render existing classification rules obsolete. To repeat the whole analysis phase can be time-consuming, while placing a considerable strain on existing resources. The set of rules can also be more difficult to extract from the dataset if more attributes are added. Therefore, the competitiveness and productivity of any company or person may be seriously impeded.

Professionals such as mathematicians, physicists, chemists, engineers, medics, economists and many others can also gain from using knowledge that describes data (meta-data) that they work with. For example, a chemical plant can be designed optimally

if certain constraints such as running costs, environmental pollution and the required maintenance intervals are known in advance during the design phase.

Clearly, the process of automated knowledge discovery compared to a traditional type of analysis done by humans has some advantages, namely:

- shorter time to solution (classification) period
- more adjustable and dynamic to parameter and attribute changes
- improved accuracy in solutions (classifications)
- less costly and more consistent than human classifiers
- no or very little expertise required to perform analysis, and
- may be used for multiple problems.

The next Section defines and describes the process of knowledge discovery.

2.4.2 Knowledge Discovery

Knowledge discovery can be defined as the non-trivial process of identifying and extracting useful, valid, novel and understandable knowledge from data. The process of knowledge discovery may be described as the accumulation of knowledge about data, in other words, meta-data, and the subsequent analysis of that meta-data in order to extract rules that describe with sufficient accuracy the behaviour of the data. The extracted rules can then be applied in identifying concepts in the data with the same syntax and semantics within the same context. The general form of a rule (also known as a 'conditional expression') is exemplified as:

IF antecedent THEN consequence (2.1)

Two types of rules can be defined, namely characterisation rules and discriminant rules.With characterisation rules, the objective is to find rules that describe the properties of a concept. Characterisation rules have the form:

IF concept THEN characteristic (2.2)

With discriminant rules, the objective is to find rules that allow the selection (discrimination) of the objects (data records), belonging to a given concept (class), from the rest of the objects (data records or classes). Discriminant rules have the form:

IF characteristic THEN concept (2.3)

Note that the inverse implication of the characteristic rule is not a discriminant rule. An extracted rule from a dataset that performs a classification of instances belonging to a class in the set is sometimes referred to as a classifier. For example, the following unordered rule set can be used to divide and group data instances in a two-dimensional (x and y attributes) problem space. The unordered rule set is visually represented by Figure 2.4:

IF $y \leq 3$ THEN class 1;

IF $y > 3$ AND $x \leq 3$ THEN class 2;

IF $y > 3$ AND $x > 3$ AND $x \leq 4$ THEN class 3;

IF $y > 3$ AND $x > 4$ THEN class 4;

If the rules are read in a certain order, then the ordered rules can be further simplified as follows:

1. IF $y \leq 3$ THEN class 1;
2. IF $x \leq 3$ THEN class 2;
3. IF $x > 3$ AND $x \leq 4$ THEN class 3;
4. IF $x > 4$ THEN class 4;

Note that each condition in the rules above is a boundary as illustrated in Figure 2.4. The data instances are divided (split) into the illustrated classes if they fall within the class regions formed by the boundaries as illustrated.

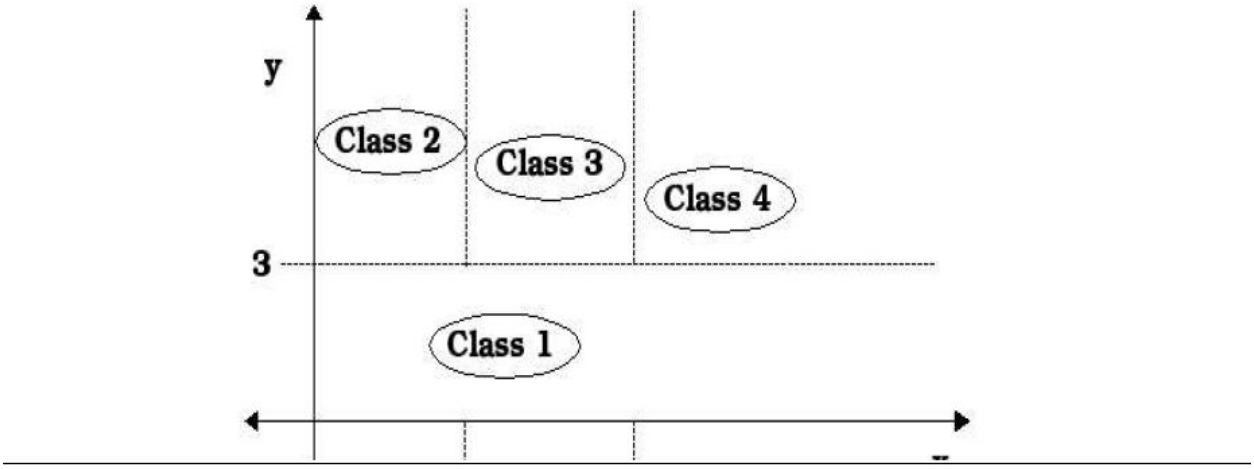

Figure 2.4: Data instances are grouped into classes by rules.

In order for extracted rules to be useful, the rules must describe with sufficient accuracy the behavior exhibited by the relevant data. The data that is used in the knowledge discovery process is typically ordered as a set of instances with equal numbers of attributes per instance. The attributes are the different properties that characterize the concept represented by an instance. The concept characteristics of an individual can be expressed as the set of k attributes a_1,a_2,……, a_k, and their respective values v_1,v_2, ……., v_k that are characteristic for a given concept C. Assume that there exist a dataset of n records with a special attribute a_c, called the class (decision) attribute. The class attribute values v_{c1},v_{c2}, ……., v_{cn} are also known as the class labels. The attribute a_c partitions the records in the dataset, i.e. divides the records into disjoint subsets defined by the values of the class attribute, called classes that classify the records. The number of disjoint subsets is equal to the number of classes present in the dataset.

For example, botanists classify the subspecies of the iris plant by comparing the different properties (attributes) such as petal length, petal width, sepal width and sepal length. Therefore, the attributes encapsulate information that is useful when a classification of the instance is required. The knowledge that is accumulated during the process of knowledge discovery is typically knowledge that botanists have gained with

experience. An experienced botanist may know that to classify subspecies of the iris plant, one needs only to compare the petal length attribute of the different instances while ignoring other attributes. Therefore, inexperienced botanists could also quickly classify the subspecies of the iris plant, if a rule regarding the iris petal length attribute existed, knowledge about data can be very useful.

A common problem that all classifiers encounter is linked to the quality associated with the data on which knowledge discovery is performed. In a perfect world, the data that is used in the knowledge discovery process is flawless. However, data in the real world has two common defects namely, 'outlier' data instances and so called 'white noise' data. An outlier data instance has a class attribute value that incorrectly classifies the concept that the rest of the attributes of the instance described. White noise data, also known as Gaussian noise, is defined as data with a small variance and zero mean.

Therefore, it is data that has a close proximity to the true data point, and is incorrect with only a small difference in a few attributes. In other words, for a white noise data instance the class attribute value is correct, but some of the instance attribute values do not describe correctly, to some degree, the concept given by the class attribute value. White noise data may also contain incomplete data attribute values that do not influence the classification of the data instances to which they belong. A good EC algorithm that is robust is not significantly influenced when producing solutions while using data that contains both outlier data instances, and white noise data. On the other hand, a human may have difficulty discerning and disregarding the outlier and the white noise data instances.

2.4.3 Rule Extraction

The process of rule extraction (RE) can be thought of as a transformation of nominal, continuous or discrete knowledge from a dataset into useful symbolic propositional logical rules that describe with sufficient accuracy the knowledge embedded in datasets.

The rules (knowledge) extracted during the knowledge discovery process need to adhere to the following criteria:

- Rules must be comprehensible.
- Rules must be short, simple, crisp and clear.
- Rules must be accurate in describing the data from which they were extracted.
- Avoid duplication of rules.
- Knowledge encapsulated in rules must be useful.

Data mining is the process whereby useful and valid rules (knowledge) are extracted from large databases by means of a classification method. The process of data mining is as follows.

1. Pre-process data in the database that is to be used with a rule extraction method. For example, the raw data in the database may not be in a format that is useful to the algorithm used. Depending on the algorithm used, missing attribute values may need to be replaced with a value that is the average value calculated for the specific attribute.
2. An EA needs as input data some knowledge about the attributes of the given dataset, such as which attributes are continuous, discrete or nominal.
3. Create training and validation (test) datasets if a supervised learning strategy is used.
4. Use an algorithm or method to find a solution that describes the data in the database with sufficient accuracy.
5. A Post-process procedure which involves extracting useful rules from the solution found.

The post-processing procedure is the means whereby useful rules are extracted and presented to a human being for interpretation and use. Since this thesis concentrates on a supervised learning method for rule extraction, the focus will be supervised learning. To build the rule set to classify a dataset, a local, or a global, or a hybrid of a local and a global algorithm is used in two phases: training and validation (testing). In Section 2.1 the process of supervised learning and the use of training and testing datasets were

introduced. To elaborate on the supervised learning process, a further example is now provided.

Assume that there exists a database with a dataset of instances of which a subset of the instances is already classified, therefore the classifications are already known. The disjoint subset consists of the unknown data instances that have to be classified. Therefore, rules (classifiers) are required to classify the unknown instances. The subset of known classified instances is used to build the classifiers, which is further divided into disjoint sets of training and test datasets. The training dataset is used to determine the class boundaries. Hence, the class attribute values are required for the training set. The test dataset has the same format as the training dataset, but does not require the class attribute values. The latter are used with the test set to determine whether the predictions of the classifier coincide with the target prediction. Therefore, the test dataset is used to determine the predictive accuracy of the set of rules. This accuracy is expressed as a percentage of the correctly classified instances in the test dataset. The process of selecting the test and training datasets can be quite complex and is outside the scope of this thesis. The predictive accuracy is used as a measure of fitness to determine how accurate the rules are that were built during the training phase. The algorithm used will terminate when the predictive accuracy reaches an acceptable level, or when a predetermined time has elapsed.

Therefore, classifiers (extracted rules) in the data mining process are the final product of a classification algorithm and a dataset. Examples of classification algorithms include [2,6,25,68,53,18,64,56,38,43,78,72,71,14,33,34,62,63] :

- Neural Networks
- Bayesian Networks
- Rough Sets
- Decision Trees and
- Evolutionary Algorithms.

Other known methods for performing classification include case-based reasoning, fuzzy set approaches, and the k-nearest neighbor classification method. The latter method

is performed whereby data instances are represented as points in Euclidean space. The distance to a neighboring instance is measured as a Euclidean distance; hence groups of neighbors are commonly classified. A short taxonomy of the different available classification algorithms used for performing rule extraction is now provided.

2.4.3.1 Neural Networks

The objective of a neural network (NN) is to simulate operations of the human brain. Three types of nodes, namely, input, hidden and output nodes, are used with interconnecting weights between them. Note that input nodes situated in the first level exclusively connect with interconnecting weights to hidden nodes in the next level. Hidden nodes connect either to more hidden nodes in the next level or to the output nodes in the final level. Note that hidden nodes cannot connect to hidden nodes in the same or previous levels, except when working with recurrent or Hopfield NNs. When a node is identified as being important, the interconnecting weight is changed (increased or decreased depending on how the identified node is increased), which means that the node will have more influence that other nodes at the same level. The NN 'learns' by adjusting weights between the nodes. Hence, the objective of a NN is to obtain a set of interconnecting weight and nodes through training that classifies almost all the data into the correct classes in the training data.

Some of the advantages of NNs include a generally high predictive accuracy. A NN has high robustness when data contains errors such as 'white noise'. The output of a NN is always one or more continuous values, except when an additional function is used to map the continuous values to discrete values. A NN that was trained beforehand is quick when performing a classification or prediction.

Some of the disadvantages of NNs are that a NN trains notoriously slowly when the NN are used for different problems. A NN is not robust when the data contains errors such as outlier instances. Only when a 'robust estimator' is used with a NN, can data with outlier instances effectively be classified. Complex NNs (those with many node levels)

produce complex rules. The production of simpler rules which are more understandable to humans requires the use of pruning algorithms in order to reduce the number of hidden nodes (or hidden node levels). The pruning algorithms on their part require problem specific knowledge with regard to each problem about the parameters of the associated input data.

2.4.3.2 Bayesian Networks

Bayesian networks (BN) implement a probabilistic prediction learning method whereby a mathematical calculus is given for the degrees of belief. The latter is defined as the description of what it means for beliefs to be consistent, and how such beliefs must change when the evidence in the data suggest. In other words, explicit probabilities are calculated for hypotheses (beliefs). The Bayes theorem states that given training data D, the posteriori probability P of a hypothesis h, can be calculated as:

$$P=\frac{(h)}{D}$$

The probability that a hypothesis is correct is increased or decreased by each training example. Therefore, a weight can be calculated for each of the multiple hypotheses that were initially calculated. Hence, a hypothesis is not assigned to a class explicitly, but rather has a probability of belonging to a class. BNs are seen as clustering methods. Training is performed incrementally by adjusting the weight associated with each hypothesis. Bayesian methods provide a standard of optimal decision making against which other methods can be measured, even if the Bayesian method used is computationally intractable. Some of the advantages include the fact that prior knowledge about the problem can be combined with several hypotheses about observed data. The BN does not over-fit the data, because the complexity of the classes is exchanged for predictive accuracy [14,16]. Unfortunately, there is a significant computational cost

involved with regards to initialisation since prior knowledge is required of the many probable hypotheses. Hence, the naïve Bayesian classifier method was created where all problem attributes are assumed to be relevant and conditionally independent of each other within each class. In other words, the naïve Bayesian classifier counts only the class distribution which significantly reduces the computational cost. The main disadvantage of BNs is that the rule set (solution) produced is usually very complex since all attributes are assumed to be relevant.

2.4.3.3 Rough Sets

Rough set theory, which provides definitions and methods for finding attributes that separate one class or classification from another, was developed by Zdzislaw Pawlak in the early 1980s. As a methodology rough sets can perform analysis and classification of knowledge expressed as data that was acquired from experience, or that is uncertain, incomplete or imprecise. The main reason for the useful analysis and classification ability of rough sets is that inconsistencies are allowed in data, and set membership classification is not absolute, hence noise in data is handled gracefully. The search space, also known as the approximation space, is classified into sets of disjoint categories. Set membership classification is performed by using objects with concepts that are known to belong to a set (a class), as well as those objects with concepts that possibly may belong to a set. Note that objects belonging to a category are not distinguishable, meaning that the objects may be members of an arbitrary set. However, it may not be possible to define the membership of the objects to the arbitrary set. Therefore, a lower as well as an upper approximation is used to describe membership of a set (a class). Those objects that are known with certainty to belong to a set are described by the lower approximation description of the set. Thus, a rough set is said to be defined through its lower and upper approximations.

Results from training performed by a rough set algorithm are usually a set of propositional rules. A disadvantage of the rough set methodology is that there is an

associated time complexity involved, possibly non-polynomial, with training and extraction of usable rules. When compared to the Bayesian approach, the rough set methodology has the advantage that no background knowledge about the data is required and no assumptions about the independence of the attributes in a dataset have to be made. Note that fuzzy sets and rough sets have complimentary notions, but must not be confused with each other since they are fundamentally different [14].

2.4.3.4 Decision Trees and Rule Induction

The ID3 and C4.5 algorithms utilize decision trees and were developed to perform classification tasks [68,53,64,56]. ID3 and the C4.5 algorithms are greedy algorithm strategies that construct decision trees in a top-down structure; in a manner that recursively divides the search space. The decision tree can be described as a flow-chart-like tree structure which is used to perform the classification of a dataset. Each node in the decision tree corresponds to a non-categorical attribute. Each arc extending from a node in the decision tree corresponds to a non-categorical attribute. Each arc extending from a node in the decision tree corresponds to a possible value of the attribute represented by the node. A leaf node of the decision tree represents the expected value of a categorical attribute. The path from the root node to a leaf node represents the subset of data records that are classified by the expected value for the categorical attribute kept in the leaf node. The selection of the non-categorical attribute associated with a node is based on whichever attribute is the most informative (best information gain), which is calculated as an entropy measure as described by Quinlan [64]. Note that for each path from the root node to a leaf node, a rule can be extracted from the decision tree [64,56].

The C4.5 algorithm is an extension of the ID3 algorithm. Attributes that can be used with the C4.5 decision tree are discrete, nominal or continuous attribute value ranges. Note that the ID3 algorithm does not deal with continuous attribute value ranges. To build a decision tree, two phases are required, namely, tree construction and tree pruning. The decision tree's construction phase will generally result in, rules that are

extracted, which over-fit the dataset, and is performed as described above. The tree pruning phase is used to improve the generalisation of the data by extracted rules. Tree pruning is used to trim decision trees in order to remove the branches from trees that reflect the instances of outlier and white noise data. Tree pruning is performed whereby a whole sub-tree is replaced by a leaf node. The function of tree pruning is to produce less complex decision trees which result in fewer and simpler rules that generalise the data well.

A disadvantage of using decision tree algorithms such as ID3 and C4.5 is that rules such as 'If $A_2 \neq A_6$' where A_2 and A_6 are both input attributes, cannot be extracted. The ID3 algorithm has an additional problem with regards to class attributes that have continuous valued ranges. Since it is impractical to grow the decision trees by adding more branches to handle all the values in the continuous valued ranges, the ID3 algorithm cannot classify datasets that include continuous value ranges. The C4.5 algorithm was developed as an extension to the ID3 algorithm to allow for continuous valued ranges and deals with the problem as follow. Assume that one of the attributes A_1 in a dataset has a continuous range. Also assume that the values for the mentioned attribute in the training set have an increasing order, V_1, V_2,…, V_m. The records in the test set can then be partitioned for each attribute value V_j, j = 1, 2,…, m into those records that have A_1 values up to and including V_j, as well as those records that have rules greater than V_j. The gain is calculated for each of the partitions, and the partition is chosen which maximizes the gain. The problem with the above method used by the C4.5 algorithm is that it is computationally expensive.

The CN2 algorithm is an example that utilises rule induction (RI) [17,73,75]. The latter does not use decision trees. Instead, RI builds sets of conditions (if-then rules) by performing a beam search. The rules produced are given as an ordered set, meaning that the rule classifying the most records in the training dataset is given first. The rule classifying the least number of records in the training dataset, are given last in the ordered dataset. A heuristic function is used to terminate the CN2 algorithm, based on the noise present in the training dataset. Unordered rules can be produced by CN2 if the evaluation

function (entropy measure) is replaced appropriately [17]. The same disadvantages mentioned for decision tree strategies are also applicable for the CN2 rule induction algorithm.

2.4.4 Evolutionary Algorithms

Several methods for extracting rules with EAs exist in the literature [8,24,28,66,39,40,42]. A method for extracting rules using an EA such as a GA, is first to create a set of rules that is subsequently optimised by the GA [66,39,40]. A method of producing the set of rules is to use the different possible combinations of the attributes in the antecedents of the rules in combination with each class attribute placed in the threshold. Each rule produced is then represented by a specific binary string. In the final phase a GA is used to optimise the rule set where the objectives are to maximise the classification accuracy, to minimise the number of selected rules, and to minimise the total rule length.

An obvious problem with the method of producing the set of rules above is that the method is practical only for a dataset containing a small number of attributes and class attributes. However, with each attribute that is added to the dataset, an exponential growth occurs in the number of rules which can be derived. The growth in the number of rules in the rule set has the result that the GA is less efficient in optimising the rule set. In addition, it becomes more complex to derive rules with the increase in the number of used attributes. Several techniques exist to reduce the number of rules in the rule set. The new set of rules is known as a fuzzy rule set since the attribute ranges are chosen by using a fuzzy range set selection method. One such technique is to divide the search space into a grid-type partition. In other words, for each input the domain interval of the input is divided into antecedent fuzzy sets. The fuzzy sets are then labeled with linguistic labels as illustrated in Figure 2.5. The produced set of fuzzy rules is pre-screened, based on fuzzy versions of two rule evaluation criteria (i.e., confidence and support) for association rules. The pre-screening (heuristic) procedure consists of dividing the rules

according to their consequent classes. The groups (classes) of rules are then sorted according to the descending order of the product of confidence and support. Finally, a shortened list is selected from the groups of classes by specifying the maximum number of rules selected from each group. A GA is then used to optimise the reduced list of rules.

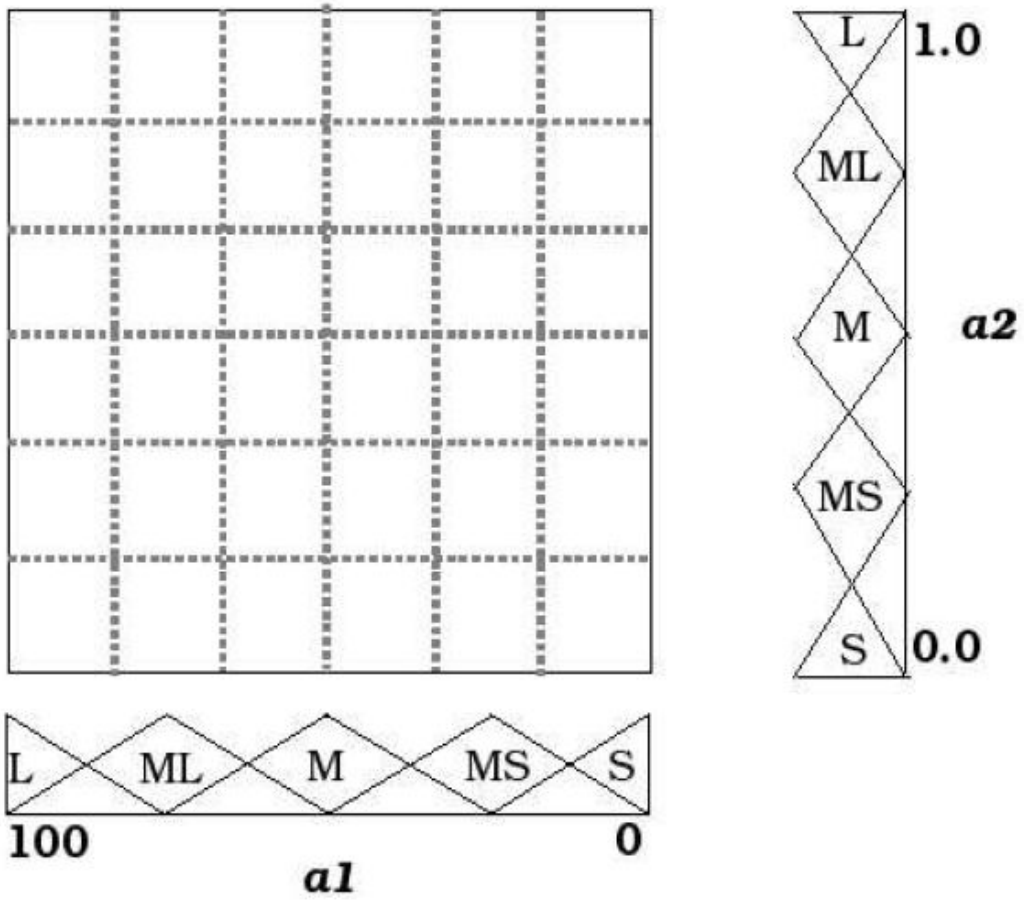

Figure 2.5: A two dimensional input (search) space partitioned by a 5 x 5 fuzzy grid.

2.5 Concluding Remarks

This chapter discussed and gave an overview of optimisation and classification methods. A thorough introduction and background regarding the different global search methods were discussed, wherein the focus was placed on Evolutionary Algorithms(EAs) in general and the Genetic Algorithm(One of EAs) in particular.

Chapter 3

Genetic Algorithms –An Overview

3.1 Introduction

Creationists occasionally charge that evolution is useless as a scientific theory because it produces no practical benefits and has no relevance to daily life. However, the evidence of biology alone shows that this claim is untrue. There are numerous natural phenomena for which evolution gives us a sound theoretical underpinning. For example, the observed development of resistance - to insecticides in crop pests, to antibiotics in bacteria, to chemotherapy in cancer cells, and to anti-retroviral drugs in viruses such as HIV - is a straightforward consequence of the laws of mutation and selection, and understanding these principles has helped us to craft strategies for dealing with these harmful organisms. The evolutionary postulate of common descent has aided the development of new medical drugs and techniques by giving researchers a good idea of which organisms they should experiment on to obtain results that are most likely to be relevant to humans. Finally, the principle of selective breeding has been used to great effect by humans to create customized organisms unlike anything found in nature for their own benefit. This field is computer science, and the benefits come from a programming strategy called genetic algorithms.

3.2 Genetic algorithm

A genetic algorithm (GA) is a programming technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a fitness function that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them, but more often they are generated at random.

The GA then evaluates each candidate according to the fitness function. In a pool of randomly generated candidates, of course, most will not work at all, and these will be deleted. However, purely by chance, a few may hold promise - they may show activity, even if only weak and imperfect activity, toward solving the problem.

These promising candidates are kept and allowed to reproduce. Multiple copies are made of them, but the copies are not perfect; random changes are introduced during the copying process. These digital offspring then go on to the next generation, forming a new pool of candidate solutions, and are subjected to a second round of fitness evaluation. Those candidate solutions which were worsened, or made no better, by the changes to their code are again deleted; but again, purely by chance, the random variations introduced into the population may have improved some individuals, making them into better, more complete or more efficient solutions to the problem at hand. Again these winning individuals are selected and copied over into the next generation with random changes, and the process is repeated. The expectation is that the average fitness of the population will increase each round, and so by repeating this process for hundreds or thousands of rounds, very good solutions to the problem can be obtained.

3.2.1 Methods of representation

Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form that a computer can process. One

common approach is to encode solutions as binary strings: sequences of 1's and 0's, where the digit at each position represents the value of some aspect of the solution. Another, similar approach is to encode solutions as arrays of integers or decimal numbers, with each position again representing some particular aspect of the solution. This approach allows for greater precision and complexity than the comparatively restricted method of using binary numbers only and often "is intuitively closer to the problem space" [80].

This technique was used, for example, in the work of Steffen Schulze-Kremer, who wrote a genetic algorithm to predict the three-dimensional structure of a protein based on the sequence of amino acids that go into it [81] Schulze-Kremer's GA used real-valued numbers to represent the so-called "torsion angles" between the peptide bonds that connect amino acids. Genetic algorithms for training neural networks often use this method of encoding also.

A third approach is to represent individuals in a GA as strings of letters, where each letter again stands for a specific aspect of the solution. One example of this technique is Hiroaki Kitano's "grammatical encoding" approach, where a GA was put to the task of evolving a simple set of rules called a context-free grammar that was in turn used to generate neural networks for a variety of problems [81].

The virtue of all three of these methods is that they make it easy to define operators that cause the random changes in the selected candidates: flip a 0 to a 1 or vice versa, add or subtract from the value of a number by a randomly chosen amount, or change one letter to another. (See the section on Methods of change for more detail about the genetic operators.) Another strategy, developed principally by John Koza of Stanford University and called genetic programming, represents programs as branching data structures called trees [82]. In this approach, random changes can be brought about by changing the operator or altering the value at a given node in the tree, or replacing one sub tree with another.

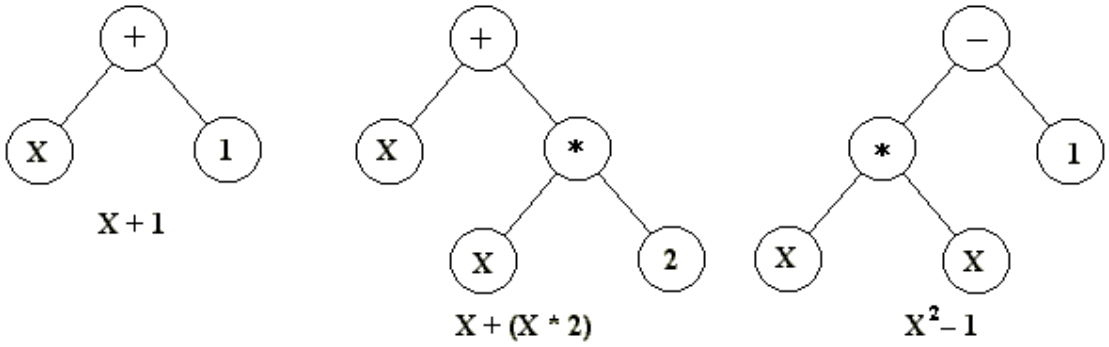

Figure 3.1: Three simple program trees of the kind normally used in genetic programming.

The mathematical expression that each one represents is given underneath.

It is important to note that evolutionary algorithms do not need to represent candidate solutions as data strings of fixed length. Some do represent them in this way, but others do not; for example, Kitano's grammatical encoding discussed above can be efficiently scaled to create large and complex neural networks, and Koza's genetic programming trees can grow arbitrarily large as necessary to solve whatever problem they are applied to.

3.2.2 Methods of selection

There are many different techniques which a genetic algorithm can use to select the individuals to be copied over into the next generation, but listed below are some of the most common methods. Some of these methods are mutually exclusive, but others can be and often are used in combination.

Elitist selection : The most fit members of each generation are guaranteed to be selected. Fitness-proportionate selection: More fit individuals are more likely, but not certain, to be selected.

Roulette-wheel selection : A form of fitness-proportionate selection in which the chance of an individual's being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.

Scaling selection : As the average fitness of the population increases, the strength of the selective pressure also increases and the fitness function becomes more discriminating. This method can be helpful in making the best selection later on when all individuals have relatively high fitness and only small differences in fitness distinguish one from another.

Tournament selection : Subgroups of individuals are chosen from the larger population, and members of each subgroup compete against each other. Only one individual from each subgroup is chosen to reproduce.

Rank selection : Each individual in the population is assigned a numerical rank based on fitness, and selection is based on these ranking rather than absolute differences in fitness. The advantage of this method is that it can prevent very fit individuals from gaining dominance early at the expense of less fit ones, which would reduce the population's genetic diversity and might hinder attempts to find an acceptable solution.

Generational selection : The offspring of the individuals selected from each generation become the entire next generation. No individuals are retained between generations.

Steady-state selection : The offspring of the individuals selected from each generation go back into the pre-existing gene pool, replacing some of the less fit members of the previous generation. Some individuals are retained between generations.

Hierarchical selection : Individuals go through multiple rounds of selection each generation. Lower-level evaluations are faster and less discriminating, while those that survive to higher levels are evaluated more rigorously. The advantage of this method is that it reduces overall computation time by using faster, less selective evaluation to weed

out the majority of individuals that show little or no promise, and only subjecting those who survive this initial test to more rigorous and more computationally expensive fitness evaluation.

3.2.3 Methods of change

Once selection has chosen fit individuals, they must be randomly altered in hopes of improving their fitness for the next generation. There are two basic strategies to accomplish this. The first and simplest is called mutation. Just as mutation in living things changes one gene to another, so mutation in a genetic algorithm causes small alterations at single points in an individual's code.

The second method is called crossover, and entails choosing two individuals to swap segments of their code, producing artificial "offspring" that are combinations of their parents. This process is intended to simulate the analogous process of recombination that occurs to chromosomes during sexual reproduction. Common forms of crossover include single-point crossover, in which a point of exchange is set at a random location in the two individuals' genomes, and one individual contributes all its code from before that point and the other contributes all its code from after that point to produce an offspring, and uniform crossover, in which the value at any given location in the offspring's genome is either the value of one parent's genome at that location or the value of the other parent's genome at that location, chosen with 50/50 probability.

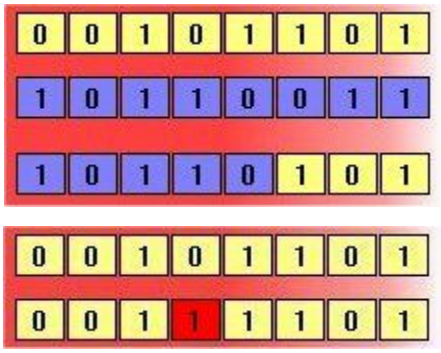

Figure 3.2: Crossover and mutation.

The above diagram illustrates the effect of each of these genetic operators on individuals in a population of 8-bit strings. The upper diagram shows two individuals undergoing single-point crossover; the point of exchange is set between the fifth and sixth positions in the genome, producing a new individual that is a hybrid of its progenitors. The second diagram shows an individual undergoing mutation at position 4, changing the 0 at that position in its genome to a 1.

3.3 The Strengths of GAs

The first and most important point is that genetic algorithms are intrinsically parallel. Most other algorithms are serial and can only explore the solution space to a problem in one direction at a time, and if the solution they discover turns out to be suboptimal, there is nothing to do but abandon all work previously completed and start over. However, since GAs have multiple offspring, they can explore the solution space in multiple directions at once. If one path turns out to be a dead end, they can easily eliminate it and continue work on more promising avenues, giving them a greater chance each run of finding the optimal solution.

However, the advantage of parallelism goes beyond this. Consider the following: All the 8-digit binary strings (strings of 0's and 1's) form a search space, which can be represented as ******** (where the * stands for "either 0 or 1"). The string 01101010 is one member of this space. However, it is also a member of the space 0*******, the space 01******, the space 0******0, the space 0*1*1*1*, the space 01*01**0, and so on. By evaluating the fitness of this one particular string, a genetic algorithm would be sampling each of these many spaces to which it belongs. Over many such evaluations, it would build up an increasingly accurate value for the average fitness of each of these spaces, each of which has many members. Therefore, a GA that explicitly evaluates a small number of individuals is implicitly evaluating a much larger group of individuals - just as a pollster who asks questions of a certain member of an ethnic, religious or social group hopes to learn something about the opinions of all members of that group, and therefore

can reliably predict national opinion while sampling only a small percentage of the population.

Due to the parallelism that allows them to implicitly evaluate many schema at once, genetic algorithms are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search exhaustively in any reasonable amount of time. Most problems that fall into this category are known as "nonlinear". In a linear problem, the fitness of each component is independent, so any improvement to any one part will result in an improvement of the system as a whole. Needless to say, few real-world problems are like this. Nonlinearity is the norm, where changing one component may have ripple effects on the entire system, and where multiple changes that individually are detrimental may lead to much greater improvements in fitness when combined. Nonlinearity results in a combinatorial explosion: the space of 1,000-digit binary strings can be exhaustively searched by evaluating only 2,000 possibilities if the problem is linear, whereas if it is nonlinear, an exhaustive search requires evaluating 2^{1000} possibilities - a number that would take over 300 digits to write out in full.

Fortunately, the implicit parallelism of a GA allows it to surmount even this enormous number of possibilities, successfully finding optimal or very good results in a short period of time after directly sampling only small regions of the vast fitness landscape [85]. For example, a genetic algorithm developed jointly by engineers from General Electric and Rensselaer Polytechnic Institute produced a high-performance jet engine turbine design that was three times better than a human-designed configuration and 50% better than a configuration designed by an expert system by successfully navigating a solution space containing more than 10^{387} possibilities. Conventional methods for designing such turbines are a central part of engineering projects that can take up to five years and cost over $2 billion; the genetic algorithm discovered this solution after two days on a typical engineering desktop workstation [84].

Another notable strength of genetic algorithms is that they perform well in problems for which the fitness landscape is complex - ones where the fitness function is discontinuous, noisy, changes over time, or has many local optima. Most practical

problems have a vast solution space, impossible to search exhaustively; the challenge then becomes how to avoid the local optima - solutions that are better than all the others that are similar to them, but that are not as good as different ones elsewhere in the solution space. Many search algorithms can become trapped by local optima: if they reach the top of a hill on the fitness landscape, they will discover that no better solutions exist nearby and conclude that they have reached the best one, even though higher peaks exist elsewhere on the map.

Evolutionary algorithms, on the other hand, have proven to be effective at escaping local optima and discovering the global optimum in even a very rugged and complex fitness landscape. (It should be noted that, in reality, there is usually no way to tell whether a given solution to a problem is the one global optimum or just a very high local optimum. However, even if a GA does not always deliver a provably perfect solution to a problem, it can almost always deliver at least a very good solution.) All four of a GA's major components - parallelism, selection, mutation, and crossover - work together to accomplish this. In the beginning, the GA generates a diverse initial population, casting a "net" over the fitness landscape. [82] compares this to an army of parachutists dropping onto the landscape of a problem's search space, with each one being given orders to find the highest peak.) Small mutations enable each individual to explore its immediate neighbourhood, while selection focuses progress, guiding the algorithm's offspring uphill to more promising parts of the solution space [84].

Crossover is the key element that distinguishes genetic algorithms from other methods such as hill-climbers and simulated annealing. Without crossover, each individual solution is on its own, exploring the search space in its immediate vicinity without reference to what other individuals may have discovered. However, with crossover in place, there is a transfer of information between successful candidates - individuals can benefit from what others have learned, and schemata can be mixed and combined, with the potential to produce an offspring that has the strengths of both its parents and the weaknesses of neither. This point is illustrated in [86], where the authors discuss a problem of synthesizing a lowpass filter using genetic programming. Another

area in which genetic algorithms excel is their ability to manipulate many parameters simultaneously [85]. Many real-world problems cannot be stated in terms of a single value to be minimized or maximized, but must be expressed in terms of multiple objectives, usually with tradeoffs involved: one can only be improved at the expense of another. GAs are very good at solving such problems: in particular, their use of parallelism enables them to produce multiple equally good solutions to the same problem, possibly with one candidate solution optimizing one parameter and another candidate optimizing a different one [83], and a human overseer can then select one of these candidates to use. If a particular solution to a multi objective problem optimizes one parameter to a degree such that that parameter cannot be further improved without causing a corresponding decrease in the quality of some other parameter, that solution is called Pareto optimal or non-dominated [87].

Finally, one of the qualities of genetic algorithms which might at first appear to be a liability turns out to be one of their strengths: namely, GAs know nothing about the problems they are deployed to solve. Instead of using previously known domain-specific information to guide each step and making changes with a specific eye towards improvement, as human designers do, they are "blind watchmakers"; they make random changes to their candidate solutions and then use the fitness function to determine whether those changes produce an improvement.

3.4 The Limitations of GAs

The first, and most important, consideration in creating a genetic algorithm is defining a representation for the problem. The language used to specify candidate solutions must be robust; i.e., it must be able to tolerate random changes such that fatal errors or nonsense do not consistently result. There are two main ways of achieving this. The first, which is used by most genetic algorithms, is to define individuals as lists of numbers - binary-valued, integer-valued, or real-valued - where each number represents some aspect of a candidate solution. If the individuals are binary strings, 0 or 1 could

stand for the absence or presence of a given feature. If they are lists of numbers, these numbers could represent many different things: the weights of the links in a neural network, the order of the cities visited in a given tour, the spatial placement of electronic components, the values fed into a controller, the torsion angles of peptide bonds in a protein, and so on. Mutation then entails changing these numbers, flipping bits or adding or subtracting random values. In this case, the actual program code does not change; the code is what manages the simulation and keeps track of the individuals, evaluating their fitness and perhaps ensuring that only values realistic and possible for the given problem result.

In another method, genetic programming, the actual program code does change. As discussed in the section Methods of representation, GP represents individuals as executable trees of code that can be mutated by changing or swapping subtrees. Both of these methods produce representations that are robust against mutation and can represent many different kinds of problems, and as discussed in the section Some specific examples, both have had considerable success.

This issue of representing candidate solutions in a robust way does not arise in nature, because the method of representation used by evolution, namely the genetic code, is inherently robust: with only a very few exceptions, such as a string of stop codons, there is no such thing as a sequence of DNA bases that cannot be translated into a protein. Therefore, virtually any change to an individual's genes will still produce an intelligible result, and so mutations in evolution have a higher chance of producing an improvement. This is in contrast to human-created languages such as English, where the number of meaningful words is small compared to the total number of ways one can combine letters of the alphabet, and therefore random changes to an English sentence are likely to produce nonsense.

The problem of how to write the fitness function must be carefully considered so that higher fitness is attainable and actually does equate to a better solution for the given problem. If the fitness function is chosen poorly or defined imprecisely, the genetic algorithm may be unable to find a solution to the problem, or may end up solving the

wrong problem. (This latter situation is sometimes described as the tendency of a GA to "cheat", although in reality all that is happening is that the GA is doing what it was told to do, not what its creators intended it to do.) An example of this can be found in [88] , in which researchers used an evolutionary algorithm in conjunction with a reprogrammable hardware array, setting up the fitness function to reward the evolving circuit for outputting an oscillating signal. At the end of the experiment, an oscillating signal was indeed being produced - but instead of the circuit itself acting as an oscillator, as the researchers had intended, they discovered that it had become a radio receiver that was picking up and relaying an oscillating signal from a nearby piece of electronic equipment!

In addition to making a good choice of fitness function, the other parameters of a GA - the size of the population, the rate of mutation and crossover, the type and strength of selection - must be also chosen with care. If the population size is too small, the genetic algorithm may not explore enough of the solution space to consistently find good solutions. If the rate of genetic change is too high or the selection scheme is chosen poorly, beneficial schema may be disrupted and the population may enter error catastrophe, changing too fast for selection to ever bring about convergence.

One type of problem that genetic algorithms have difficulty dealing with are problems with "deceptive" fitness functions [81], those where the locations of improved points give misleading information about where the global optimum is likely to be found. For example, magine a problem where the search space consisted of all eight-character binary strings, and the fitness of an individual was directly proportional to the number of 1s in it - i.e., 00000001 would be less fit than 00000011, which would be less fit than 00000111, and so on - with two exceptions: the string 11111111 turned out to have very low fitness, and the string 00000000 turned out to have very high fitness. In such a problem, a GA (as well as most other algorithms) would be no more likely to find the global optimum than random search.

One well-known problem that can occur with a GA is known as premature convergence. If an individual that is more fit than most of its competitors emerges early

on in the course of the run, it may reproduce so abundantly that it drives down the population's diversity too soon, leading the algorithm to converge on the local optimum that that individual represents rather than searching the fitness landscape thoroughly enough to find the global optimum [85,81]. This is an especially common problem in small populations, where even chance variations in reproduction rate may cause one genotype to become dominant over others.

The most common methods implemented by GA researchers to deal with this problem all involve controlling the strength of selection, so as not to give excessively fit individuals too great of an advantage. Rank, scaling and tournament selection, discussed earlier, are three major means for accomplishing this; some methods of scaling selection include sigma scaling, in which reproduction is based on a statistical comparison to the population's average fitness, and Boltzmann selection, in which the strength of selection increases over the course of a run in a manner similar to the "temperature" variable in simulated annealing [81].

Premature convergence does occur in nature (where it is called genetic drift by biologists). This should not be surprising; as discussed above, evolution as a problem-solving strategy is under no obligation to find the single best solution, merely one that is good enough. However, premature convergence in nature is less common since most beneficial mutations in living things produce only small, incremental fitness improvements; mutations that produce such a large fitness gain as to give their possessors dramatic reproductive advantage are rare.

Finally, several researchers [84,85,83] advise against using genetic algorithms on analytically solvable problems. It is not that genetic algorithms cannot find good solutions to such problems; it is merely that traditional analytic methods take much less time and computational effort than GAs and, unlike GAs, are usually mathematically guaranteed to deliver the one ex+act solution. Of course, since there is no such thing as a mathematically perfect solution to any problem of biological adaptation, this issue does not arise in nature.

3.5 Examples of GAs

As the power of evolution gains increasingly widespread recognition, genetic algorithms have been used to tackle a broad variety of problems in an extremely diverse array of fields, clearly showing their power and their potential. This section will discuss some of the more noteworthy uses to which they have been put.

- Molecular biology
- Pattern recognition and data mining

3.5.1 Molecular biology

In living things, transmembrane proteins are proteins that protrude through a cellular membrane. Transmembrane proteins often perform important functions such as sensing the presence of certain substances outside the cell or transporting them into the cell. Understanding the behavior of a transmembrane protein requires identifying the segment of that protein that is actually embedded within the membrane, which is called the transmembrane domain. Over the last two decades, molecular biologists have published a succession of increasingly accurate algorithms for this purpose.

All proteins used by living things are made up of the same 20 amino acids. Some of these amino acids are hydrophobic, meaning they are repelled by water, and some are hydrophilic, meaning they are attracted to water. Amino acid sequences that are part of a transmembrane domain are more likely to be hydrophobic. However, hydrophobicity is not a precisely defined characteristic, and there is no one agreed-upon scale for measuring it [86] used genetic programming to design an algorithm to identify transmembrane domains of a protein. Genetic programming was given a set of standard mathematical operators to work with, as well as a set of Boolean amino-acid-detecting functions that return +1 if the amino acid at a given position is the amino acid they detect and otherwise return-1. A single shared memory variable kept a running count of the overall sum, and when the algorithm was completed, the protein segment was identified as a transmembrane domain if its value was positive. Given only these tools, would it

entail the creation of new information for a human designer to produce an efficient solution to this problem?

The solutions produced by genetic programming were evaluated for fitness by testing them on 246 protein segments whose transmembrane status was known. The best-of-run individual was then evaluated on 250 additional, out-of-sample, test cases and compared to the performance of the four best known human-written algorithms for the same purpose. The result: Genetic programming produced a transmembrane segment-identifying algorithm with an overall error rate of 1.6% - significantly lower than all four human-written algorithms, the best of which had an error rate of 2.5%. The genetically designed algorithm, which the authors dubbed the 0-2-4 rule, operates as follows:

Increment the running sum by 4 for each instance of glutamic acid (an electrically charged and highly hydrophilic) amino acid in the protein segment.

Increment the running sum by 0 for each instance of alanine, phenylalanine, isoleucine, leucine, methionine, or valine (all highly hydrophobic amino acids) in the protein segment.

Increment the running sum by 2 for each instance of all other amino acids.

If [(SUM - 3.1544)/0.9357] is less than the length of the protein segment, classify that segment as a transmembrane domain; otherwise, classify it as a nontransmembrane domain.

3.5.2 Pattern recognition and data mining

Competition in the telecommunications industry today is fierce, and a new term - "churn" - has been coined to describe the rapid rate at which subscribers switch from one service provider to another. Churn costs telecom carriers a large amount of money each year, and reducing churn is an important factor in increasing profitability. If carriers can contact customers who are likely to switch and offer them special incentives to stay, churn rates can be reduced; but no carrier has the resources to contact more than a small percent of its customers. The problem is therefore how to identify customers who are

more likely to churn. All carriers have extensive databases of customer information that can theoretically be used for this purpose; but what method works best for sifting through this vast amount of data to identify the subtle patterns and trends that signify a customer's likelihood of churning?

[89] Applied genetic algorithms to this problem to generate a set of if-then rules that predict the churning probability of different groups of customers. In their GA, the first generation of rules, all of which had one condition, was generated using a probabilistic induction technique. Subsequent generations then refine these, combining simple, single-condition rules into more complex, multi-condition rules. The fitness measure used an objective "interestingness" measure of correlation which requires no subjective input. The evolutionary data-mining algorithm was tested on a real-world database of 100,000 subscribers provided by a Malaysian carrier, and its performance was compared against two alternative methods: a multilayer neural network and a widely used decision-tree-based algorithm, C4.5. The authors state that their EA was able to discover hidden regularities in the database and was "able to make accurate churn prediction under different churn rates", outperforming C4.5 under all circumstances, outperforming the neural network under low monthly churn rates and matching the neural network under larger churn rates, and reaching conclusions more quickly in both cases. Some further advantages of the evolutionary approach are that it can operate efficiently even when some data fields are missing and that it can express its findings in easily understood rule sets, unlike the neural net.

3.6 Concluding Remarks

Even creationists find it impossible to deny that the combination of mutation and natural selection can produce adaptation. Nevertheless, they still attempt to justify their rejection of evolution by dividing the evolutionary process into two categories - "microevolution" and "macroevolution" - and arguing that only the second is controversial, and that any evolutionary change we observe is only an example of the first.

Now, microevolution and macroevolution *are* terms that have meaning to biologists; they are defined, respectively, as evolution below the species level and evolution at or above the species level. But the crucial difference between the way creationists use these terms and the way scientists use them is that scientists recognize that these two are fundamentally the same process with the same mechanisms, merely operating at different scales. Creationists, however, are forced to postulate some type of unbridgeable gap separating the two, in order for them to deny that the processes of change and adaptation we see operating in the present can be extrapolated to produce all the diversity observed in the living world.

However, genetic algorithms make this view untenable by demonstrating the fundamental seamlessness of the evolutionary process. Take, for example, a problem that consists of programming a circuit to discriminate between a 1-kilohertz and a 10-kilohertz tone, and respond respectively with steady outputs of 0 and 5 volts. Say we have a candidate solution that can accurately discriminate between the two tones, but its outputs are not quite steady as required; they produce small waveforms rather than the requisite unchanging voltage. Presumably, according to the creationist view, to change this circuit from its present state to the perfect solution would be "microevolution", a small change within the ability of mutation and selection to produce. But surely, a creationist would argue, to arrive at this same final state from a completely random initial arrangement of components would be "macroevolution" and beyond the reach of an evolutionary process. However, genetic algorithms were able to accomplish both, evolving the system from a random arrangement to the near-perfect solution and finally to the perfect, optimal solution. At no step of the way did an insoluble difficulty or a gap that could not be bridged turn up. At no point whatsoever was human intervention required to assemble an irreducibly complex core of components (despite the fact that the finished product does contain such a thing) or to "guide" the evolving system over a difficult peak. The circuit evolved, without any intelligent guidance, from a completely random and non-functional state to a tightly complex, efficient and optimal state. How can this *not* be a compelling experimental demonstration of the power of evolution?

It has been said that human cultural evolution has super ceded the biological kind - that we as a species have reached a point where we are able to consciously control our society, our environment and even our genes to a sufficient degree to make the evolutionary process irrelevant. It has been said that the cultural whims of our rapidly changing society, rather than the comparatively glacially slow pace of genetic mutation and natural selection, is what determines fitness today. In a sense, this may well be true.

But in another sense, nothing could be further from the truth. Evolution is a problem-solving process whose power we are only beginning to understand and exploit; despite this, it is already at work all around us, shaping our technology and improving our lives, and in the future, these uses will only multiply. Without a detailed understanding of the evolutionary process, none of the countless advances we owe to genetic algorithms would have been possible. There is a lesson here to those who deny the power of evolution, as well as those who deny that knowledge of it has any practical benefit. As incredible as it may seem, evolution *works*.

Chapter 4

Entropy-based Adaptive Genetic Algorithm for Learning Classification Rules

4.1 Introduction

Genetic algorithms have been successfully applied to a wide range of optimization problems including design, scheduling, routing, and control, etc. Data mining is also one of the important application fields of genetic algorithms. In data mining, GA can be used to either optimize parameters for other kinds of data mining algorithms or discover knowledge by itself. In this latter task the rules that GA found are usually more general because of its global search nature. In contrast, most other data mining methods are based on the rule induction paradigm, where the algorithm usually performs a kind of local search. The advantage of GA becomes more obvious when the search space of a task is large.

In this chapter a genetic algorithm approach has been applied for classification problems. First we use binary coding in which an individual solution candidate consists of a fixed number of rules. In each rule, k bits are used for the possible k values of a certain attribute. Continuous attributes are modified to threshold-based boolean attributes before coding. Rule consequent is not explicitly coded in the string, instead, the consequent of a rule is determined by the majority of training examples it matches.

Four important factors are considered in our evaluation functions. Error rate is calculated by the predicting results on the training examples. Entropy is used to measure the homogeneity of the examples that a certain rule matches. Rule consistency is a measure on the consistency of classification conclusions of a certain training example given by a set of rules. Finally, hole ratio is used to evaluate the percentage of training

examples that a set of rules does not cover. The researcher tried to include related information as complete as possible in the evaluation function so that the overall performance of a rule set can be better.

An adaptive asymmetric mutation operator is applied in our reproduction step. When a bit is selected to mutate, the inversion probability from 1-0 (0-1) is not 50% as usual. The value of this probability is asymmetric and self-adaptive during the running of the program. This is made to reach the best match of a certain rule to training examples. For crossover, two-point crossover is adopted in our approach.

The scholar has used three real databases to test our approach: Database A,B and C. He has compared our performance with four well-known methods from data mining, namely Induction Decision Trees (ID3) [96], ID3 with Boosting [98] Neural Networks, and Naive Bayes [93]. The appropriate state-of-the-art techniques are incorporated in these non-GA methods to improve their performances. The results show that our GA approach [100] outperformed other approaches on both the prediction accuracy and the standard deviation.

4.2 Our GA approach

In this section GA approach for classification problem has been presented. The key idea of the algorithm is general and should be applicable for various kinds of classification problems. Some parameter values used in the algorithm might be task dependent.

4.2.1 Individual's encoding

Each individual in the population consists of a fixed number of rules. In other words, the individual itself is a complete solution candidate. In our current implementation, we set this fixed number as 10 which well satisfy the requirement of our

testing databases. The antecedent of a certain rule in the individual is formed by a conjunction of n attributes, where n is number of attributes being mined. K bits will be used to stand for an attribute if this attribute has k possible values. Continuous attributes will be partitioned to threshold-based boolean attribute in which the threshold is a boundary (adjacent examples across this boundary differ in their target classification) that maximizes the information gain. Therefore two bits will be used for a continuous attribute. The consequent of a rule is not explicitly encoded in the string. In contrast, it is automatically given based on the proportion of positive/negative examples it matches in the training set. We will illustrate the encoding method by the following example.

Suppose our task has three attributes, and they have 4, 2, 5 possible values respectively. Then an individual in the population can be represented as following:

A1 A2 A3 A1 A2 A3 A1 A2 A3

0110 11 10110 1110 01 10011 …… 1100 11 01110

Rule 1 Rule 2 …… Rule 10

Ten rules are included in this individual. The architecture of each rule is same. We will use rule 1 to explain the meaning of encoding. In this example, the meaning of the antecedent of rule 1 is:

If (A1=value 2 OR value 3) AND (A2=value 1 OR value 2) AND (A3=value 1 OR value 3 OR value 4)

If all the bits belong to one attribute are 0s, it means that attribute can not equal to any possible value and hence this is meaningless. To avoid this, we add one step before the evaluation of the population. We will check each rule in each individual one by one, if the above case happens, we will randomly select one bit of that attribute and change it to one.

The consequent of a rule is not encoded in the string. It will be determined by the proportion situation of the training examples that rule matches. Suppose i is one of the classifications, the consequent of a rule will be i if $\frac{N_{matched\_i}}{N_{matched}} > \frac{N_{training\_i}}{N_{training}}$,

where $N_{matched\_i}$ is the number of examples whose classification is i and matched by that rule, $N_{matched}$ is the total number of examples that the rule matches; $N_{training\_i}$ is the number of training examples whose classification is i, $N_{training}$ is the total number of training examples.

For example, if the distribution of the positive examples and negative examples in the training set is 42% and 58%, and among the examples of rule 1 matches positive / negative examples are half to half, then the consequent of rule 1 should be positive because 0.5 > 0.42. Since the testing databases we use at this time don't have a very uneven distribution on the classification of training examples, in our current implementation we didn't specially consider the interestingness of rules but use this strategy to keep enough rules to match examples with minor classification. Our encoding method is not limited to two-category classification but applicable to multi target value problems.

4.2.2 Fitness function

It is very important to define a good fitness function that rewards the right kinds of individuals. The author has tried to consider affecting factors as complete as possible to improve the results of classification. Our fitness function is defined as following:

Fitness = Error rate + Entropy measure + Rule consistency + Hole ratio

Each part in the fitness function will be elaborated below.

1) Error rate :

It is well known that accuracy is the most important and commonly used measure in the fitness function as the final goal of data mining is to get good prediction results. Since our objective function here is minimization, we use error rate to represent this information. It is calculated as:

Error rate = percent of misclassified examples

If a rule matches a certain example, the classification it gives is its consequent part. If it doesn't match, no classification is given. An individual consists of a set of rules, the final classification predicted by this rule set is based on the voting of those rules that match the example. The classifier gives all matching rules equal weight. For instance, in an individual (which has ten rules here), one rule doesn't match, six rules give positive classification and three rules give negative classification on a training example, then the final conclusion given by this individual on that training example is positive. If a tie happens (i.e., four positive classifications and four negative classifications), the final conclusion will be the majority classification in the training examples. If none of the rules in the individual matches that example, the final conclusion will also be the majority classification in the training examples. The error rate measure of this individual is the percent of misclassified examples among all training examples.

2) Entropy measure :

Entropy is a commonly used measure in information theory. Originally it is used to characterize the (im)purity of an arbitrary collection of examples. In our implementation entropy is used to measure the homogeneity of the examples that a rule matches.

Given a collection S, containing the examples that a certain rule R matches, let P_i be the proportion of examples in S belonging to class i, then the entropy Entropy(R) related to this rule is defined as:

$Entropy(R) = -\sum_{i=1}^{n} (p_i \log_2 (p_i))$ where n is the number of target classifications.

While an individual consists of a number of rules, the entropy measure of an individual is calculated by averaging the entropy of each rule:

$Entropy(individual) = \frac{\sum_{i=1}^{N_R} Entropy(R_i)}{N_R}$ where N_R is number of rules in the individual (in our current implementation it is 10).

The rationale of using entropy measure in fitness function is to prefer those rules that match less examples whose target values are different from rule's consequent. High

accuracy does not implicitly guarantee the entropy measure is good because the final classification conclusion of a certain training example is based on the comprehensive results of a number of rules. It is very possible that each rule in the individual has a bad entropy measure but the whole rule set still gives the correct classification. Keeping low entropy value of an individual will be helpful to get better predicting results for untrained examples.

3) Rule consistency :

As stated in the above sections, the final predicted classification of a training example is the majority classification made by rules in an individual. Let's consider the following classifications made by two individuals on an example:

Individual a: six rules → +, four rules → -, final classification: +

Individual b: nine rules → +, one rule → -, final classification: +

We will prefer the second individual since it is less ambiguous. To address this rule consistency issue, we add another measure in the fitness function. The calculation is similar to the entropy measure. Let $P_{correct}$ be the proportion of rules in one individual whose consequent is same with the target value of the training example, then

$$Rule_{consistency}(individual) = -p_{correct}\log_2 p_{correct} - (1-p_{correct})\log_2(1-p_{correct})$$

We should notice that this formula will give the same rule consistency value when $p_{correct}$ and $(1\text{-}p_{correct})$ switch each other. Therefore a penalty will be given when $p_{correct}$ is smaller than 0.5. In this case $Rule_{consistency} = 2 - Rule_{consistency}$.

The above calculation is based on the predicting results for one training example. The complete measure of rule consistency of an individual should be averaged by the number of training examples.

4) Hole ratio :

The last element in the fitness function is the hole ratio. It is a measure of rule's coverage on training examples. Coverage is not a problem for traditional inductive learning methods like decision trees, since the process of creating trees guarantees that all

the training examples will be covered in the tree. However, this also brings a new problem that it may be sensitive to noise. GA approach does not guarantee that the generated rules will cover all the training examples. This allows flexibility and may be potentially useful for future prediction. In real implementation we still hope the coverage should reach a certain point. For instance, if a rule only matches one training example and its consequent is correct, the accuracy and entropy measure of this rule are both excellent but we do not prefer this rule because its coverage is too low.

In our fitness function the hole ratio equals to 1-coverage, in which the latter is calculated by the union of examples that are matched and also correctly predicted by the rules in an individual. Totally misclassified examples (not classified correctly by any rule in the individual) will not be included even though they are matched by some rules. The following is the formula to calculate the hole ratio for binary classification problem (positive, negative).

$Hole = 1 - \frac{\left|\bigcup_i P_i^+\right| + \left|\bigcup_i N_i^-\right|}{|S|}$ where P_i^+ stands for those examples whose target value is positive and classified as positive by at least one rule in the individual, N_i^- stands for those examples whose target value is negative and classified as negative by at least one rule in the individual. $|S|$ is the total number of training examples.

4.2.3 Adaptive asymmetric mutation

In our reproduction step, traditional bit inversion is used on mutation. However, we found many examples will not be matched if we keep number of 1's and 0's approximately equivalent in an individual (i.e., the inversion probability from 1-0 and 0-1 are both 50%). The learning process will become a majority guess if there are too many unmatched examples. Therefore, we put forward a strategy of adaptive asymmetric mutation in which the inversion probability from 1-0 (0-1) is self-adaptive during the

process of run. The asymmetric mutation biases the population toward generating rules with more coverage on training examples. The self-adaptation of inversion probability makes the optimal mutation parameter be automatically adjusted.

An adaptive simplex genetic algorithm is presented before [100] in which the percentage of simplex operator is self-adaptive during the process of run. Similar idea is adopted here that average of fitness is used as a feedback to adjust the inversion probability. The process of self-adaptation is described as following:

1) An initial inversion probability is set (e.g., 0.5 for 1-0). Use this probability on mutation to produce a new generation. Calculate the average fitness of this generation.

2) Randomly select the direction of changing this probability (increase, decrease). Modify the probability along that direction with a small amount (0.02 in our current implementation). Use the new probability to produce the next generation and calculate the average fitness of the new generation.

3) If the fitness is better (value is smaller), continue on this direction and the amount of change is:

$\Delta p = \max\{0.05, (1- fitness_{new} / fitness_{old}) * 0.1\}$ If the fitness is worse (value is larger), reverse the direction and the amount of change is:

$\Delta p = \max\{0.05, (fitness_{new} / fitness_{old} - 1) * 0.05\}$ Use the new probability to produce the next generation and calculate the average fitness of the new generation.

Repeat step 3 until the program ends.

4.3 The information of databases

The scholar has tested his approach on three real databases and compared his approach with four other traditional data mining techniques. This section will present the testing results.

1) Database A

This database concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. This database is interesting because there is a good mix of attributes ---- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are 15 attributes plus one target attribute. Total number of instances is 690.

2) Database B

This database saves 1984 United States Congressional Voting Records. The data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition). There are 16 attributes plus one target attribute. Total number of instances is 435 (267 democrats, 168 republicans).

3) Database C

This database concerns heart disease diagnosis. The data was provided by V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D. There are 14 attributes plus one target attribute. Total number of instances is 303.

4.3.1 The description of non-GA approaches

The researcher used four well-known methods from machine learning, namely Induction Decision Trees (ID3) [96], Decision Trees with Boosting [98], Neural Networks, and Naïve Bayes [93] to compare the performance of our improved GA. Appropriate state-of-the-art techniques have been incorporated in most of the non-GA methods to improve their performance. The following is a description of the non-GA approaches we used for the performance comparison studies.

1) Induction Decision Trees

The construction of a decision tree is divided into two stages. First, creating an initial, large decision tree using a set of training set. Second, pruning the initial decision tree, if applicable, using a validation set. Given a noise-free training set, the first stage will generate a decision tree that can classify correctly all examples in the set. Except that the training set covers all instances in the domain, the initial decision tree generated will over fit the training data, which then reduce its performance in the test data. The second stage helps alleviate this problem by reducing the size of the tree. This process has an effect in generalizing the decision tree that hopefully could improve its performance in the test data.

During the construction of an initial decision tree, the selection of the best attribute is based on either the information gain (IG) or gain ratio (GR). A binary split is applied in nodes with continuous-valued attributes. The best cut-off value of a continuous-valued attribute is locally selected within each node in the tree based on the remaining training examples. The tree's node expansion stops either when the remaining training set is homogeneous (e.g., all instances have the same target attribute values) or when no attribute remains for selection. The decision on a leaf resulting from the latter case is determined by selecting the majority of the target attribute value in the remaining training set.

Decision tree pruning is a process of replacing sub-trees with leaves to reduce the size of the decision tree while retaining and hopefully increasing the accuracy of tree's classification. To obtain the best result from the induction decision tree method, he varied the use of pruning algorithm to the initial decision tree. He has considered using the following decision trees pruning algorithms: critical-value pruning [92], minimum-error pruning [94], pessimistic pruning, cost-complexity pruning and reduced-error pruning [97].

2) Induction Decision Trees with Boosting

Decision Tree with Boosting is a method that generates a sequence of decision trees from a single training set by re-weighing and re-sampling the samples in the set [98]. Initially, all samples in the training set are equally weighted so that their sum is one. Once a decision tree has been created, the samples in the training set are re-weighted in such a way that misclassified examples will get higher weights than the ones that are easier to classify. The new samples weights are then renormalized, and next decision tree is created using higher-weighted samples in the training set. In effect, this process enforces the more difficult samples to be learned more frequently by decision trees. The trees generated are then given weights in accordance with their performance in the training examples (e.g., their accuracy in correctly classifying the training data). Given a new instance, the new instance class is selected from the maximum weighted average of the predicted class over all decision trees.

4) Neural Networks

Inspired in part by biological learning systems, neural networks approach is built from a densely interconnected set of simple units. Since this technique offers many design selections, we fixed some of them to the ones that had been well proven to be good or acceptable design choices. In particular, we use a network architecture with one hidden layer, the back-propagation learning algorithm [99], the delta-bar-delta adaptive learning rates [91], and the Nguyen-Widrow weight initialization [95]. Discrete-valued attributes fed into the networks input layer is represented as 1-of-N encoding using bipolar values to denote the presence (e.g., value 1) and the absence (e.g., value –1) of an attribute value. Continuous-valued attribute is scaled into a real value in the range [–1, 1]. We varied the design choices for batch versus incremental learning, and 1-of-N versus single network output encoding.

4) Naive Bayes

Naive Bayes classifier is a variant of the Bayesian learning that manipulates directly probabilities from observed data and uses these probabilities to make an optimal decision. This approach assumes that attributes are conditionally independent given a class. Based on this simplifying assumption, the probability of observing attributes' conjunction is the product of the probabilities for the individual attributes. Given an instance with a set of attribute-value pairs, the Naive Bayes approach will choose a class that maximizes the conditional probability of the class given the conjunction of attributes values. Although in practice the independence assumption is not entirely correct, it does not necessarily degrade the system performance [90].

The author of this study assumes that the values of continuous-valued attributes follow Gaussian distribution. Hence, once the mean and the standard deviation of these attributes are obtained from the training examples, the probability of the corresponding attributes can be calculated from the given attribute values. To avoid zero frequency count problem that can dampen the entire probabilities calculation, we use an m-estimate approach in calculating probabilities of discrete-valued attributes [93].

Chapter 5

Feature Selection in Data-Mining for Genetics Using Genetic Algorithm

5.1 Introduction

The first phase of our algorithm deals with isolating the very few relevant features from the large set. This is not exactly the classical feature selection problem known in Data mining as, in [108], for example around 50% of features are selected. Here, the scholar has the idea that less than 5% of the features have to be selected. But this problem is close from the classical feature selection problem, and he will use a genetic algorithm as he saw they are well adapted for problems with a large number of features [105,106,107].We present here the main characteristics and adaptations we made to deal with this particular feature selection problem. Our genetic algorithm has different phases. It proceeds for a fixed number of generations. A chromosome, here, is a string of bits whose size corresponds to the number of features. A 0 or 1, at position i, indicates whether the feature i, is selected (1) or not (0).

The genetic operators : These operators allow GAs to explore the search space. However, operators typically have destructive as well as constructive effects. They must be adapted to the problem.

Crossover : The researcher has used a Subset Size-Oriented Common Feature Crossover Operator (SSOCF) [2] which keeps useful informative blocks and produces offspring's which have the same distribution than the parents. Offsprings are kept, only if they fit better than the least good individual of the population. Features shared by the 2 parents are kept by offsprings and the non shared features are inherited by offsprings corresponding to the i^{th} parent with the probability $(n_i - n_c/n_u)$ where n_i is the number of selected features of the i^{th} parent, n_c is the number of commonly selected features across both mating partners and n_u is the number of non-shared selected features.

Mutation : The mutation is an operator which allows diversity. During the mutation stage, a chromosome has a probability p_{mut} to mutate. If a chromosome is selected to mutate, we choose randomly a number n of bits to be flipped then n bits are chosen randomly and flipped. In order to create a large diversity, he sets p_{mut} around 10% and n € [101, 104].

Selection : The scholar has implemented a probabilistic binary tournament selection. Tournament selection holds n tournaments to choose n individuals. Each tournament consists of sampling 2 elements of the population and choosing the best one with a probability p €, [0.5, 1].

5.2 Materials and Method

Specific Adaptations And Mechanisms: The chromosomal distance (A distance adapted to the problem) : The biologist experts indicate that a gene is correlated with its neighbors situated on the same chromosome at a distance smaller than σ equals to 20 CMorgan (a measure unit). So in order to compare two individuals, he creates a specific distance which is a kind of bit to bit distance where not a single bit i is considered but the whole window (i-σ, i+σ) of the two individuals are compared. If one and only one individual has a selected feature in this window, the distance is increased by one.

The fitness function: The fitness function he has developed refers to the support notion, for an association, which, in data mining, denotes the number of times an association is met over the number of times at least one of the members of the association is met.

The function is composed of two parts. The first one favour for a small support a small number of selected features because biologists have in mind that associations will be composed of few features and if an association has a bad support, it is better to consider less features (to have opportunity to increase the support). The second part, the most important (multiplied by 2), favour for a large support a large number of features because if an association has a good support, it is generally composed of few features and then he must try to add other features in order to have a more complete association. What is expected is to favour good associations (in term of support) with as much as features as possible. This expression may be simplified, but he lets it in this form in order to identify the two terms.

$$F = ((1-S) * (T/10-10*SF) /T)+2*(S*(T/10-10*SF)/T$$

Where :

$S = |(\alpha \cap \beta \cap \gamma \ldots)/(\alpha \cup \beta \cup \gamma \ldots)|$ where $\alpha,\beta,\gamma\ldots$ are the selected features,

T = Total number of features,

SF= Number of selected significant features.

Sharing: To avoid premature convergence and to .discover different good solutions (different relevant associations of features), he uses a niching mechanism. A comparison of such mechanisms has been done in [103]. Both crowding and sharing give good results and he chooses to implement the fitness sharing [102]. The objective is to boost the selection chance of individuals that lie in less crowded area of the search space. He uses

a niche count that measures of how crowded the neighbourhood of a solution is. The distance δ is the chromosomal distance adapted to our problem presented before. The fitness of individuals situating in high concentrated search space regions is degraded and a new fitness value is calculated and used, in place of the initial value of the fitness, for the selection.

The sharing fitness $f_{sh}(i)$ of an individual i, where n is the size of the population,α_{sh}=1 and σ_{sh}=3), is:

$$f_{sh}(i) = \frac{F(i)}{\sum_{j=1}^{n} Sh(\delta(I_i, I_j))}$$

$$\text{where : } Sh(\delta(I_i, I_j)) = \begin{cases} 1 - \left(\frac{\delta(I_i, I_j)}{\sigma sh}\right)^{\alpha_{sh}} & \text{if } \delta(I_i, I_j) < \sigma_{sh} \\ 0 & \text{else} \end{cases}$$

Random Immigrant: Random Immigrant is a method that helps to maintain diversity in the population. It should also help to avoid premature convergence [101]. He uses random immigrant as follows: if the best individual is the same during N generations, each individual of the population, whose fitness is under the mean, is replaced by a new randomly generated individual. When random immigrant is done, he adds an extra step in our algorithm.

The clustering phase : use of K-Means algorithm : The k-means algorithm is an iterative procedure for clustering which requires an initial classification of the data. The k-means algorithm proceeds as follows : it computes the center of each cluster, then computes new partitions by assigning every object to the cluster whose center is the closest (in term of the Hamming distance) to that object. This cycle is repeated during a given number of iterations or until the assignment has not changed during one iteration [104].

Since the number of features is now very small, he has implemented a classical k-means algorithm widely used in clustering, and to initialise the procedure he randomly selects initial centers.

5.3 Experiments

Validation on an artificial database : In order to validate the method, experiments have been first executed on an artificial database constructed to be close to real problems, which is a public one. he knows, by construction, the relevant associations of features which can influence the disease. Results to obtain are associations α+β+δ and ϒ+ω. This test base is composed of 491 features and 165 pairs of individuals.

For ten runs, he wanted to know how many times associations were discovered by the GA. He has noted the following results in Association Table.

Association Table :

Association	α+β	α+δ	β+δ	ϒ+ω
	100%	50%	20%	10%

Table 5.1

k-means algorithm – Occurrence

(□ □ □)	(□ □ □)	(□ □)	(□ □)	(□ □ □ □)	(□+□+□)
(□ □)	(□ □ □)	(□ □ □)	(□)	(□ □ □)	(□ □)
4	1	1	2	1	1

Table 5.2

The first phase was able to discover real interactions of locus. Some of them are more difficult than others to find. Then, we ran the k-means algorithm with the results of the GA. We gave 11 features selected by the GA, instead of the initial 491 to the k-means algorithm.

The k-means algorithm helps us to discover associations genes-genes and genes-environmental factors. We have experimented the classical k-means algorithm without any feature selection. The execution time was very large (over 7500 minutes) and results can not be interpreted (we didn't know which were the features involved in the disease) so the feature selection phase is required. With the feature selection, the time of execution of k-means had decreased to 1 minute and the results are exploitable.

The author presents here clusters obtained with k=2 and their number of occurrences (k-means algorithm-occurrence-Table2). This Table-2 shows that the k-means algorithm using results of the GA, is able to construct clusters very closely related to the solution presented in results of the workshop. Moreover this solution has been exactly found 4 times over 10 of executions.

Experiments are executed on real data provided by the General Hospital at Chennai for the study of diabetes. The dataset is composed of 1179 pairs of individuals who have diabetes. The biologists take 3552 points of comparison and 2 covariables (age at onset, the age of the individual when diabetes was diagnosed and BMI Body Mass Index, which is a measure of obesity). The data are confidential and we can not give any biological results here, but here are some aspects:

First, he tested the performances of the method in term of size of problems it can deal with. It appeared that the execution time grows linearly with the number of features and the number of pairs. So the method is able to deal with very large size problem. Then, he ran several times the algorithm. The genetic algorithm managed to select interesting features and The k - means algorithm was able to class pairs of individuals according to these features and to confirm interesting associations of features.

Chapter 6

Rule Discovery in Data Mining using Genetic Algorithms

6.1 Genetic Algorithms

Meta-heuristic inspired by genetics consists of combining the best solutions so far and changing them slightly. This incorporates Darwinian evolutionary theory with sexual reproduction. Specifically, for a population P of chromosomes the following operators are applied:

- Selection deals with the probabilistic survival of the fittest, in that more fit chromosomes are chosen to survive, where fitness is a comparable measure of how well a chromosome solves the problem at hand.
- Crossover takes individual chromosomes from P and combines them to form new ones.
- Mutation alters the new solutions so as to add stochasticity in the search for better solutions.

A variant known as elitist GA ensures the most fit solution survives intact, leading to a higher degree of exploitation rather than exploration.

6.1.1 Selection

When selecting chromosomes from P, different methods are available. The first one, roulette selection, chooses a chromosome with probability proportional to its fitness:

$$\mathbf{P_{r(c)} = Fitness(c) / \sum_{c^1 \in P} Fitness(c^1)}$$

This can be likened to assigning chromosomes to slices of a roulette wheel, sized according to their fitnesses, and then selecting the winning slice after spinning the wheel. Another method is known as tournament selection in which two chromosomes are chosen at random, then according to a fixed probability, either the more or the less fit chromosome is selected.

Since both these methods are probabilistic, it is possible - though unlikely - that only the worst fit chromosomes are selected. And it is likewise possible that the same chromosome will be chosen every time.

6.1.2 Algorithm

The algorithm is as follows:

- Initialize the chromosomes in P with random values.
- Until a termination criterion is met (e.g. number of iterations, stagnation of fitness, or fitness-threshold) repeat the following:
  - Create a temporary and initially empty set, P_s.
  - Select $n < \backslash P \backslash$ chromosomes from P and add them to P_s. If n = 0 then P_s is completely filled by the crossover in the next step.
  - Select [1]Ap- pairs of chromosomes. Chop up each of the chromosomes, and combine the pieces to form the two offspring that are added to P_s.
  - Mutate some of the chromosomes in P_s, i.e. alter features at random.
  - Replace the population with the newly created: P <- P_s.

6.1.3 Binary Coded

A natural way of coding the chromosomes is by binary strings. This makes the transition from genetics theory to computer implementation rather straightforward. For example, the crossover of the two strings $x = {}_{Xl}x_2$ and $y = {}_{Vl}y_2$ may be the two strings x_{lV2} and ${}_{Vl}x_2$ - known as single-point crossover - provided \x\ = \y\ and they are split at the same point: $_{Xl}\ = _{Vl}\$. Mutation can be done by flipping one or more randomly chosen bits in the string.

Although binary encoding is of course used for the implementation of all types of values in digital computers, there are more appropriate codings for the chromosomes. Since the genetic operators can then be better tailored to the search-space, chosing another coding scheme may improve actual performance, as well as easen the understanding of how the algorithm finds the improved solutions, aiding further development of the optimization scheme. Because the original binary realization spawned some theory[1] formalizing the validity of the algorithm as a meta-heuristic search-procedure, other coding schemes were initially disregarded by some researchers. Abstraction is however one of the main tools of not only mathematics and computer science, but most sciences - not to say of self-organization and life itself.

6.1.4 Natural Coding

One such coding arises naturally when considering function optimization in continuous search-spaces, where the chromosomes are now vectors of floating point numbers instead of bit-strings.

The selection procedure depends only on the fitness and not the specific coding, but new operators are needed for crossover and mutation. A plethora of different operators are available [112], for example flat crossover simply selects a value between the two chromosomes to be crossed - note that this only generates one offspring though, where the algorithm of section 1.2 assumes two. Another method is simple crossover which is similar to the crossover described in section 1.3.

Regarding mutation, the simplest is perhaps random mutation in which one or more values within the chromosome is simply chosen at random. A number of variations on this seek to choose the number more sensibly; for example adding a bipolar random number covering only a fraction of the search-space, may seem more appropriate.

6.1.5 Observations

Notice how the population size remains constant, whereas populations in nature have a tendency to grow unless the environment prohibits it. One reason for keeping it constant is a matter of computational resources, the proper analogy to nature would be for each chromosome to execute on its own computer. Another reason is stability of convergence.

Furthermore, there is only one species and one race. Implementing race in a GA would be similar to having subsets of P with more similar chromosomes, also called niching. Species is more difficult as the chromosomes are normally rather precisely sized candidate solutions. But it would be interesting to allow the evolution of chromosomes with different sizes (both smaller and larger), provided there is a sensible way of using them on the original problem.

A suggestion would be to use a window: If the chromosome is bigger, then choose only a portion of it matching the problem size. If it is smaller, only solve a certain part of the problem of size equal to the chromosome - with the fitness also somehow reflecting that only a part of the problem was solved. The actual growing or shrinking may be built into the mutation operator, and crossover between different species could be disallowed. Alternatively, the crossover operator could instead split the two chromosomes at different points, thus creating new chromosomes of in equal length.

This is somewhat similar to the artificial immune system described in [110] (p. 231), in which germs are bit-strings. The protective agents, so called antigens, are bit-strings of arbitrary length, offering protection against any substring they encode. The antigens may also learn from each other, and it turns out that the information they encode gets compressed, so that substrings recognizing germs start to overlap.

However, it may very well be that the larger chromosomes provide no improvement over simply increasing the number of fixed-size chromosomes.

6.2 Knowledge Discovery & Data Mining

The process of retrieving data from storage, extracting useful knowledge from it and delivering an abstract analysis to the user, is known as knowledge discovery. One purpose is to predict values from incomplete information, for example given the temperature the past few days what will it be tomorrow, or given a customer matching a certain profile, what kind of goods will she buy.

The actual extraction of a prediction model is known as data mining, and a common approach is to split the input data D into two mutually exclusive and exhaustive sets, the training set: T C D and its complement, the test set: T = D\T. The data mining algorithm must build its model based on the training set alone, and the accuracy of the model is then evaluated by using it on the test set.

The model is also assessed according to its comprehensibility - is the model understandable by a human – as well as interestingness. These are of less import if the model is exclusively used in machine learning, the automated adaption of some agent or algorithm to its surroundings.

6.2.1 Pre- and Post-Processing

To the logistical end of pre-processing there is the possible need for integration of several data sources with appropriate mappings. Then the data may also need some cleaning, so as to avoid too noisy or unreliable information. More intruiging is the discretization of values into fewer classes (e.g. salary into: Low, medium, and high) which is claimed to produce more comprehensible knowledge [111], aswell as the selection of perspective on D, supposedly because the algorithm may otherwise find inaccurate knowledge (such as the name and cleaning-assistant relationship described below).

After data mining, the model may be simplified for comprehensibility and interestingness, the latter seems subjective at first, but may be objectively evaluated by finding the outlying patterns - which essentially is another data mining task known as clustering.

6.2.2 Over fitting

Over fitting is the over influencing of the prediction model to anomalies in the training set T, that are not representative for the entire data set D. This may occur because the model is developed too much, when too few samples are present in T, or if they are too noisy. The extremity of this is memorizing in which an uncovered pattern is so specific that it only covers a single instance. In the worst case, the entire training set may be memorized, rendering the model useless. The inverse is known as underfitting, where the model is too general to express essential subtleties of the data-set.

Although [111] mentions as an example that a person's credit can not be deduced from his/her name, even though the data mining algorithm may in fact find such a pattern, the purpose of knowledge discovery is precisely to uncover previously unknown patterns in vast data sets.

6.3 Rule Discovery

The data mining task of classification revolves around discovering rules of the form:

IF < antecedent THEN <consequent>

Where the consequent is a finite set whose elements are called classes. That is, the task is to decide what class a single target attribute will be, given a number of predicting attributes. Naturally, the target attribute can not occur in the antecedent of a rule.

For example the credit of a person may be discovered to be good, if he/she has a job and a positive balance on her bank-account:

IF ((has.job) AND (positive-balance)) THEN (good-credit)

There are direct generalizations of the classification task, such as dependence modelling and association rules. But more interesting is data mining of first order Horn clauses [109], or predicate logic, that discovers relationships with variables, for example the concepts of family relationships.

Chapter 7

Optimisation of Significant GA factors using Statistical tools

7.1 Introduction

Genetic algorithms (GA) are an optimisation tool that computationally emulates the process of evolution [118]. GA have been applied across a broad front - from traditional and cutting-edge optimisation in engineering and OR to such non-traditional areas as drug design, financial predictions and data mining. [115]. The motives for such a wide-spread usage of GA are rooted in the limitations of traditional optimisation and OR methods. The great advantage of Genetic Algorithm is that they work well over a broader class of problems with reasonable effectiveness. These features of quality and efficiency, of the GA optimisation have attracted many practitioners. However, it has been found that the efficiency of Genetic Algorithms in finding good solutions strongly depends on the difficulty of the problem under consideration. As Genetic Algorithms turn to larger or harder problems, it often happens that solution times significantly increase or the quality of final solutions decreases. This necessitates the development of competent Genetic Algorithms - GA that solve hard problems quickly, reliably and accurately [115].

Since, GA make relatively few assumptions about the solution space, and because the GA search is guided only by the values of the fitness function, finding a solution to an optimisation problem may require an extremely large number of function evaluations. To make the GA solutions suitable, it is often necessary to speed up the search/optimisation process. There are a number of existing efficiency improvement schemes available,

including (1) ad hoc GA factor tuning, (2) space/time utilisation (parallelisation), (3) meta-GA optimisation, (4) adaptation tuning, and (5) hybridisation [113,116]. In this paper he proposes a new approach to making Genetic Algorithms competent with respect to their ability to quickly and reliably find accurate solutions to the problem under investigation. Our approach is based on statistical analysis of the GA factors that significantly affect the method's performance and on building a regression model in terms of these factors.

In Section 2 he define the test problem on which he show how his approach can be used. Section 3 gives the details of his experiments. The results of statistical tuning of GA factors are presented in Section 4, and Section 5 concludes this paper with the discussion on the meaning and limitations of these results.

7.2 Genetic Algorithms applied to a optimal control problem

The composition of an optimal schedule for anti-cancer chemotherapy treatment is a non-linear optimal control problem that is subject to contradictory constraints. The detailed description of the problem is given in [117]. For single-drug treatments the solutions to the problem may be expressed as vectors $\mathbf{c} = \{C_k : k = \overline{1,10}\}$ of 10 discrete doses each of which is represented by a 4-bit binary string. All solutions are evaluated according to the following function:

$$F(\mathbf{C}) = \sum_{k=1}^{10} w_k C_k - \phi_1 \max^2\left\{\sum_{k=1}^{10} C_k \Delta t - C_{\text{cum}}, 0\right\} - \phi_2 \sum_{k=1}^{10} \max^2\{y_{\min} - y(t_k), 0\} \quad (1)$$

where ϕ_1 and ϕ_2 are penalty coefficients, analogous to the Lagrange multipliers, which strengthen or loosen the influence of the cumulative dose and the maximum tumour size constraints; $y(t_k)$ characterizes the tumour size at time t_k. If a solution does not break any of the constraints then he will call it feasible. All feasible solutions comprise a feasible region in the solution space, which will be the goal of our search. The competence of Genetic Algorithms applied to our problem is measured by the time necessary for GA to find at least one feasible solution.

This time depends on a number of factors and is characterised by a random variable, the descriptions of which follow.

7.2.1 Measure of GA efficiency

In order to analyse the performance of Genetic Algorithms he introduced a random variable that reflects how well the GA works. The author will characterize the GA by the number of generations Ψ, which are required in order to reach the feasible region in the solution space. The feasible region consists of the solutions that do not break any constraints of chemotherapeutic treatment. As with all random variables, Ψ can be described by a probability distribution. In this section he tried to approximate the distribution of Ψ by a known statistical distribution. To find a suitable approximation means to be able to adjust the parameters of a known distribution in such a way that the distribution curve will fit the experimental data obtained for a given random variable[114].

To obtain the distribution list for the random variable Ψ, 50 independent runs were performed of the GA program, which implements the search for a feasible solution to the problem (1). These runs produced a positively skewed distribution, for approximation of which many common statistical distributions have been tried. The attempt to fit the Waybill distribution produced the best result (Kolmogorov-Smirnov test is safely passed: $|F_{obs}(x)-F_{exp}(x)|_{max} = 0.0512 << F_{crit} = 0.352$).

7.2.2 Factors affecting the efficiency of Genetic Algorithms

Genetic Algorithms possess explorative features, characterized by the population size and the rate of mutation, and exploitative features, characterized by the type of selection used, by the probability of crossover and by the crossover operator itself. However, the mere presence of these features does not guarantee the efficiency of GA search. It is necessary to strike the right balance between explorative and exploitative features of Genetic Algorithms.

These are quantified by the following factors:

the penalty coefficients ϕ_1 and ϕ_2 in (1);

the probability of mutation (ϕ_3);

the probability of crossover (ϕ_4);

parameters defining selection pressure (ϕ_5, ϕ_6);

a number of breaking points (ϕ_7).

The effects of varying these seven factors on the efficiency of Genetic Algorithms are different. In the following section he develops a methodology which allows identifying GA factors that most significantly affect GA performance and optimising the values of such factors.

7.3 Methodology of Statistical Analysis

Having established the measure of GA efficiency and having specified the factors that affect it, we can now attempt to find an expression for this measure:

$$\beta = \beta(\phi_i) \quad i = \overline{1,l} \qquad (2)$$

where l is the number of GA factors that significantly affect the performance. Then the problem of GA efficiency improvement can be confined to the task of finding the factor values ϕ_i' that optimize the performance measure (2). This task will be accomplished in three steps. First of all, a screening experiment will be conducted reducing the number l of GA factors that need to be included into the model (2). Only significant factors, variation of which noticeably (in a statistical sense) affect the performance, will remain in the model. After the screening experiment, a second-order regression model of β in terms of significant factors will be obtained. Finally, standard calculus techniques will be applied to the regression model and the optimal values of significant GA factors will be calculated. He will start with the screening experiment.

7.3.1 Screening experiment

In order to ascertain how many of the specified GA factors significantly affect the performance, he utilizes the fractional factorial design of a screening experiment. Factorial designs allow the study of multiple factors in the same experiment and the assessment of the manner in which these factors interact. If no interactions are present, then the effect of each factor can be evaluated with the same efficiency as if the whole experiment had been devoted entirely to that factor [119]. The analysis of the screening experiment is presented in Figure 2 where the significance of each GA factor is shown in the form of a histogram (t-statistics of significant factors stretch over the critical level):

As can be seen from the below figure only two GA factors, viz. the probabilities of mutation (ϕ_3) and crossover (ϕ_4), significantly affect the performance of this Genetic Algorithm. The effects of insignificant factors are indistinguishable from the effect which might be caused by random errors of performance measurements; thus, the other factors will be excluded from further analysis. The significant factors ϕ_3 and ϕ_4, on the other hand, will be examined more thoroughly in the next section, where a regression model will be constructed in terms of these factors.

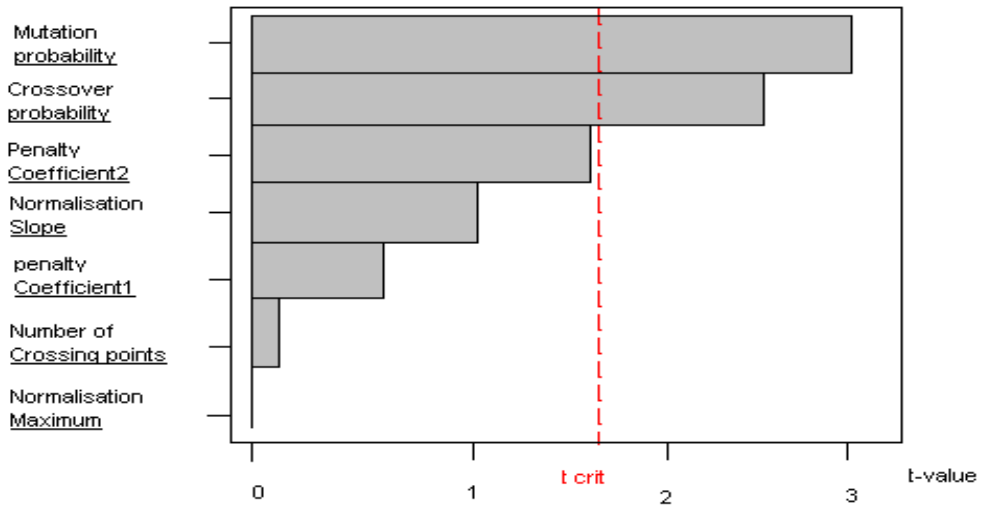

Figure 7.1 : MINITAB analysis of the screening experiment

Significance of GA factor

Response in waybill scale parameter – beta

Level of significance = 10

7.3.2 Regression model

When studying continuous factors, it is interesting to find conditions (values of the factors) that lead to a particular response, usually minimum or maximum. The responses of an experiment when considered as a function of the possible values of the factors form a response surface, which is usually expressed in the form of a regression model.

The performance measure is obtained by repeating each GA run 25 times and fitting a Waybill curve to the resultant observations. The scale parameter β of the fitted curve is used as the response attributed to each setting of the factors ϕ_3 and ϕ_4. The analysis of the experimental data yields the following regression model:

$$\beta = 1723 - 5874\phi_3 - 3731\phi_4 + 7565\phi_3^2 + 2281\phi_4^2 + 4683\phi_3\phi_4 \qquad (3)$$

As may be deduced from this formula, the curvature of the regression model indicates that within the experimental ranges of factors ϕ_3 and ϕ_4 there is a certain setting of these factors at which the performance measure β attains its minimum. Small values of β imply that Genetic Algorithms require a small number of generations Ψ to find a feasible solution to the problem under investigation and hence perform an efficient search. In order to estimate the optimal values of the significant factors he has applied conventional calculus techniques, which yield the optimal values of mutation ($\phi_3' = 0.198$) and crossover ($\phi_4' = 0.614$) factors.

Chapter 8

Results and Conclusions

8.1 Entropy-based Adaptive Genetic Algorithm for Learning Classification Rules

For each database, *k*-fold cross-validation method is used for evaluation. In this method, a data set is divided equally into *k* disjoint subsets. *k* experiments are then performed using *k* different training-test set pairs. A training-test set pair used in each experiment is generated by using one of the *k* subsets as the test set, and using the remaining subsets as the training set. Given *k* disjoint subsets, for example, the first experiment takes the first subset for the test set, and uses the second subset through the *k*th subset for the training set. The second experiment uses subset 1 and subset 3 through subset *k* for the training set; and takes subset 2 for the test set, and so on. All results from a particular database are averaged along with its variance over *k* experiment runs.

Based on their size, the credit database and voting database are partitioned into 10 disjoint subsets; the heart database is partitioned into 5 subsets. Table 1, 2, 3 show the performance comparison results of different approaches on these three databases. For decision trees with and without boosting, The author presents only the best experimental results after varying the data splitting methods as well as the pruning algorithms described earlier. Similarly, results from neural networks approach are selected from the ones that provide the best performance after varying the use of different network output encoding and *batch* versus *incremental* learning methods.

	Our GA approach	Decision trees (IG, Min-Err)	Decision trees with boosting (RG, Cost-Com, 21 trees)	Neural networks (1-of-N,batch learning)	Naive Bayes
Run 1	85.77	82.69	84.23	84.23	61.15
Run 2	84.23	79.62	81.15	81.15	73.46
Run 3	84.23	84.23	85.77	85.77	79.62
Run 4	87.31	85.77	85.77	84.23	76.54
Run 5	81.15	76.54	76.54	79.62	70.38
Run 6	84.23	82.69	82.69	82.69	75.00
Run 7	79.62	76.54	79.62	79.62	68.85
Run 8	82.69	81.15	82.69	81.15	78.08
Run 9	85.77	81.15	84.23	82.69	71.92
Run 10	81.76	83.24	86.18	81.76	70.00
Average	83.68	81.36	82.89	82.29	72.50
Standard deviation	2.37	3.06	3.08	2.03	5.36

Table 8.1 : The comparison results on the prediction accuracy and standard deviation (%) of database A.

In the above table, the decision tree is generated using information gain for data splitting and minimum-error pruning. Decision trees with boosting generates 21 different decision trees, each is constructed by using gain ratio for data splitting and cost-complexity pruning algorithm. Batch learning and 1-of-N output encoding are used in neural networks.

	Our GA approach	Decision trees (IG, Min-Err)	Decision trees with boosting (RG, Cost-Com, 21 trees)	Neural networks (1-of-N,batch learning)	Naive Bayes
Run 1	90.35	90.35	90.35	88.02	88.02
Run 2	92.67	88.02	92.67	92.67	85.70
Run 3	92.67	92.67	92.67	92.67	88.02
Run 4	90.35	90.35	92.67	92.67	83.37
Run 5	90.35	90.35	90.35	88.02	83.37
Run 6	92.67	92.67	92.67	92.67	83.37
Run 7	90.35	88.02	88.02	90.35	88.02
Run 8	92.67	90.35	90.35	90.35	85.70
Run 9	95.00	95.00	92.67	90.35	85.70
Run 10	90.83	88.75	92.92	90.03	80.42
Average	91.79	90.65	91.54	90.86	85.17
Standard deviation	1.59	2.23	1.67	1.46	2.53

Table 8.2 : The comparison results on the prediction accuracy and standard deviation (%) of database B.

In table 8.2, the best results from both decision trees with and without boosting are obtained from using information gain for data splitting and reduced-error pruning algorithm. Only three decision trees are needed in the decision trees with boosting.

	Our GA approach	Decision trees (IG, Red-Err)	Decision trees with boosting (IG, Red-Err, 3 trees)	Neural networks (1-of-N, batch learning)	Naive Bayes
Run 1	84.83	76.36	83.14	76.36	84.83
Run 2	78.05	72.97	79.75	79.75	74.66
Run 3	76.36	71.27	71.27	69.58	78.05
Run 4	83.14	71.27	78.05	84.83	78.05
Run 5	78.33	61.67	66.67	76.67	75.00
Average	80.14	70.71	75.77	77.44	78.12
Standard deviation	3.64	5.46	5.54	5.66	4.08

Table 8.3 : The comparison results on the prediction accuracy and standard deviation (%) of database C.

In table 8.3, the best results from neural networks are obtained from applying incremental learning and 1-of-N network output encoding. From the above results he can see that our GA approach outperformed other approaches on both the average prediction accuracy and the standard deviation. The advantage of our GA approach becomes more obvious on heart database, which is most difficult to learn among the three. During the process of running, he also found that the training accuracy and testing accuracy of GA approach are basically in the same level, while the training accuracy is often much higher than testing accuracy for other approaches. This proves that GA approach is less sensitive to noise and might be more effective for future prediction

8.2 Feature Selection in Data-Mining for Genetics Using GA

The *k*-means algorithm helps us to discover associations genes-genes and genes-environmental factors. The scholar has experimented the classical *k*-means algorithm without any feature selection. The execution time was very large (over 7500 minutes) and results can not be interpreted (he didn't know which were the features involved in the disease) so the feature selection phase is required. With the feature selection, the time of execution of *k*-means had decreased to 1 minute and the results are exploitable.

The scholar present here clusters obtained with k=2 and their number of occurrences (k-means algorithm-occurrence-Table2). This Table-2 shows that the *k*-means algorithm using results of the GA, is able to construct clusters very closely related to the solution presented in results of the workshop. Moreover this solution has been exactly found 4 times over 10 of executions.

Experiments are executed on real data provided by the General Hospital at Chennai for the study of diabetes. The dataset is composed of 1179 pairs of individuals who have diabetes. The biologists take 3552 points of comparison and 2 covariables (age at onset, the age of the individual when diabetes was diagnosed and BMI Body Mass Index, which is a measure of obesity). The data are confidential and can not give any biological results here, but here are some aspects:

First, performances of the method has been tested in term of size of problems it can deal with. It appeared that the execution time grows linearly with the number of features and the number of pairs. So the method is able to deal with very large size problem. Then, he ran several times the algorithm. The genetic algorithm managed to select interesting features and The *k* - means algorithm was able to class pairs of individuals according to these features and to confirm interesting associations of features.

References

[1]. E. H. L. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines: A Stochastic Approach to combinatorial optimization and Neural computing, John Wiley & Sons Ltd, 1989.

[2]. R. Agrawal et. al., Fast discovery of association rules, In: Advances in Knowledge Discovery & Data Mining, UM. Fayyad et al. (Eds.), AAAI Press, Menlo Park, pages 307 .328, 1996.

[3]. T. Bäck, Evolutionary Algorithms in Theory and Practice. , Oxford University Press, New York, USA, 1996.

[4]. T. Bäck, D. B. Fogel, and T. Michalewicz, editors, Basic Algorithms and Operators., volume 1 of Evolutionary Computation., Institute of Physics Publishing, Bristol and Philidelphia, 1999.

[5]. T. T. Bihn, U. Korn, Multi-criteria Control System Design Using an Intelligent Evolution Strategy with Dynamical Constraints Boundaries., Institute of Automation, University of Magdeburg, Germany, 1997.

[6]. H. K. Birru, K. Chellapilla, R Sathyanarayan, Local Search Operators in Fast Evolutionary Programming., IEEE Press, pages 1506 - 1513, 1999.

[7]. H. K. Birru, K. Chellapilla, R. Sathyanarayan, Effectiveness of Local Search Operators in Evolutionary Programming., Genetic Programming 98, Madison, WI, pages 753-761, 170 1998.

[8]. C. L. Blake, C. J. Merz, UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences,. http://www.ics.uci.edu/~mlearn/MLRepository.html., September 2004, 1998.

[9]. F. Boscheti, M. C. Dentith, R. D. List, Inversion of seismic refraction data using genetic algorithms., Geophysics, Volume 61, number 6, pages 1715 - 1727, 1996.

[10]. R. Brits, Niching strategies for particle swarm optimization., Submitted M.Sc. thesis, University of Pretoria, Pretoria, 2002.

[11]. C. J. Burgess, A. G. Chalmers, The Optimisation of Irregular Multiprocessor Computer Architectures using Genetic Algorithms., Baltzer Journals, August 2, 1999.

[12]. E. Cant-Paz, C. Kamath, On the Use of Evolutionary Algorithms in Data mining., Published in Data Mining : A Heuristic Approach.., Idea Group Publishing, 2001.

[13]. M. J. Cavaretta, K. Chellapilla, Data Mining using Genetic Programming : The Implications of Parsimony on Generalization Error., In Proceedings of the 1999 Congress on Evolutionary Computation, pages 1330 - 1337, IEEE-press, 1999.

[14]. P. Cheeseman, M. Self, J. Kelly, J. Stutz, W. Taylor, D. Freeman, Bayesian Classification., In Proceedings of American Association of Artificial Intelligence (AAAI), 7th Annual Conference on A.I., pages 607 - 611, Morgan Kaufmann, 1988.

[15]. K. Chellapilla, Evolving Computer Programs Without Subtree Crossover, IEEE Transactions on Evolutionary Computation, volume 1, number 3, pages 209 - 216, 1997.

[16]. L. Chen, Z. Qin, J. S. Liu, Exploring Hybrid Monte Carlo in Bayesian Computation., In Proceedings ISBA 2000, ISBA and Eurostat, 2000.

[17]. P. Clark, T. Niblett, The CN2 Induction Algorithm, Machine Learning, pages 261- 283, 1989.

[18]. C. A. Coella, A Comprehensive Survey of Evolutionary-Based Multi-objective Optimization Techniques., Laboratorio Nacional de Informatica Avanzada, Xalapa,Veracruz, México, Engineering Design Centre, University of Plymouth, Plymouth, UK,1999.

[19]. C. A. Coella, A. D. Christiansen, Multi-objective Optimization of Trusses using Genetic Algorithms., Engineering Design Centre, University of Plymouth, Plymouth, UK

[20] C. Cotta, E. Alba, J. M. Troya, Stochastic Reverse Hillclimbing and Iterated Local Search., IEEE Press, pages 1558 - 1565, 1999.

[21]. C. R. Darwin, On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life., Murray, London, 1859.

[22]. L. Davis, Applying Adaptive Algorithms to Epistatic Domains. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 162 - 164, 1985.

[23] R. Dawkins, The Selfish Gene., Oxford University Press, 1990.

[24]. I. De Falco, A. Della Cioppa, E. Tarantino, Discovering interesting classification rules with genetic programming., Applied Soft Computing, Elsevier Science Publishing, Amsterdam, pages 257 - 269, 2002.

[25]. J. Denzinger, T. Offermann, On Cooperation between Evolutionary Algorithms and other Search Paradigms., IEEE Press, pages 2317 - 2324, 1999.

[26]. R. Eberhart, P. Simpson, R. Dobbins, Computational Intelligence PC tools., AP Professional, 1996.

[27]. L. J. Eshelman, J. D. Schaffer, Real-coded genetic algorithms and interval schemata., In Foundations of Genetic Algorithm II, pages 187 - 202, Morgan Kaufmann, San Mateo, CA, USA, 1990.

[28]. S. Forrest, M. Mitchell, Relative Building-block Fitness and the Building-block Hypothesis., In Foundations of Genetic Algorithms II, Morgan Kaufmann, San Mateo, CA, 1993.

[29]. J. M. Garibaldi, E. C. Ifeachor, Application of Simulated Annealing Fuzzy Model Tuning to Umbilical Cord Acid-Base Interpretation, IEEE-FS Transactions On Fuzzy Systems,Vol. 7, No.1, page 72, February 1999.

[30]. D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning., Addison-Wesley, MA, 1989.

[31]. D. E. Goldberg, S Voessner, Optimizing global-local search hybrids., In GECCO'99 volume 1, pages 220 - 228, Morgan Kaufmann, San Francisco, 1999.

[32]. M. Gorges-Schleuter, ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy., In Proceedings of the Third International Conference on Genetic Algorithms, pages 422 - 427, Mogan Kaufmann, 1989.

[33]. R. Hanson, J. Stutz, P. Cheeseman, Bayesian Classification with Correlation and Inheritance., In Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, August, pages 692 - 698, Morgan Kaufmann, 1991.

[34]. R. Hanson, J. Stutz, P. Cheeseman, Bayesian Classification Theory., In Technical Report FIA-90-12-7-01, NASA Ames Research Center, Artificial Intelligence Branch, May, 1991.

[35]. R. J. W. Hodgson., Memetic Algorithms and the Molecular Geometry Optimisation Problem., In Proceedings of the 2000 Congress on Evolutionary Computation (CEC2000), San Diego, CA., pages 625 - 632, 2000.

[36]. J. H. Holland, Adaptation in Natural and Artificial Systems., Ann Arbor, MI, University of Michigan Press, 1975.

[37]. A. Imada, Downhill Walk from the Top of a Hill by Evolutionary Programming., IEEE Press pages 1414 - 1418, 1999.

[38]. H. Ishibuchi, M. Nii, K. Tanaka, Linguistic Rule Extraction from Neural Networks for High-dimensional Classification Problems., Complexity International, volume 6,Department of Industrial Engineering, Osaka Prefecture University, Japan, In Procedings of Complex Systems .98 (Sydney, Australia) pages 210 - 218, 1998.

[39]. H. Ishibuchi, T. Yamamoto, Effects of Three-Objective Genetic Rule Selection on the Generalization Ability of Fuzzy Rule-Based Systems, Proceedings of Second International Conference on Evolutionary Multi-Criterion Optimization, pages 608 - 622, 2003.

[40]. H. Ishibuchi, T. Yamamoto, Fuzzy Rule Selection by Data Mining Criteria and Genetic Algorithms, The Genetic and Evolutionary Computation Conference (New York, USA), pages 399 - 406, 2002.

[41]. C. Z. Janikow, Z. Michalewicz, An Experimental Comparison of Binary and Floating point Representations in Genetic Algorithms, In Proceedings of the 4th International Conference on Genetic Algorithms, pages 31 - 36. Morgan Kaufmann, San Diego, CA, USA, 1991.

[42]. U. Johansson, R. König, L. Niklasson, The Truth is In There - Rule Extraction from Opaque Models Using Genetic Programming, In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Symposium Conference, Valerie Barr, Zdravko Markov (Eds.) Miami Beach, Florida, USA, AAAI Press, 2004.

[43]. N. Kasabov, On-line learning, reasoning, rule extraction and aggregation in locally optimised evolving fuzzy neural networks, In Neurocomputing, 41 (2001), pages 25 . Elsevier, 2001.

[44]. J. D. Knowles, D. W. Corne, M-PAES: A Memetic Algorithm for Multi-objective Optimization., In Proceedings of the Congress on Evolutionary Computation (CEC00), pages 325 - 332, IEEE Press, Piscataway, NJ, 2000.

[45]. J. D. Knowles, D. W. Corne, The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multi-objective Optimisation., Department of Computer Science, University of Reading, UK, 1999.

[46]. J. R. Koza, Genetic Programming: On Programming Computers by means of Natural Selection and Genetics., MIT Press, Cambridge, MA, 1992.

[47]. R. Ladner, F. E. Petry, M A Cobb, Fuzzy Set Approaches to Spatial Data Mining of Association Rules, Transactions in GIS, Volume 7, Issue 1, page 123, January 2003.

[48] .J. Lewis, E. Hart, G Ritchie, A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems., In Parallel Problem Solving from Nature – PPSN V., A.E. Eiben, T. Back, M. Schoenauer and H-P. Schwefel (editors). Springer-Verlag, pages 139 - 148, 1998.

[49]. K-H. Liang, X. Yao, C. Newton, Combining Landscape Approximation and Local Search in Global Optimisation., Congress on Evolutionary Computation, IEEE Press, pages 1514 - 1520, 1999.

[50]. T. Loveard, V. Ciesielski, Representing Classification Problems in Genetic Programming., In Proceedings of the 2001 Congress on Evolutionary Computation, CEC2001, IEEE Press, 2001.

[51]. E. Mayr, Animal Species and Evolution., Bellknap, Cambridge, MA, 1963.

[52] J. Mayer, J. Mayer, Stochastic Linear Programming Algorithms: A Comparison Based BIBLIOGRAPHY 176 on a Model Management System (Optimization Theory and Applications)., T&F STM , 1998.

[52] R. E. Marmelstein, Gary B. Lamont, Pattern Classification using a Hybrid Genetic Program - Decision Tree Approach., Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Dayton, 1998.

[53] P. Merz, B. Freisleben, A Comparison of Memetic Algorithms, Tabu Search, and Ant-Colonies for the Quadratic Assignment Problem., IEEE Press, pages 2063 - 2070, 1999.

[54] M. Mirmehdi, P. L. Palmer, J. Kittler, Optimising the Complete Image Feature Extraction Chain., The Third Asian Conference on Computer Vision, Volume II, pages 307 - 314, Springer Verlag, January, 1998.

[55] T. M. Mitchell, Machine Learning., McGraw-Hill, USA, 1997.

[56] P. Moscato, Memetic algorithms: a short introduction., McGraw-Hill Ltd., UK, 1999.

[57] P. Moscato, An introduction to population approaches for optimization and hierarchical objective functions: The role of tabu search., Annals of Operations Research, 41(1-4), pages 85 - 121, 1993.

[58] H. Mühlenbein, Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization., In Proceedings of the Third International Conference on Genetic Algorithms, pages 416 - 421, Mogan Kaufmann, 1989.

[59] H. Mühlenbein, How Genetic Algorithms really work. Part I: Mutation and Hillclimbing., In Parallel Problem Solving from Nature, Elsevier Science Publishers, Amsterdam, 1992.

[60] M. G. Norman, P. Moscato, A Memetic Approach for the Travelling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems., In Proceedings of the International Conference on Parallel Computing and Transputer Applications, Amsterdam, IOS Press, pages 177 - 186, 1992.

[61] U-M. O'Reilly, F. Oppacher, The Troubling Aspects of a Building Block Hypothesis for Genetic Programming., In Foundations of Genetic Algorithms, 3, L. Darrell Whitley, Michael D. Vose (editors), Morgan Kaufmann, 1995.

[62] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data., Kluwer Academic Publishers, Dordrecht, 1991.

[63] J. R. Quinlan, C4.5: Programs for Machine Learning., Morgan Kaufmann, San Mateo, CA, 1993.

[64] N. J. Radcliffe, P. D. Surry, Formal Memetic Algorithms., University of Edinburgh, In Evolutionary Computing: AISB Workshop, T. Fogarty (Editors), Springer-Verlag, 1994.

[65] D. Rodich, A.P. Engelbrecht, A Hybrid Exhaustive and Heuristic Rule Extraction Approach., In Development and Practice of Artificial Intelligence Techniques, V.B. Bajic, D. Sha (editors), pages 25 - 28, Proceedings of the International Conference on Artificial Intelligence, Durban, South Africa, 1999.

[66] C. Roos, T. Terlaky, J.-Ph. Vial, Theory and Algorithms for Linear Optimization: An Interior Point Approach., John Wiley & Sons, 1st edition, 1997.

[67] S. E. Rouwhorst, A. P. Engelbrecht, Searching the Forest: A Building Block Approach to Genetic Programming for Classification Problems in Data Mining., Submitted M.Sc thesis, Department of Mathematics and Informatics, Vrije Universiteit Amsterdam, The Netherlands. Research performed at University of Pretoria, South Africa, 2000.

[68] S. E. Rouwhorst, A. P. Engelbrecht, Searching the Forest: Using Decision Trees as Building Blocks for Evolutionary Search in Classification Databases., In Proceedings of the 2000 Congress on Evolutionary Computation, CEC2000, Vol. 1, pages 633 - 638, IEEE Press, 2000.

[69] B. Sánchez, Sergio, Learning with nearest neighbour classifiers, Doctoral Thesis, Universitat Politècnica de Catalunya, Chapter 9, 2003.

[70] R. Setiono, W. K. Leow, J. Y-L. Thong, Opening the neural network blackbox: An algorithm for extracting rules from function approximating neural networks., In Proceedings of ICIS 2000, International Conference on Information Systems, Brisbane,Australia, December, 2000.

[71] R. Setiono, J. Y-L. Thong, C. Yap, Symbolic rule extraction from neural networks: An application to identifying organizations adopting IT., In Information and Management, Vol. 34, No. 2, pages 91 - 101, 1998.

[72] J. R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, School of Computer Science, Carnegie Mellon University, Pittsburgh, August 1994.

[73] R. Storn, Sytem Design by Constraint Adaptation and Differential Evolution, IEEE Trans. on Evolutionary Computation, Vol. 3, No. 1, pages 22 - 34, 1999.

[74] C. Thornton, The Building Block Fallacy., Complexity International, volume 4, published by Monash University, Cognitive and Computing Sciences, University of Sussex, UK, 1997.

[75] F. van den Bergh, An Analysis of Particle Swarm Optimizers., Submitted Ph.D. thesis, University of Pretoria, Pretoria, 2001.

[76] D. Whitley, Modelling Hybrid Genetic Algorithms., In Genetic Algorithms in Engineering and Computer Science., G.Winter, et al., John Wiley, pages 191 - 201, Chichester, 1995.

[78] B. Woodford, N. Kasabov, Rule Insertion and Rule Extraction from Evolving Fuzzy Neural Networks: Algorithms and Applications for Building Adaptive, Intelligent Expert Systems, In Proceedings of the FUZZ-IEEE'99 International Conference on Fuzzy Systems, BIBLIOGRAPHY 179 August, Seoul, Korea, pages 1406 - 1411, 1999.

[79] E. Zitzler, J. Teich, S. S. Bhattacharyya, Optimizing the Efficiency of Parameterized Local Search within Global Search: A Preliminary Study., In Proceedings of the Congress on Evolutionary Computation, San Diego, 2000.

[80] Fleming, Peter and R.C. Purshouse. "Evolutionary algorithms in control systems engineering: a survey." (2002).

[81] Mitchell, Melanie. *An Introduction to Genetic Algorithms.* MIT Press, 1996.

[82] Koza, Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers, 2003.

[83] Haupt, Randy and Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, 1998.

[84] Holland, John. "Genetic algorithms." *Scientific American*, July 1992.

[85] Forrest, Stephanie. "Genetic algorithms: principles of natural selection applied to computation." *Science*, (1993).

[86] Koza, John, Forest Bennett, David Andre and Martin Keane. *Darwinian Invention and Problem Solving.* Morgan Kaufmann Publishers, 1999.

[87] Coello, Carlos. "An updated survey of GA-based multiobjective optimization techniques." *ACM Computing Surveys*, (2000).

[88] Graham-Rowe, Duncan. "Radio emerges from the electronic soup." *New Scientist*,
(2002).

[89] Au, Wai-Ho, Keith Chan, and Xin Yao. "A novel evolutionary data mining algorithm with applications to churn prediction." *IEEE Transactions on Evolutionary Computation*, (December 2003)

[90] Domingos, P. and Pazzani, M. (1997) On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning Comparison between Traditional, 29, 103-130.

[91] Jacobs, R.A. (1988) Increased Rates of Convergence Through Learning Rate Adaptation. Neural Networks, 1(4): 295-307.

[92] Mingers, J. (1987) Expert Systems – Rule Induction with Statistical Data. Journal of the Operational Research Society, 38, 39-47.

[93] Mitchell, Tom. (1997) Machine Learning. New York: McGraw-Hill.

[94] Niblett, T. (1986) Constructing Decision Trees in Noisy Domains. In I. Bratko and N. Lavrac (Eds). Progress in Machine Learning. England: Sigma Press.

[95] Nguyen, D. and Widrow, B. (1990) Improving the Learning Speed of Two-Layer Networks by Choosing Initial Values of the Adaptive Weights. International Joint Conference on Neural Networks, San Diego, CA, III:21-26.

[96] Quinlan, J.R. (1986) Induction of Decision Trees. Machine Learning, 1, 81-106.

[97] Quinlan, J.R. (1987) Simplifying Decision Trees. International Journal of Man-Machine Studies, 27, 221-234.

[98] Quinlan, J. R. (1996) Bagging, Boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 725-730.

[99] Rumelhart, D. E., Hinton, G.E., and William, R. J. (1986) Learning Representations by Back-Propagation Error. Nature, 323:533-536.

[100] Yang, L. and Yen, J. (2000) An adaptive simplex genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), July 2000, pp. 379.

[101] C. Bates Congdon, 2002 *A comparison of genetic algorithm and other machine learning systems on a complex classification task from common disease research*. Ph.D thesis, University of Michigan.

[102] J. Horn, D.E. Goldberg, and K. Deb,1994 Implicit Niching in a learning classifier system : Nature's way. *Evolutionary Computation*, 2(1):37–66..

[103] S.W. Mahfoud, 2004 *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois.

[104] N. Monmarch´e, M. Slimane, and G. Venturini.2001 Antclass : discovery of cluster in numeric data by an hybridization of an ant colony with the kmeans algorithm. Technical Report 213, Ecole d'Ing´enieurs en Informatique pour l'Industrie (E3i), Universit´e de Tours.

[105] M. Pei, E.D. Goodman, and W.F. Punch,1997 Feature extraction using genetic algorithms. Technical report, Michigan State University : GARAGe.

[106] M. Pei, E.D. Goodman, W.F. Punch, and Y. 1995 Ding. Genetic algorithms for classi.cation and feature extraction. In *Annual Meeting : Classi.cation Society of North America.*

[107] M. Pei, M. Goodman, and W.F. Punch,1997 Pattern discovery from data using genetic algorithm. In *Proc of the .rst Paci.c-Asia Conference on Knowledge Discovery and Data Mining*, Feb.

[108] J. Yang and V. Honoavar. *Feature Extraction,2005 Construction and Selection : A data Mining Perspective*, chapter 1: Feature Subset Selection Using a Genetic Algorithm, pages 117–136. H. Liu and H. Motoda Eds, massachussetts : kluwer academic publishers edition.

[109] Machine Learning, Tom M. Mitchell McGraw Hill 1997 ISBN 0-07-042807-7

[110] Swarm Intelligence, James Kennedy, Russell C. Eberhart Morgan Kaufmann Publishers 2001 ISBN 1-55860-595-9

[111] A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery Alex A. Freitas Pontificia Universidade Catolica do Parana http://www.cs.kent.ac.uk/people/staff/aaf/

[112] Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioral Analysis F. Herrera, M. Lozano, J. L. Verdegay Artificial Intelligence Review 12 pp. 265-319 Kluwer Academic Publishers 1998 http://decsai.vgr.es/~herrera/

[113] Baeck T, Fogel D and Michalewicz Z (eds.), 1997. Handbook of Evolutionary Computation. Oxford University Press.

[114] Devore, J. L. (1995) Probability and Statistics for Engineering and the Sciences. 4th edn. Duxbury Press.

[115] Goldberg, D. E. (1999) Genetic and Evolutionary Algorithms in the Real World. IlliGAL Report #99013, University of Illinois at Urbana-Champaign, IL, U.S.A.

[116] Hatta K, Matsuda K, Wakabayashi S and Kiode T (1997). *On-the-Fly Crossover Adaptation of Genetic Algorithms. Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems*. Innovations and Applications: University of Strathclyde – Glasgow, U.K. September 1997, pp. 197-202.

[117] McCall, J. and Petrovski, A. (1999) *A Decision Support System for Cancer Chemotherapy using Genetic Algorithms*. Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation: Vienna, Austria. IOS Press, ISBN 90 5199 474 5, pp. 65-70.

[118] Mitchell M. (1996). *An introduction to Genetic Algorithms*. Cambridge, Mass. – MIT Press.

[119] Wadsworth HM (1998). *Handbook of Statistical Methods for Engineers and Sci-*
entists. McGraw-hill, 2nd edn.

LIST OF PAPERS PUBLISHED BASED ON THIS RESEARCH

INTERNATIONAL JOURNALS

1. V.N.Rajavarman, Dr.S.P.Rajagopalan, "Comparison between Traditional Data Mining Techniques and Entropy - based Adaptive Genetic Algorithm for Learning Classification Rules"- Inter National Journal of Soft computing 2(4):555-561 2007.

2. V.N.Rajavarman, Dr.S.P.Rajagopalan, "Feature Selection in Data-Mining for Genetics Using GA" - Journal of Computer Science 3(9):723-725, 2007 ISSN 1549-3636.

3. V.N.Rajavarman, Dr.S.P.Rajagopalan, "Optimisation of Significant GA factors using Statistical tools" - International Journal of computing and Applications, Vol.4, No. 1,June 2009, pp.7-12

NATIONAL JOURNALS

1. V.N.Rajavarman, Dr.S.P.Rajagopalan, "Genetic Algorithms-An Overview"- Acta ciencia Indica, vol. XXXIII M, No.4,1615(2007)

2. V.N.Rajavarman, Dr.S.P.Rajagopalan, " Rule discovery in Data Mining using Genetic Algorithms" - National journal of ACCST in Vol.VII,2-April 2009

CONFERENCES

1. "Learning Classification Rules using Genetic Algorithm" – paper presented at the National Conference on BUSINESS INTELLIGENCE IN ENTERPRISE RESOURCE PLANNING ON BIERP'07 29TH March, 2007 organized by SREE SASTHA Institute of Engineering and Technology, Chennai, pp 15 in the Proceedings book (2007).

2. "Effective Information Retrieval using Genetic Programming"—paper presented at the National Conference on COMPUTATIONAL INTELIGENCE held on 22nd February 2007 organized by BHARATH UNIVERSIRTY, Chennai.

3. "Approximate Description of Decision Classes in Data Mining Sets" – paper presented at the National Conference on BUSINESS

INTELLIGENCE IN ENTERPRISE RESOURCE PLANNING ON BIERP'07 29TH March, 2007 organized by SREE SASRHA Institute of Engineering and Technology ,Chennai, pp 20 in the Proceedings book (2007).

4. "Localized Network layer Protocol in Wireless Sensor Networks" – at the National Conference on RECENT ISSUES IN NETWORK GALAXY held on 12th and 13th February 2007 organized by R.M.D Engineering College, Chennai.

5. Software testing with KITSS" –paper presented at the National Conference on RECENT TRENDS IN SCIENCE AND TECHNOLOGY HELD ON 26th and 27th May 2005 organized by Dr. M.G.R. DEEMED UNIVERSITY, Chennai.

VITAE

V.N.Rajavarman received the B.Sc. Computer Science (1990) and M.Sc. Computer science (1992) degree from Bharathidasan University, Tiruchirapalli, Tamil Nadu, India. He has received M.E. Information Technology from Vinayaga

Mission Research Foundation, Selam, Tamil Nadu in 2007. He worked as lecturer in Computer Science in AVVM Sri Poondi Pushpam College, Thanjavur, Tamil Nadu during the academic year 1992-1993. From 1994 to 2000 he served as lecturer in Computer Science in Ponnaiyah Ramajayam College of Science, Thanjavur, Tamil Nadu. He joined as Operations Manager in Information Fusion (India) Pvt. Ltd., Ashok Nagar, Chennai, Tamil Nadu in year 2001-2003. He worked as Sr.Lecturer in Computer Science in Jaya Engineering College, Thiruninravur, Chennai, Tamil Nadu during the academic year 2003-2004. He has been working as Assistant professor in the department of Computer Science and Engineering from 2004 and also a Ph.D. scholar at Dr. M.G.R Educational Research Institute, Maduravoyal, Chennai 600 095, Tamil Nadu, India. His research areas and interests are Genetic Algorithm, Data mining and Rule Discovery.

Email : nrajavarman2003@gmail.com
n_rajavarman2003@yahoomail.com
Phone (o) : India – 044-23782186
(r) : India – 044-32009510
Mobile : 9381051320

8.3 Rule Discovery in Data Mining using Genetic Algorithms

Using GAs for classification the scholar again needs a representation of the chromosomes, reasonable operators, as well as the fitness function that measures how well the discovered rules work.

First off, if a chromosome encodes only one rule, it is known as a Michigan model, if an entire set of rules is encoded, it is a Pittsburgh model. Since a normal GA has a tendency to converge its entire population to a single point, it is necessary to run a Michigan model several times to discover a set of rules - or use a niching method as described in section 1.5.

Regardless of encoding of the rules, there is the question of how to decide the consequent, and measure fitness. As always, there are several solutions; one is simply to encode the consequent in the chromosome making it subject to evolution also. Another is to choose the class that has more samples satisfying the antecedent, or even maximizes fitness. The fitness itself can be defined as a ratio of how many classes were correctly predicted, as well as how many were not predicted, as in [111]:

$$\textbf{Fitness(c) = TP / A . TP /(TP + FN)}$$

Where A is the number of samples satisfying the given antecedent, the true positives (TP) is the number of samples satisfying both the antecedent and consequent, and the false negatives (FN) is the number of samples not satisfying the antecedent, but satisfying the consequent. The left-hand (TP/A) is obviously maximized by over fitting and memorizing, where the right-hand side counters this. If comprehensibility is also desired, the fitness function may also take this into account, e.g. by favoring simpler rules.

8.4 Optimisation of Significant GA factors using Statistical tools

In order to verify that the factors estimates obtained from the regression model really cause the decrease in the number of generations required to reach a feasible region in the space of potential solutions has been run a confirmation experiment. In the confirmation run the mutation and crossover factors have been set to their optimal levels and Genetic Algorithms have been run 25 times to obtain a sample with the size sufficient for statistical inference. The comparative results of the confirmation experiment and those of an average GA run prior to the tuning procedure are listed below in Table 8.1

	Scale parameter β	**Mean value**	**Median value**
Average GA	167.10	150.51	144.2
Confirmation experiment	27.69	28.96	21.0

TABLE 8.4 : *Tuned vs. untuned GA*

The results of the confirmation run give a Waybill *scale parameter* estimate of 27.69. It shows that he has managed to tune significant factors in such a way that the computational time has been reduced substantially even in comparison with an average run (let alone the worst case). Moreover, from the sample of the results of the confirmation run he can estimate the following important characteristics (**mean**(Ψ)=28.96 and **median**(Ψ) = 21.00), which distinctively show the success of the statistical inference and regression analysis with respect to modifying the distribution of Ψ. With such distribution of experimental results he does not have to run the GA program for more than 30 generations since the 95% confidence interval for the median value is [13.0, 24.0]. If a feasible solution has

not been found during that time then it is easy to restart the process and obtain a solution from successive runs.

It is known that one serious disadvantage of evolutionary optimisation in general and Genetic Algorithms in particular is that it is a time consuming procedure. However, by tuning certain factors, which determine the properties of the optimisation process, it is possible to noticeably improve the efficiency of GA. Here, a novel approach to GA efficiency improvement has been developed based on statistical identification of significant factors and on optimisation of their values. As he has shown, this approach is capable of attaining a substantial improvement in the search speed of Genetic Algorithms (the number of generations required to find a feasible region in the solution space has been reduced from ~170 to ~30). Moreover, statistical modeling of GA performance provides the data, on the basis of which it is possible to predict the likely time of search completion and, as a consequence, the safe restarting time in each particular case.

However, the statistical approach is not a panacea for determining the optimal values of GA factors. There exist situations when due to either a large random variation in experimental responses or due to an inappropriate choice of factor domains the screening experiment cannot reliably identify the significant factors; then the consequent regression analysis becomes meaningless. What one may attempt to do in that case is to modify the factor ranges using an intuitive approach and to repeat the experiment. If this attempt fails, the meaning is that GA performance cannot be controlled by varying factors. Therefore, the process of factor tuning should be abandoned. Still, the merit of the statistical approach is that it necessitates the evaluation of much fewer factor settings (in comparison, for example, with meta-GA optimisation) to establish the fact that GA performance is not affected by the factors' values.

8.5 Future Research

In the case of feature selection in data mining for genetics using GAs , the algorithms seems to be robust and it is able to isolate interesting associations. The biologist can study these associations and then validate them.

With regard to rule discovery in data mining using genetic algorithms, the population size has been taken constant, but populations in nature have a tendency to grow. Further research can be considered how GAs works in different population sizes.

Regarding optimization of significant GA factors using statistical tools can be extend to the situations when due to either a large random variation in experimental responses or due to an inappropriate choice of factor domains the screening experiment cannot reliably identify the significant factors

In general, research work may be under taken to develop on variants of GAs to tailor some specific tasks. GAs frame work for two major data mining tasks, namely classification and generalized rule induction. The framework specifies a Genetic Programming(GP) phenotype and a GP genotype for these two tasks. In addition, future work will involve the evaluation of algorithms developed under the proposed framework on real databases.